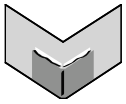
The Journal of Machine Learning Research
Volume 7
Print-Archive Edition

Pages 1–1384



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 7
Print-Archive Edition

The Journal of Machine Learning Research (JMLR) is an open access journal. All articles published in JMLR are freely available via electronic distribution. This Print-Archive Edition is published annually as a means of archiving the contents of the journal in perpetuity. The contents of this volume are articles published electronically in JMLR in 2006.

JMLR is abstracted in ACM Computing Reviews, INSPEC, and Psychological Abstracts/PsycINFO.

JMLR is a publication of Journal of Machine Learning Research, Inc. For further information regarding JMLR, including open access to articles, visit http://www.jmlr.org/.

JMLR Print-Archive Edition is a publication of Microtome Publishing under agreement with Journal of Machine Learning Research, Inc. For further information regarding the Print-Archive Edition, including subscription and distribution information and background on open-access print archiving, visit Microtome Publishing at http://www.mtome.com/.

ISSN 1532-4435 (print)
ISSN 1533-7928 (online)

JMLR Advisory Board

JMLR Web Master

JMLR Journal of Machine Learning Research Vol. 7 2006

Journal of Machine Learning Research 7 (2006) 1–30 Submitted 8/04; Revised 4/05; Published 1/06

Statistical Comparisons of Classifiers over Multiple Data Sets

Janez Demšar JANEZ.DEMSAR@FRI.UNI-LJ.SI
Faculty of Computer and Information Science
Tržaška 25
Ljubljana, Slovenia

Editor: Dale Schuurmans

Abstract

While methods for comparing two learning algorithms on a single data set have been scrutinized for quite some time already, the issue of statistical tests for comparisons of more algorithms on multiple data sets, which is even more essential to typical machine learning studies, has been all but ignored. This article reviews the current practice and then theoretically and empirically examines several suitable tests. Based on that, we recommend a set of simple, yet safe and robust non-parametric tests for statistical comparisons of classifiers: the Wilcoxon signed ranks test for comparison of two classifiers and the Friedman test with the corresponding post-hoc tests for comparison of more classifiers over multiple data sets. Results of the latter can also be neatly presented with the newly introduced CD (critical difference) diagrams.

Keywords: comparative studies, statistical methods, Wilcoxon signed ranks test, Friedman test, multiple comparisons tests

1. Introduction

Over the last years, the machine learning community has become increasingly aware of the need for statistical validation of the published results. This can be attributed to the maturity of the area, the increasing number of real-world applications and the availability of open machine learning frameworks that make it easy to develop new algorithms or modify the existing, and compare them among themselves.

In a typical machine learning paper, a new machine learning algorithm, a part of it or some new pre- or postprocessing step has been proposed, and the implicit hypothesis is made that such an enhancement yields an improved performance over the existing algorithm(s). Alternatively, various solutions to a problem are proposed and the goal is to tell the successful from the failed. A number of test data sets is selected for testing, the algorithms are run and the quality of the resulting models is evaluated using an appropriate measure, most commonly classification accuracy. The remaining step, and the topic of this paper, is to statistically verify the hypothesis of improved performance.

The following section explores the related theoretical work and existing practice. Various researchers have addressed the problem of comparing two classifiers on a single data set and proposed several solutions. Their message has been taken by the community, and the overly confident paired t-tests over cross validation folds are giving place to the McNemar test and 5×2 cross validation. On the other side, comparing multiple classifiers over multiple data sets—a situation which is even more common, especially when general performance and not the performance on certain specific

problem is tested—is still theoretically unexplored and left to various *ad hoc* procedures that either lack statistical ground or use statistical methods in inappropriate ways. To see what is used in the actual practice, we have studied the recent (1999-2003) proceedings of the International Conference on Machine Learning. We observed that many otherwise excellent and innovative machine learning papers end by drawing conclusions from a matrix of, for instance, McNemar's tests comparing all pairs of classifiers, as if the tests for multiple comparisons, such as ANOVA and Friedman test are yet to be invented.

The core of the paper is the study of the statistical tests that could be (or already are) used for comparing two or more classifiers on multiple data sets. Formally, assume that we have tested k learning algorithms on N data sets. Let c_i^j be the performance score of the j-th algorithm on the i-th data set. The task is to decide whether, based on the values c_i^j, the algorithms are statistically significantly different and, in the case of more than two algorithms, which are the particular algorithms that differ in performance. We will not record the variance of these scores, $\sigma_{c_i^j}$, but will only assume that the measured results are "reliable"; to that end, we require that enough experiments were done on each data set and, preferably, that all the algorithms were evaluated using the same random samples. We make no other assumptions about the sampling scheme.

In Section 3 we shall observe the theoretical assumptions behind each test in the light of our problem. Although some of the tests are quite common in machine learning literature, many researchers seem ignorant about what the tests actually measure and which circumstances they are suitable for. We will also show how to present the results of multiple comparisons with neat space-friendly graphs. In Section 4 we shall provide some empirical insights into the properties of the tests.

2. Previous Work

Statistical evaluation of experimental results has been considered an essential part of validation of new machine learning methods for quite some time. The tests used have however long been rather naive and unverified. While the procedures for comparison of a pair of classifiers on a single problem have been proposed almost a decade ago, comparative studies with more classifiers and/or more data sets still employ partial and unsatisfactory solutions.

2.1 Related Theoretical Work

One of the most cited papers from this area is the one by Dietterich (1998). After describing the taxonomy of statistical questions in machine learning, he focuses on the question of deciding which of the two algorithms under study will produce more accurate classifiers when tested on a given data set. He examines five statistical tests and concludes the analysis by recommending the newly crafted 5×2cv t-test that overcomes the problem of underestimated variance and the consequently elevated Type I error of the more traditional paired t-test over folds of the usual k-fold cross validation. For the cases where running the algorithm for multiple times is not appropriate, Dietterich finds McNemar's test on misclassification matrix as powerful as the 5×2cv t-test. He warns against t-tests after repetitive random sampling and also discourages using t-tests after cross-validation. The 5×2cv t-test has been improved by Alpaydın (1999) who constructed a more robust 5×2cv F test with a lower type I error and higher power.

Bouckaert (2003) argues that theoretical degrees of freedom are incorrect due to dependencies between the experiments and that empirically found values should be used instead, while Nadeau and Bengio (2000) propose the corrected resampled t-test that adjusts the variance based on the overlaps between subsets of examples. Bouckaert and Frank (Bouckaert and Frank, 2004; Bouckaert, 2004) also investigated the replicability of machine learning experiments, found the 5×2cv t-test dissatisfactory and opted for the corrected resampled t-test. For a more general work on the problem of estimating the variance of k-fold cross validation, see the work of Bengio and Grandvalet (2004).

None of the above studies deal with evaluating the performance of multiple classifiers and neither studies the applicability of the statistics when classifiers are tested over multiple data sets. For the former case, Salzberg (1997) mentions ANOVA as one of the possible solutions, but afterwards describes the binomial test with the Bonferroni correction for multiple comparisons. As Salzberg himself notes, binomial testing lacks the power of the better non-parametric tests and the Bonferroni correction is overly radical. Vázquez et al. (2001) and Pizarro et al. (2002), for instance, use ANOVA and Friedman's test for comparison of multiple models (in particular, neural networks) on a single data set.

Finally, for comparison of classifiers over multiple data sets, Hull (1994) was, to the best of our knowledge, the first who used non-parametric tests for comparing classifiers in information retrieval and assessment of relevance of documents (see also Schütze et al., 1995). Brazdil and Soares (2000) used average ranks to compare classification algorithms. Pursuing a different goal of choosing the optimal algorithm, they do not statistically test the significance of differences between them.

2.2 Testing in Practice: Analysis of ICML Papers

We analyzed the papers from the proceedings of five recent International Conferences on Machine Learning (1999-2003). We have focused on the papers that compare at least two classifiers by measuring their classification accuracy, mean squared error, AUC (Beck and Schultz, 1986), precision/recall or some other model performance score.

The sampling methods and measures used for evaluating the performance of classifiers are not directly relevant for this study. It is astounding though that classification accuracy is usually still the only measure used, despite the voices from the medical (Beck and Schultz, 1986; Bellazzi and Zupan, 1998) and the machine learning community (Provost et al., 1998; Langley, 2000) urging that other measures, such as AUC, should be used as well. The only real competition to classification accuracy are the measures that are used in the area of document retrieval. This is also the only field where the abundance of data permits the use of separate testing data sets instead of using cross validation or random sampling.

Of greater interest to our paper are the methods for analysis of differences between the algorithms. The studied papers published the results of two or more classifiers over multiple data sets, usually in a tabular form. We did not record how many of them include (informal) statements about the overall performance of the classifiers. However, from one quarter and up to a half of the papers include some statistical procedure either for determining the optimal method or for comparing the performances among themselves.

The most straightforward way to compare classifiers is to compute the average over all data sets; such averaging appears naive and is seldom used. Pairwise t-tests are about the only method used for assessing statistical significance of differences. They fall into three categories: only two methods

	1999	2000	2001	2002	2003
Total number of papers	54	152	80	87	118
Relevant papers for our study	19	45	25	31	54
Sampling method [%]					
cross validation, leave-one-out	22	49	44	42	56
random resampling	11	29	44	32	54
separate subset	5	11	0	13	9
Score function [%]					
classification accuracy	74	67	84	84	70
classification accuracy - *exclusively*	68	60	80	58	67
recall, precision...	21	18	16	25	19
ROC, AUC	0	4	4	13	9
deviations, confidence intervals	32	42	48	42	19
Overall comparison of classifiers [%]	53	44	44	26	45
averages over the data sets	0	4	6	0	10
t-test to compare two algorithms	16	11	4	6	7
pairwise t-test one vs. others	5	11	16	3	7
pairwise t-test each vs. each	16	13	4	6	4
counts of wins/ties/losses	5	4	0	6	9
counts of *significant* wins/ties/losses	16	4	8	16	6

Table 1: An overview of the papers accepted to International Conference on Machine Learning in years 1999—2003. The reported percentages (the third line and below) apply to the number of papers relevant for our study.

are compared, one method (a new method or the base method) is compared to the others, or all methods are compared to each other. Despite the repetitive warnings against multiple hypotheses testing, the Bonferroni correction is used only in a few ICML papers annually. A common non-parametric approach is to count the number of times an algorithm performs better, worse or equally to the others; counting is sometimes pairwise, resulting in a matrix of wins/ties/losses count, and the alternative is to count the number of data sets on which the algorithm outperformed all the others. Some authors prefer to count only the differences that were statistically significant; for verifying this, they use various techniques for comparison of two algorithms that were reviewed above.

This figures need to be taken with some caution. Some papers do not explicitly describe the sampling and testing methods used. Besides, it can often be hard to decide whether a specific sampling procedure, test or measure of quality is equivalent to the general one or not.

3. Statistics and Tests for Comparison of Classifiers

The overview shows that there is no established procedure for comparing classifiers over multiple data sets. Various researchers adopt different statistical and common-sense techniques to decide whether the differences between the algorithms are real or random. In this section we shall examine

several known and less known statistical tests, and study their suitability for our purpose from the point of what they actually measure and of their safety regarding the assumptions they make about the data.

As a starting point, two or more learning algorithms have been run on a suitable set of data sets and were evaluated using classification accuracy, AUC or some other measure (see Tables 2 and 6 for an example). We do not record the variance of these results over multiple samples, and therefore assume nothing about the sampling scheme. The only requirement is that the compiled results provide reliable estimates of the algorithms' performance on each data set. In the usual experimental setups, these numbers come from cross-validation or from repeated stratified random splits onto training and testing data sets.

There is a fundamental difference between the tests used to assess the difference between two classifiers on a single data set and the differences over multiple data sets. When testing on a single data set, we usually compute the mean performance and its variance over repetitive training and testing on random samples of examples. Since these samples are usually related, a lot of care is needed in designing the statistical procedures and tests that avoid problems with biased estimations of variance.

In our task, multiple resampling from each data set is used only to assess the performance score and not its variance. The sources of the variance are the differences in performance over (independent) data sets and not on (usually dependent) samples, so the elevated Type 1 error is not an issue. Since multiple resampling does not bias the score estimation, various types of cross-validation or leave-one-out procedures can be used without any risk.

Furthermore, the problem of correct statistical tests for comparing classifiers on a single data set is not related to the comparison on multiple data sets in the sense that we would first have to solve the former problem in order to tackle the latter. Since running the algorithms on multiple data sets naturally gives a sample of independent measurements, such comparisons are even simpler than comparisons on a single data set.

We should also stress that the "sample size" in the following section will refer to the number of data sets used, not to the number of training/testing samples drawn from each individual set or to the number of instances in each set. The sample size can therefore be as small as five and is usually well below 30.

3.1 Comparisons of Two Classifiers

In the discussion of the tests for comparisons of two classifiers over multiple data sets we will make two points. We shall warn against the widely used t-test as usually conceptually inappropriate and statistically unsafe. Since we will finally recommend the Wilcoxon (1945) signed-ranks test, it will be presented with more details. Another, even more rarely used test is the sign test which is weaker than the Wilcoxon test but also has its distinct merits. The other message will be that the described statistics measure differences between the classifiers from different aspects, so the selection of the test should be based not only on statistical appropriateness but also on what we intend to measure.

3.1.1 AVERAGING OVER DATA SETS

Some authors of machine learning papers compute the average classification accuracies of classifiers across the tested data sets. In words of Webb (2000), "it is debatable whether error rates in different domains are commensurable, and hence whether averaging error rates across domains is very mean-

ingful". If the results on different data sets are not comparable, their averages are meaningless. A different case are studies in which the algorithms are compared on a set of related problems, such as medical databases for a certain disease from different institutions or various text mining problems with similar properties.

Averages are also susceptible to outliers. They allow classifier's excellent performance on one data set to compensate for the overall bad performance, or the opposite, a total failure on one domain can prevail over the fair results on most others. There may be situations in which such behaviour is desired, while in general we probably prefer classifiers that behave well on as many problems as possible, which makes averaging over data sets inappropriate.

Given that not many papers report such averages, we can assume that the community generally finds them meaningless. Consequently, averages are also not used (nor useful) for statistical inference with the z- or t-test.

3.1.2 PAIRED T-TEST

A common way to test whether the difference between two classifiers' results over various data sets is non-random is to compute a paired t-test, which checks whether the average difference in their performance over the data sets is significantly different from zero.

Let c_i^1 and c_i^2 be performance scores of two classifiers on the i-th out of N data sets and let d_i be the difference $c_i^2 - c_i^1$. The t statistics is computed as $\overline{d}/\sigma_{\overline{d}}$ and is distributed according to the Student distribution with $N-1$ degrees of freedom.

In our context, the t-test suffers from three weaknesses. The first is commensurability: the t-test only makes sense when the differences over the data sets are commensurate. In this view, using the paired t-test for comparing a pair of classifiers makes as little sense as computing the averages over data sets. The average difference $\overline{d}$ equals the difference between the averaged scores of the two classifiers, $\overline{d} = \overline{c^2} - \overline{c^1}$. The only distinction between this form of the t-test and comparing the two averages (as those discussed above) directly using the t-test for unrelated samples is in the denominator: the paired t-test decreases the standard error $\sigma_{\overline{d}}$ by the variance between the data sets (or, put another way, by the covariance between the classifiers).

Webb (2000) approaches the problem of commensurability by computing the geometric means of relative ratios, $(\prod_i c_i^1/c_i^2)^{1/N}$. Since this equals to $e^{1/N\sum_i(\ln c_i^1 - \ln c_i^2)}$, this statistic is essentially the same as the ordinary averages, except that it compares logarithms of scores. The utility of this transformation is thus rather questionable. Quinlan (1996) computes arithmetic means of relative ratios; due to skewed distributions, these cannot be used in the t-test without further manipulation. A simpler way of compensating for different complexity of the problems is to divide the difference by the average score, $d_i = \frac{c_i^1 - c_i^2}{(c_i^1 + c_i^2)/2}$.

The second problem with the t-test is that unless the sample size is large enough ($\sim$ 30 data sets), the paired t-test requires that the differences between the two random variables compared are distributed normally. The nature of our problems does not give any provisions for normality and the number of data sets is usually much less than 30. Ironically, the Kolmogorov-Smirnov and similar tests for testing the normality of distributions have little power on small samples, that is, they are unlikely to detect abnormalities and warn against using the t-test. Therefore, for using the t-test we need normal distributions because we have small samples, but the small samples also prohibit us from checking the distribution shape.

	C4.5	C4.5+m	difference	rank
adult (sample)	0.763	0.768	+0.005	3.5
breast cancer	0.599	0.591	−0.008	7
breast cancer wisconsin	0.954	0.971	+0.017	9
cmc	0.628	0.661	+0.033	12
ionosphere	0.882	0.888	+0.006	5
iris	0.936	0.931	−0.005	3.5
liver disorders	0.661	0.668	+0.007	6
lung cancer	0.583	0.583	0.000	1.5
lymphography	0.775	0.838	+0.063	14
mushroom	1.000	1.000	0.000	1.5
primary tumor	0.940	0.962	+0.022	11
rheum	0.619	0.666	+0.047	13
voting	0.972	0.981	+0.009	8
wine	0.957	0.978	+0.021	10

Table 2: Comparison of AUC for C4.5 with m = 0 and C4.5 with m tuned for the optimal AUC. The columns on the right-hand illustrate the computation and would normally not be published in an actual paper.

The third problem is that the t-test is, just as averaging over data sets, affected by outliers which skew the test statistics and decrease the test's power by increasing the estimated standard error.

3.1.3 WILCOXON SIGNED-RANKS TEST

The Wilcoxon signed-ranks test (Wilcoxon, 1945) is a non-parametric alternative to the paired t-test, which ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences.

Let d_i again be the difference between the performance scores of the two classifiers on i-th out of N data sets. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the data sets on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i) \qquad R^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \text{rank}(d_i).$$

Let T be the smaller of the sums, $T = \min(R^+, R^-)$. Most books on general statistics include a table of exact critical values for T for N up to 25 (or sometimes more). For a larger number of data sets, the statistics

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}$$

is distributed approximately normally. With $\alpha = 0.05$, the null-hypothesis can be rejected if z is smaller than -1.96.

Let us illustrate the procedure on an example. Table 2 shows the comparison of AUC for C4.5 with m (the minimal number of examples in a leaf) set to zero and C4.5 with m tuned for the optimal AUC. For the latter, AUC has been computed with 5-fold internal cross validation on training examples for $m \in \{0,1,2,3,5,10,15,20,50\}$. The experiments were performed on 14 data sets from the UCI repository with binary class attribute. We used the original Quinlan's C4.5 code, equipped with an interface that integrates it into machine learning system Orange (Demšar and Zupan, 2004), which provided us with the cross validation procedures, classes for tuning arguments, and the scoring functions. We are trying to reject the null-hypothesis that both algorithms perform equally well.

There are two data sets on which the classifiers performed equally (lung-cancer and mushroom); if there was an odd number of them, we would ignore one. The ranks are assigned from the lowest to the highest absolute difference, and the equal differences (0.000, ± 0.005) are assigned average ranks.

The sum of ranks for the positive differences is $R^+ = 3.5+9+12+5+6+14+11+13+8+10+1.5 = 93$ and the sum of ranks for the negative differences equals $R^- = 7+3.5+1.5 = 12$. According to the table of exact critical values for the Wilcoxon's test, for a confidence level of $\alpha = 0.05$ and $N = 14$ data sets, the difference between the classifiers is significant if the smaller of the sums is equal or less than 21. We therefore reject the null-hypothesis.

The Wilcoxon signed ranks test is more sensible than the t-test. It assumes commensurability of differences, but only qualitatively: greater differences still count more, which is probably desired, but the absolute magnitudes are ignored. From the statistical point of view, the test is safer since it does not assume normal distributions. Also, the outliers (exceptionally good/bad performances on a few data sets) have less effect on the Wilcoxon than on the t-test.

The Wilcoxon test assumes continuous differences d_i, therefore they should not be rounded to, say, one or two decimals since this would decrease the power of the test due to a high number of ties.

When the assumptions of the paired t-test are met, the Wilcoxon signed-ranks test is less powerful than the paired t-test. On the other hand, when the assumptions are violated, the Wilcoxon test can be even more powerful than the t-test.

3.1.4 COUNTS OF WINS, LOSSES AND TIES: SIGN TEST

A popular way to compare the overall performances of classifiers is to count the number of data sets on which an algorithm is the overall winner. When multiple algorithms are compared, pairwise comparisons are sometimes organized in a matrix.

Some authors also use these counts in inferential statistics, with a form of binomial test that is known as the sign test (Sheskin, 2000; Salzberg, 1997). If the two algorithms compared are, as assumed under the null-hypothesis, equivalent, each should win on approximately $N/2$ out of N data sets. The number of wins is distributed according to the binomial distribution; the critical number of wins can be found in Table 3. For a greater number of data sets, the number of wins is under the null-hypothesis distributed according to $N(N/2, \sqrt{N}/2)$, which allows for the use of z-test: if the number of wins is at least $N/2 + 1.96\sqrt{N}/2$ (or, for a quick rule of a thumb, $N/2 + \sqrt{N}$), the algorithm is significantly better with $p < 0.05$. Since tied matches support the null-hypothesis we should not discount them but split them evenly between the two classifiers; if there is an odd number of them, we again ignore one.

#data sets	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$w_{0.05}$	5	6	7	7	8	9	9	10	10	11	12	12	13	13	14	15	15	16	17	18	18
$w_{0.10}$	5	6	6	7	7	8	9	9	10	10	11	12	12	13	13	14	14	15	16	16	17

Table 3: Critical values for the two-tailed sign test at $\alpha = 0.05$ and $\alpha = 0.10$. A classifier is significantly better than another if it performs better on at least w_α data sets.

In example from Table 2, C4.5+m was better on 11 out of 14 data sets (counting also one of the two data sets on which the two classifiers were tied). According to Table 3 this difference is significant with $p < 0.05$.

This test does not assume any commensurability of scores or differences nor does it assume normal distributions and is thus applicable to any data (as long as the observations, *i.e.* the data sets, are independent). On the other hand, it is much weaker than the Wilcoxon signed-ranks test. According to Table 3, the sign test will not reject the null-hypothesis unless one algorithm almost always outperforms the other.

Some authors prefer to count only the significant wins and losses, where the significance is determined using a statistical test on each data set, for instance Dietterich's 5×2cv. The reasoning behind this practice is that "some wins and losses are random and these should not count". This would be a valid argument if statistical tests could distinguish between the random and non-random differences. However, statistical tests only measure the improbability of the obtained experimental result if the null hypothesis was correct, which is not even the (im)probability of the null-hypothesis.

For the sake of argument, suppose that we compared two algorithms on one thousand different data sets. In each and every case, algorithm A was better than algorithm B, but the difference was never significant. It is true that for each single case the difference between the two algorithms can be attributed to a random chance, but how likely is it that one algorithm was just lucky in all 1000 out of 1000 independent experiments?

Contrary to the popular belief, counting only significant wins and losses therefore does not make the tests more but rather less reliable, since it draws an arbitrary threshold of $p < 0.05$ between what counts and what does not.

3.2 Comparisons of Multiple Classifiers

None of the above tests was designed for reasoning about the means of multiple random variables. Many authors of machine learning papers nevertheless use them for that purpose. A common example of such questionable procedure would be comparing seven algorithms by conducting all 21 paired t-tests and reporting results like "algorithm A was found significantly better than B and C, and algorithms A and E were significantly better than D, while there were no significant differences between other pairs". When so many tests are made, a certain proportion of the null hypotheses is rejected due to random chance, so listing them makes little sense.

The issue of multiple hypothesis testing is a well-known statistical problem. The usual goal is to control the *family-wise error*, the probability of making at least one Type 1 error in any of the comparisons. In machine learning literature, Salzberg (1997) mentions a general solution for the

problem of multiple testing, the Bonferroni correction, and notes that it is usually very conservative and weak since it supposes the independence of the hypotheses.

Statistics offers more powerful specialized procedures for testing the significance of differences between multiple means. In our situation, the most interesting two are the well-known ANOVA and its non-parametric counterpart, the Friedman test. The latter, and especially its corresponding Nemenyi post-hoc test are less known and the literature on them is less abundant; for this reason, we present them in more detail.

3.2.1 ANOVA

The common statistical method for testing the differences between more than two related sample means is the *repeated-measures ANOVA* (or *within-subjects ANOVA*) (Fisher, 1959). The "related samples" are again the performances of the classifiers measured across the same data sets, preferably using the same splits onto training and testing sets. The null-hypothesis being tested is that all classifiers perform the same and the observed differences are merely random.

ANOVA divides the total variability into the variability between the classifiers, variability between the data sets and the residual (error) variability. If the between-classifiers variability is significantly larger than the error variability, we can reject the null-hypothesis and conclude that there *are* some differences between the classifiers. In this case, we can proceed with a post-hoc test to find out which classifiers actually differ. Of many such tests for ANOVA, the two most suitable for our situation are the Tukey test (Tukey, 1949) for comparing all classifiers with each other and the Dunnett test (Dunnett, 1980) for comparisons of all classifiers with the control (for instance, comparing the base classifier and some proposed improvements, or comparing the newly proposed classifier with several existing methods). Both procedures compute the standard error of the difference between two classifiers by dividing the residual variance by the number of data sets. To make pairwise comparisons between the classifiers, the corresponding differences in performances are divided by the standard error and compared with the critical value. The two procedures are thus similar to a t-test, except that the critical values tabulated by Tukey and Dunnett are higher to ensure that there is at most 5 % chance that one of the pairwise differences will be erroneously found significant.

Unfortunately, ANOVA is based on assumptions which are most probably violated when analyzing the performance of machine learning algorithms. First, ANOVA assumes that the samples are drawn from normal distributions. In general, there is no guarantee for normality of classification accuracy distributions across a set of problems. Admittedly, even if distributions are abnormal this is a minor problem and many statisticians would not object to using ANOVA unless the distributions were, for instance, clearly bi-modal (Hamilton, 1990). The second and more important assumption of the repeated-measures ANOVA is sphericity (a property similar to the homogeneity of variance in the usual ANOVA, which requires that the random variables have equal variance). Due to the nature of the learning algorithms and data sets this cannot be taken for granted. Violations of these assumptions have an even greater effect on the post-hoc tests. ANOVA therefore does not seem to be a suitable omnibus test for the typical machine learning studies.

We will not describe ANOVA and its post-hoc tests in more details due to our reservations about the parametric tests and, especially, since these tests are well known and described in statistical literature (Zar, 1998; Sheskin, 2000).

		ANOVA		
		$p < 0.01$	$0.01 \leq p \leq 0.05$	$0.05 < p$
Friedman	$p < 0.01$	16	1	0
test	$0.01 \leq p \leq 0.05$	4	1	4
	$0.05 < p$	0	2	28

Table 4: Friedman's comparison of his test and the repeated-measures ANOVA on 56 independent problems (Friedman, 1940).

3.2.2 FRIEDMAN TEST

The Friedman test (Friedman, 1937, 1940) is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2..., as shown in Table 6. In case of ties (like in iris, lung cancer, mushroom and primary tumor), average ranks are assigned.

Let r_i^j be the rank of the j-th of k algorithms on the i-th of N data sets. The Friedman test compares the average ranks of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom, when N and k are big enough (as a rule of a thumb, $N > 10$ and $k > 5$). For a smaller number of algorithms and data sets, exact critical values have been computed (Zar, 1998; Sheskin, 2000).

Iman and Davenport (1980) showed that Friedman's χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. The table of critical values can be found in any statistical book.

As for the two-classifier comparisons, the (non-parametric) Friedman test has theoretically less power than (parametric) ANOVA when the ANOVA's assumptions are met, but this does not need to be the case when they are not. Friedman (1940) experimentally compared ANOVA and his test on 56 independent problems and showed that the two methods mostly agree (Table 4). When one method finds significance at $p < 0.01$, the other shows significance of at least $p < 0.05$. Only in 2 cases did ANOVA find significant what was insignificant for Friedman, while the opposite happened in 4 cases.

If the null-hypothesis is rejected, we can proceed with a post-hoc test. The Nemenyi test (Nemenyi, 1963) is similar to the Tukey test for ANOVA and is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$$

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

(a) Critical values for the two-tailed Nemenyi test

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.10}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

(b) Critical values for the two-tailed Bonferroni-Dunn test; the number of classifiers include the control classifier.

Table 5: Critical values for post-hoc tests after the Friedman test

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$ (Table 5(a)).

When all classifiers are compared with a control classifier, we can instead of the Nemenyi test use one of the general procedures for controlling the family-wise error in multiple hypothesis testing, such as the Bonferroni correction or similar procedures. Although these methods are generally conservative and can have little power, they are in this specific case more powerful than the Nemenyi test, since the latter adjusts the critical value for making $k(k-1)/2$ comparisons while when comparing with a control we only make $k-1$ comparisons.

The test statistics for comparing the i-th and j-th classifier using these methods is

$$z = (R_i - R_j) \Big/ \sqrt{\frac{k(k+1)}{6N}} \,.$$

The z value is used to find the corresponding probability from the table of normal distribution, which is then compared with an appropriate α. The tests differ in the way they adjust the value of α to compensate for multiple comparisons.

The Bonferroni-Dunn test (Dunn, 1961) controls the family-wise error rate by dividing α by the number of comparisons made ($k-1$, in our case). The alternative way to compute the same test is to calculate the CD using the same equation as for the Nemenyi test, but using the critical values for $\alpha/(k-1)$ (for convenience, they are given in Table 5(b)). The comparison between the tables for Nemenyi's and Dunn's test shows that the power of the post-hoc test is much greater when all classifiers are compared only to a control classifier and not between themselves. We thus should not make pairwise comparisons when we in fact only test whether a newly proposed method is better than the existing ones.

For a contrast from the single-step Bonferroni-Dunn procedure, step-up and step-down procedures sequentially test the hypotheses ordered by their significance. We will denote the ordered p values by p_1, p_2, ..., so that $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. The simplest such methods are due to Holm (1979) and Hochberg (1988). They both compare each p_i with $\alpha/(k-i)$, but differ in the order

of the tests.[1] Holm's step-down procedure starts with the most significant p value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and we are allowed to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, the test proceeds with the third, and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well. Hochberg's step-up procedure works in the opposite direction, comparing the largest p value with α, the next largest with $\alpha/2$ and so forth until it encounters a hypothesis it *can* reject. All hypotheses with smaller p values are then rejected as well.

Hommel's procedure (Hommel, 1988) is more complicated to compute and understand. First, we need to find the largest j for which $p_{n-j+k} > k\alpha/j$ for all $k = 1..j$. If no such j exists, we can reject all hypotheses, otherwise we reject all for which $p_i \leq \alpha/j$.

Holm's procedure is more powerful than the Bonferroni-Dunn's and makes no additional assumptions about the hypotheses tested. The only advantage of the Bonferroni-Dunn test seems to be that it is easier to describe and visualize because it uses the same CD for all comparisons. In turn, Hochberg's and Hommel's methods reject more hypotheses than Holm's, yet they may under some circumstances exceed the prescribed family-wise error since they are based on the Simes conjecture which is still being investigated. It has been reported (Holland, 1991) that the differences between the enhanced methods are in practice rather small, therefore the more complex Hommel method offers no great advantage over the simple Holm method.

Although we here use these procedures only as post-hoc tests for the Friedman test, they can be used generally for controlling the family-wise error when multiple hypotheses of possibly various types are tested. There exist other similar methods, as well as some methods that instead of controlling the family-wise error control the number of falsely rejected null-hypotheses (false discovery rate, FDR). The latter are less suitable for the evaluation of machine learning algorithms since they require the researcher to decide for the acceptable false discovery rate. A more complete formal description and discussion of all these procedures was written, for instance, by Shaffer (1995).

Sometimes the Friedman test reports a significant difference but the post-hoc test fails to detect it. This is due to the lower power of the latter. No other conclusions than that some algorithms do differ can be drawn in this case. In our experiments this has, however, occurred only in a few cases out of one thousand.

The procedure is illustrated by the data from Table 6, which compares four algorithms: C4.5 with `m` fixed to 0 and `cf` (confidence interval) to 0.25, C4.5 with `m` fitted in 5-fold internal cross validation, C4.5 with `cf` fitted the same way and, finally, C4.5 in which we fitted both arguments, trying all combinations of their values. Parameter `m` was set to 0, 1, 2, 3, 5, 10, 15, 20, 50 and `cf` to 0, 0.1, 0.25 and 0.5.

Average ranks by themselves provide a fair comparison of the algorithms. On average, C4.5+m and C4.5+m+cf ranked the second (with ranks 2.000 and 1.964, respectively), and C4.5 and C4.5+cf the third (3.143 and 2.893). The Friedman test checks whether the measured average ranks are significantly different from the mean rank $R_j = 2.5$ expected under the null-hypothesis:

$$\chi_F^2 = \frac{12 \cdot 14}{4 \cdot 5}\left[(3.143^2 + 2.000^2 + 2.893^2 + 1.964^2) - \frac{4 \cdot 5^2}{4}\right] = 9.28$$

$$F_F = \frac{13 \cdot 9.28}{14 \cdot 3 - 9.28} = 3.69.$$

1. In the usual definitions of these procedures k would denote the number of hypotheses, while in our case the number of hypotheses is $k-1$, hence the differences in the formulae.

	C4.5	C4.5+m	C4.5+cf	C4.5+m+cf
adult (sample)	0.763 (4)	0.768 (3)	0.771 (2)	0.798 (1)
breast cancer	0.599 (1)	0.591 (2)	0.590 (3)	0.569 (4)
breast cancer wisconsin	0.954 (4)	0.971 (1)	0.968 (2)	0.967 (3)
cmc	0.628 (4)	0.661 (1)	0.654 (3)	0.657 (2)
ionosphere	0.882 (4)	0.888 (2)	0.886 (3)	0.898 (1)
iris	0.936 (1)	0.931 (2.5)	0.916 (4)	0.931 (2.5)
liver disorders	0.661 (3)	0.668 (2)	0.609 (4)	0.685 (1)
lung cancer	0.583 (2.5)	0.583 (2.5)	0.563 (4)	0.625 (1)
lymphography	0.775 (4)	0.838 (3)	0.866 (2)	0.875 (1)
mushroom	1.000 (2.5)	1.000 (2.5)	1.000 (2.5)	1.000 (2.5)
primary tumor	0.940 (4)	0.962 (2.5)	0.965 (1)	0.962 (2.5)
rheum	0.619 (3)	0.666 (2)	0.614 (4)	0.669 (1)
voting	0.972 (4)	0.981 (1)	0.975 (2)	0.975 (3)
wine	0.957 (3)	0.978 (1)	0.946 (4)	0.970 (2)
average rank	3.143	2.000	2.893	1.964

Table 6: Comparison of AUC between C4.5 with `m` = 0 and C4.5 with parameters `m` and/or `cf` tuned for the optimal AUC. The ranks in the parentheses are used in computation of the Friedman test and would usually not be published in an actual paper.

With four algorithms and 14 data sets, F_F is distributed according to the F distribution with $4-1=3$ and $(4-1)\times(14-1)=39$ degrees of freedom. The critical value of $F(3,39)$ for $\alpha=0.05$ is 2.85, so we reject the null-hypothesis.

Further analysis depends upon what we intended to study. If no classifier is singled out, we use the Nemenyi test for pairwise comparisons. The critical value (Table 5(a)) is 2.569 and the corresponding CD is $2.569\sqrt{\frac{4\cdot5}{6\cdot14}}=1.25$. Since even the difference between the best and the worst performing algorithm is already smaller than that, we can conclude that the post-hoc test is not powerful enough to detect any significant differences between the algorithms.

At p=0.10, CD is $2.291\sqrt{\frac{4\cdot5}{6\cdot14}}=1.12$. We can identify two groups of algorithms: the performance of pure C4.5 is significantly worse than that of C4.5+m and C4.5+m+cf. We cannot tell which group C4.5+cf belongs to. Concluding that it belongs to both would be a statistical nonsense since a subject cannot come from two different populations. The correct statistical statement would be that *the experimental data is not sufficient to reach any conclusion regarding C4.5+cf*.

The other possible hypothesis made before collecting the data could be that it is possible to improve on C4.5's performance by tuning its parameters. The easiest way to verify this is to compute the CD with the Bonferroni-Dunn test. In Table 5(b) we find that the critical value $q_{0.05}$ for 4 classifiers is 2.394, so CD is $2.394\sqrt{\frac{4\cdot5}{6\cdot14}}=1.16$. C4.5+m+cf performs significantly better than C4.5 ($3.143-1.964=1.179>1.16$) and C4.5+cf does not ($3.143-2.893=0.250<1.16$), while C4.5+m is just below the critical difference, but close to it ($3.143-2.000=1.143\approx1.16$). We can conclude that the experiments showed that fitting `m` seems to help, while we did not detect any significant improvement by fitting `cf`.

For the other tests we have to compute and order the corresponding statistics and p values. The standard error is $SE=\sqrt{\frac{4\cdot5}{6\cdot14}}=0.488$.

i	classifier	$z = (R_0 - R_i)/SE$	p	α/i
1	C4.5+m+cf	$(3.143-1.964)/0.488 = 2.416$	0.016	0.017
2	C4.5+m	$(3.143-2.000)/0.488 = 2.342$	0.019	0.025
3	C4.5+cf	$(3.143-2.893)/0.488 = 0.512$	0.607	0.050

The Holm procedure rejects the first and then the second hypothesis since the corresponding p values are smaller than the adjusted α's. The third hypothesis cannot be rejected; if there were any more, we would have to retain them, too.

The Hochberg procedure starts from the bottom. Unable to reject the last hypothesis, it check the second last, rejects it and among with it all the hypotheses with smaller p values (the top-most one).

Finally, the Hommel procedure finds that $j = 3$ does not satisfy the condition at $k = 2$. The maximal value of j is 2, and the first two hypotheses can be rejected since their p values are below $\alpha/2$.

All step-down and step-up procedure found C4.5+cf+m and C4.5+m significantly different from C4.5, while the Bonferroni-Dunn test found C4.5 and C4.5+m too similar.

3.2.3 CONSIDERING MULTIPLE REPETITIONS OF EXPERIMENTS

In our examples we have used AUCs measured and averaged over repetitions of training/testing episodes. For instance, each cell in Table 6 represents an average over five-fold cross validation. Could we also consider the variance, or even the results of individual folds?

There are variations of the ANOVA and the Friedman test which can consider *multiple observations per cell* provided that the observations are independent (Zar, 1998). This is not the case here, since training data in multiple random samples overlaps. We are not aware of any statistical test that could take this into account.

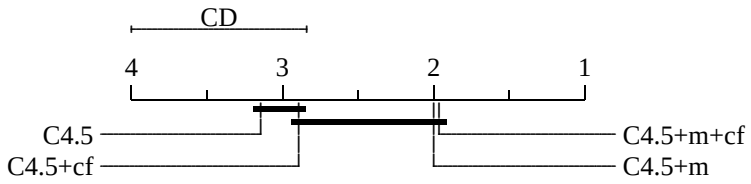
3.2.4 GRAPHICAL PRESENTATION OF RESULTS

When multiple classifiers are compared, the results of the post-hoc tests can be visually represented with a simple diagram. Figure 1 shows the results of the analysis of the data from Table 6. The top line in the diagram is the axis on which we plot the average ranks of methods. The axis is turned so that the lowest (best) ranks are to the right since we perceive the methods on the right side as better.

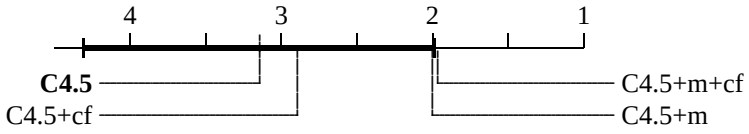
When comparing all the algorithms against each other, we connect the groups of algorithms that are not significantly different (Figure 1(a)). We also show the critical difference above the graph.

If the methods are compared to the control using the Bonferroni-Dunn test we can mark the interval of one CD to the left and right of the average rank of the control algorithm (Figure 1(b)). Any algorithm with the rank outside this area is significantly different from the control. Similar graphs for the other post-hoc tests would need to plot a different adjusted critical interval for each classifier and specify the procedure used for testing and the corresponding order of comparisons, which could easily become confusing.

For another example, Figure 2 graphically represents the comparison of feature scoring measures for the problem of keyword prediction on five domains formed from the Yahoo hierarchy studied by Mladenić and Grobelnik (1999). The analysis reveals that Information gain performs significantly worse than Weight of evidence, Cross entropy Txt and Odds ratio, which seem to have



(a) Comparison of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $p = 0.10$) are connected.



(b) Comparison of one classifier against the others with the Bonferroni-Dunn test. All classifiers with ranks outside the marked interval are significantly different ($p < 0.05$) from the control.

Figure 1: Visualization of post-hoc tests for data from Table 6.

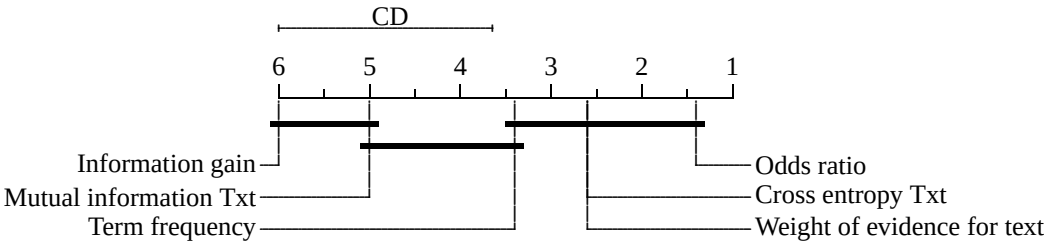


Figure 2: Comparison of recalls for various feature selection measures; analysis of the results from the paper by Mladenić and Grobelnik (1999).

equivalent performances. The data is not sufficient to conclude whether Mutual information Txt performs the same as Information gain or Term Frequency, and similarly, whether Term Frequency is equivalent to Mutual information Txt or to the better three methods.

4. Empirical Comparison of Tests

We experimentally observed two properties of the described tests: their replicability and the likelihood of rejecting the null-hypothesis. Performing the experiments to answer questions like "which statistical test is most likely to give the correct result" or "which test has the lowest Type 1/Type 2 error rate" would be a pointless exercise since the proposed inferential tests suppose different kinds

of commensurability and thus compare the classifiers from different aspects. The "correct answer", rejection or non-rejection of the null-hypothesis, is thus not well determined and is, in a sense, related to the choice of the test.

4.1 Experimental Setup

We examined the behaviour of the studied tests through the experiments in which we repeatedly compared the learning algorithms on sets of ten randomly drawn data sets and recorded the p values returned by the tests.

4.1.1 DATA SETS AND LEARNING ALGORITHMS

We based our experiments on several common learning algorithms and their variations: C4.5, C4.5 with m and C4.5 with cf fitted for optimal accuracy, another tree learning algorithm implemented in Orange (with features similar to the original C4.5), naive Bayesian learner that models continuous probabilities using LOESS (Cleveland, 1979), naive Bayesian learner with continuous attributes discretized using Fayyad-Irani's discretization (Fayyad and Irani, 1993) and kNN (k=10, neighbour weights adjusted with the Gaussian kernel).

We have compiled a sample of forty real-world data sets,[2] from the UCI machine learning repository (Blake and Merz, 1998); we have used the data sets with discrete classes and avoided artificial data sets like Monk problems. Since no classifier is optimal for all possible data sets, we have simulated experiments in which a researcher wants to show particular advantages of a particular algorithm and thus selects a corresponding compendium of data sets. We did this by measuring the classification accuracies of the classifiers on all data sets in advance by using ten-fold cross validation. When comparing two classifiers, samples of ten data sets were randomly selected so that the probability for the data set i being chosen was proportional to $1/(1+e^{-kd_i})$, where d_i is the (positive or negative) difference in the classification accuracies on that data set and k is the bias through which we can regulate the differences between the classifiers.[3] Whereas at $k = 0$ the data set selection is random with the uniform distribution, with higher values of k we are more likely to select the sets that favour a particular learning method. Note that choosing the data sets with knowing their success (as estimated in advance) is only a simulation, while the researcher would select the data sets according to other criteria. Using the described procedure in practical evaluations of algorithms would be considered cheating.

We decided to avoid "artificial" classifiers and data sets constructed specifically for testing the statistical tests, such as those used, for instance, by Dietterich (1998). In such experimental procedures some assumptions need to be made about the real-world data sets and the learning algorithms, and the artificial data and algorithms are constructed in a way that mimics the supposed real-world situation in a controllable manner. In our case, we would construct two or more classifiers with a prescribed probability of failure over a set of (possible imaginary) data sets so that we could,

2. The data sets used are: adult, balance-scale, bands, breast cancer (haberman), breast cancer (lju), breast cancer (wisc), car evaluation, contraceptive method choice, credit screening, dermatology, ecoli, glass identification, hayes-roth, hepatitis, housing, imports-85, ionosphere, iris, liver disorders, lung cancer, lymphography, mushrooms, pima indians diabetes, post-operative, primary tumor, promoters, rheumatism, servo, shuttle landing, soybean, spambase, spect, spectf, teaching assistant evaluation, tic tac toe, titanic, voting, waveform, wine recognition, yeast.
3. The function used is the logistic function. It was chosen for its convenient shape; we do not claim that such relation actually occurs in practice when selecting the data sets for experiments.

knowing the correct hypothesis, observe the Type 1 and 2 error rates of the proposed statistical tests.

Unfortunately, we do not know what should be our assumptions about the real world. To what extent are the classification accuracies (or other measures of success) incommensurable? How (ab)normal is their distribution? How homogenous is the variance? Moreover, if we do make certain assumptions, the statistical theory is already able to tell the results of the experiments that we are setting up. Since the statistical tests which we use are theoretically well understood, we do not need to test the tests but the compliance of the real-world data to their assumptions. In other words, we know, from the theory, that the t-test on a small sample (that is, on a small number of data sets) requires the normal distribution, so by constructing an artificial environment that will yield non-normal distributions we can make the t-test fail. The real question however is whether the *real world* distributions are normal enough for the t-test to work.

Cannot we test the assumptions directly? As already mentioned in the description of the t-test, the tests like the Kolmogorov-Smirnov test of normality are unreliable on small samples where they are very unlikely to detect abnormalities. And even if we did have suitable tests at our disposal, they would only compute the degree of (ab)normality of the distribution, non-homogeneity of variance *etc*, and not the sample's suitability for t-test.

Our decision to use real-world learning algorithms and data sets in unmodified form prevents us from artificially setting the differences between them by making them intentionally misclassify a certain proportion of examples. This is however compensated by our method of selecting the data sets: we can regulate the differences between the learning algorithms by affecting the data set selection through regulating the bias k. In this way, we perform the experiments on real-world data sets and algorithms, and yet observe the performance of the statistics at various degrees of differences between the classifiers.

4.1.2 MEASURES OF POWER AND REPLICABILITY

Formally, the power of a statistical test is defined as the probability that the test will (correctly) reject the false null-hypothesis. Since our criterion of what is actually false is related to the selection of the test (which should be based on the kind of differences between the classifiers we want to measure), we can only observe the probability of the rejection of the null-hypothesis, which is nevertheless related to the power.

We do this in two ways. First, we set the significance level at 5% and observe in how many experiments out of one thousand does a particular test reject the null-hypothesis. The shortcoming of this is that it observes only the behaviour of statistics at around $p = 0.05$ (which is probably what we are interested in), yet it can miss a bigger picture. We therefore also observed the average p values as another measure of "power" of the test: the lower the values, the more likely it is for a test to reject the null-hypothesis at a set confidence level.

The two measures for assessing the power of the tests lead to two related measures of replicability. Bouckaert (2004) proposed a definition which can be used in conjuction with counting the rejections of the null-hypothesis. He defined the replicability as the probability that two experiments with the same pair of algorithms will produce the same results, that is, that both experiments accept or reject the null-hypothesis, and devised the optimal unbiased estimator of this probability,

$$R(e) = \sum_{1 \le i < j \le n} \frac{I(e_i = e_j)}{n(n-1)/2}$$

where e_i is the outcome of the i-th experiment out of n (e_i is 1 if the null-hypothesis is accepted, 0 if it is not) and I is the indicator function which is 1 if its argument is true and 0 otherwise. Bouckaert also describes a simpler way to compute $R(e)$: if the hypothesis was accepted in p and rejected in q experiments out of n, $R(e)$ equals $(p(p-1)+q(q-1))/n(n-1)$. The minimal value of R, 0.5, occurs when $p=q=n/2$, and the maximal, 1.0, when either p or q is zero.

The disadvantage of this measure is that a statistical test will show a low replicability when the difference between the classifiers is marginally significant. When comparing two tests of different power, the one with results closer to the chosen α will usually be deemed as less reliable.

When the power is estimated by the average of p values, the replicability is naturally defined through their variance. The variance of p is between 0 and 0.25; the latter occurs when one half of p's equals zero and the other half equals one.[4] To allow for comparisons with Bouckaert's $R(e)$, we define the replicability with respect to the variance of p as

$$R(p) = 1 - 2 \cdot \mathrm{var}(p) = 1 - 2\frac{\sum_i (p_i - \overline{p})^2}{n-1}.$$

A problem with this measure of replicability when used in our experimental procedure is that when the bias k increases, the variability of the data set selection decreases and so does the variance of p. The size of the effect depends on the number of data sets. Judged by the results of the experiments, our collection of forty data sets is large enough to keep the variability practically unaffected for the used values of k (see the left graph in Figure 4.c; if the variability of selections decreased, the variance of p could not remain constant).

The described definitions of replicability are related. Since $I(e_i = e_j)$ equals $1-(e_i-e_j)^2$, we can reformulate $R(e)$ as

$$R(e) = \sum_{1 \le i < j \le n} \frac{1-(e_i-e_j)^2}{n(n-1)/2} = 1 - \sum_i \sum_j \frac{(e_i-e_j)^2}{n(n-1)} = 1 - \sum_i \sum_j \frac{((e_i-\overline{e})-(e_j-\overline{e}))^2}{n(n-1)}.$$

From here, it is easy to verify that

$$R(e) = 1 - 2\frac{\sum_i (e_i - \overline{e})^2}{n-1}.$$

The fact that Bouckaert's formula is the optimal unbiased estimator for $R(e)$ is related to $\sum_i (e_i - \overline{e})^2/(n-1)$ being the optimal unbiased estimator of the population variance.

4.2 Comparisons of Two Classifiers

We have tested four statistics for comparisons of two classifiers: the paired t-test on absolute and on relative differences, the Wilcoxon test and the sign test. The experiments were run on 1000 random selections of ten data sets, as described above.

The graphs on the left hand side of Figure 3 show the average p values returned by the tests as a function of the bias k when comparing C4.5-cf, naive Bayesian classifier and kNN (note that the scale is turned upside down so the curve rises when the power of the test increases). The graphs on the right hand side show the number of experiments in which the hypothesis was rejected at

4. Since we estimate the population variance from the sample variance, the estimated variance will be higher by $0.25/(n-1)$. With any decent number of experiments, the difference is however negligible.

$\alpha = 5\%$. To demonstrate the relation between power (as we measure it) and Bouckaert's measure of replicability we have added the right axis that shows $R(e)$ corresponding to the number of rejected hypothesis.

Note that at $k = 0$ the number of experiments in which the null hypothesis is rejected is not 50%. Lower settings of k do not imply that both algorithms compared should perform approximately equally, but only that we do not (artificially) bias the data sets selection to favour one of them. Therefore, at $k = 0$ the tests reflect the number of rejections of the null-hypothesis on a completely random selection of data sets from our collection.

Both variations of the t-test give similar results, with the test on relative differences being slightly, yet consistently weaker. The Wilcoxon signed-ranks test gives much lower p values and is more likely to reject the null-hypothesis than t-tests in almost all cases. The sign test is, as known from the theory, much weaker than the other tests.

The two measures of replicability give quite different results. Judged by $R(p)$ (graphs on the left hand side of Figure 4), the Wilcoxon test exhibits the smallest variation of p values. For a contrast, Bouckaert's $R(e)$ (right hand side of Figure 4) shows the Wilcoxon test as the least reliable. However, the shape of the curves on these graphs and the right axes in Figure 3 clearly show that the test is less reliable (according to $R(e)$) when the p values are closer to 0.05, so the Wilcoxon test seems unreliable due to its higher power keeping it closer to p=0.05 than the other tests.

Table 7 shows comparisons of all seven classifiers with k set to 15. The numbers below the diagonal show the average p values and the related replicability $R(p)$, and the numbers above the diagonal represent the number of experiments in which the null-hypothesis was rejected at $\alpha = 5\%$ and the related $R(e)$. The table again shows that the Wilcoxon test almost always returns lower p values than other tests and more often rejects the null hypothesis. Measured by $R(p)$, the Wilcoxon test also has the highest replicability. $R(e)$, on the other hand, again prefers other tests with p values farther from the critical 0.05.

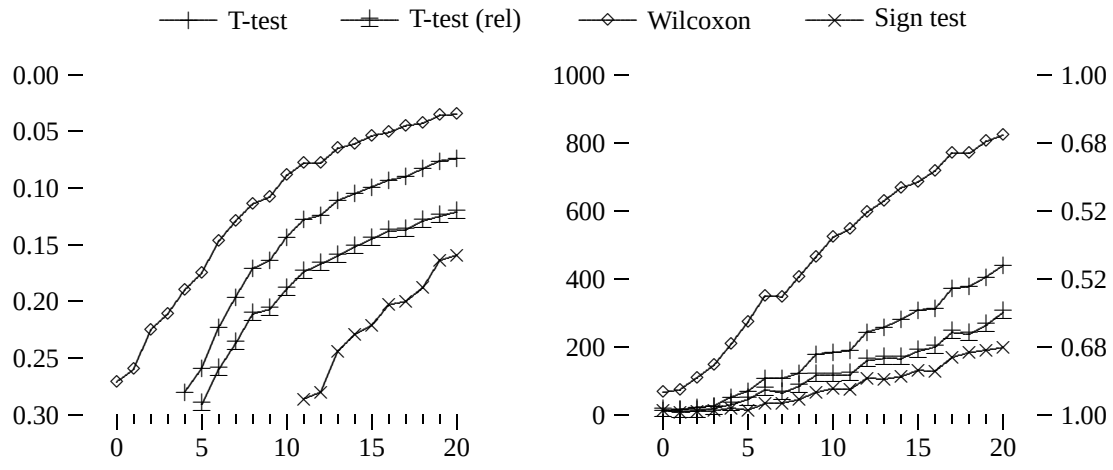
Overall, it is known that parametric tests are more likely to reject the null-hypothesis than the non-parametric unless their assumptions are violated. Our results suggest that the latter is indeed happening in machine learning studies that compare algorithms across collections of data sets. We therefore recommend using the Wilcoxon test, unless the t-test assumptions are met, either because we have many data sets or because we have reasons to believe that the measure of performance across data sets is distributed normally. The sign test, as the third alternative, is too weak to be generally useful.

Low values of $R(e)$ suggest that we should ensure the reliability of the results (especially when the differences between classifiers are marginally significant) by running the experiments on as many appropriate data sets as possible.

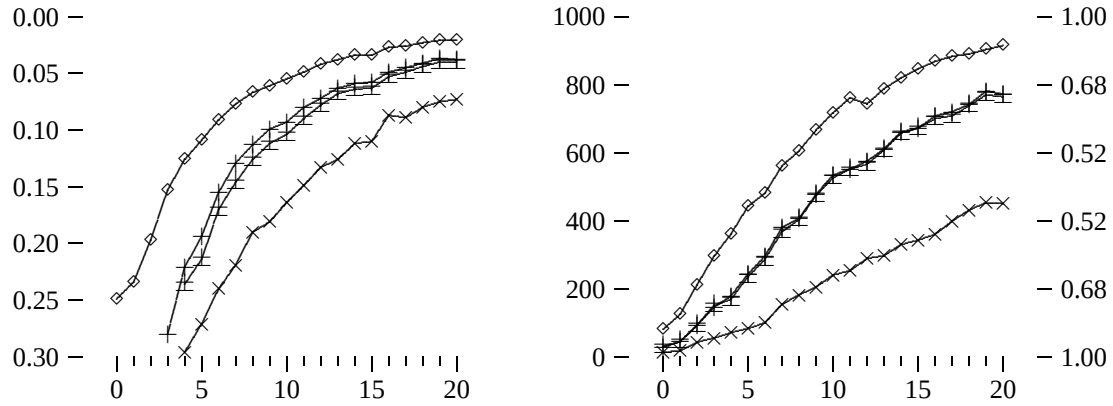
4.3 Comparisons of Multiple Classifiers

For comparison of multiple classifiers, samples of data sets were selected with the probabilities computed from the differences in the classification accuracy of C4.5 and naive Bayesian classifier with Fayyad-Irani discretization. These two classifiers were chosen for no particular reason; we have verified that the choice has no practical effect on the results.

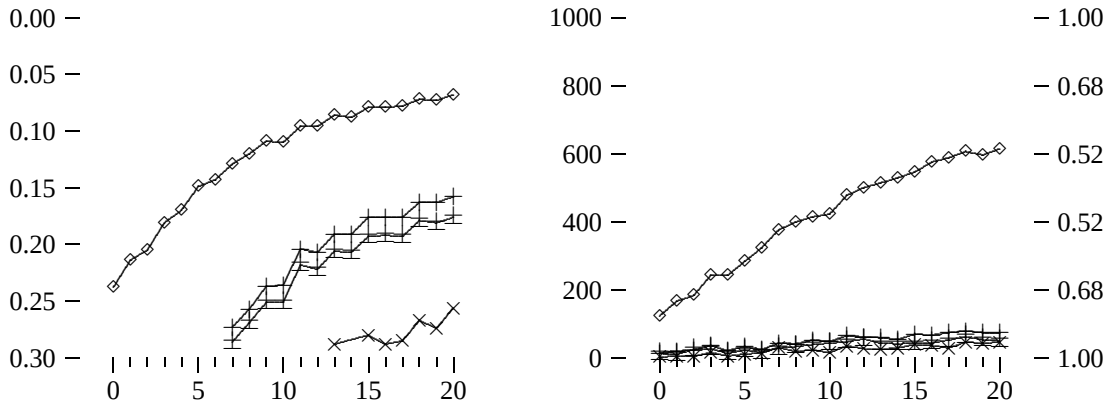
Results are shown in Figure 5. When the algorithms are more similar (at smaller values of k), the non-parametric Friedman test again appears stronger than the parametric, ANOVA. At greater



(a) C45-cf vs. naive Bayes

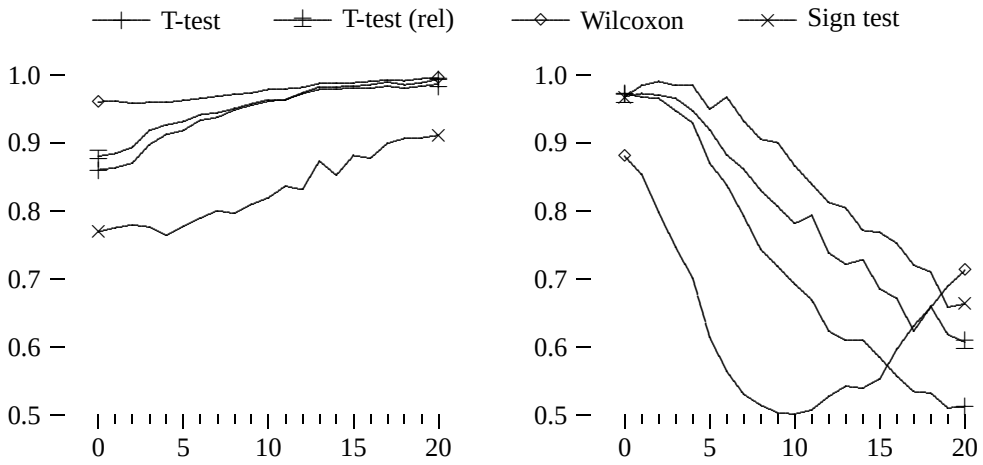


(b) C45-cf vs. kNN

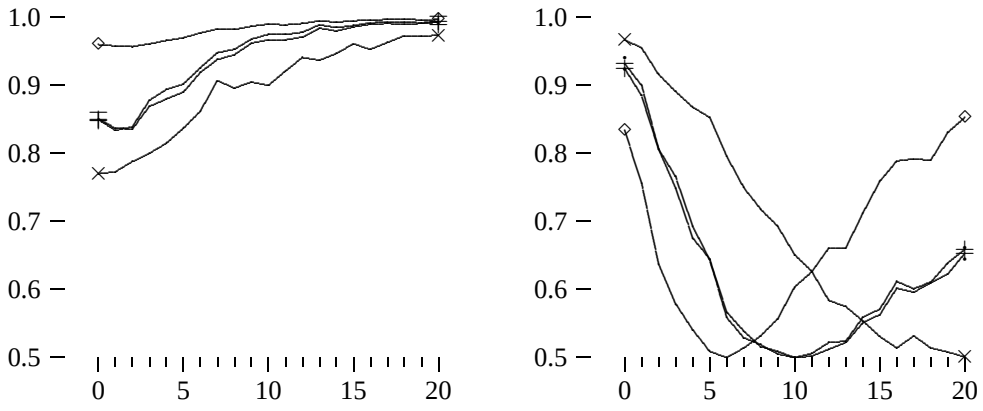


(c) Naive Bayes vs. kNN

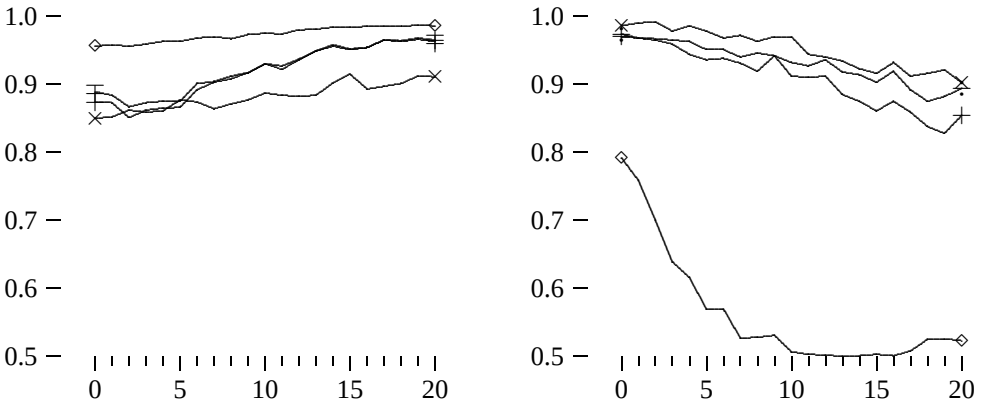
Figure 3: Power of statistical tests for comparison of two classifiers. Left: p values as a function of bias (k). Right: number of times the hypothesis was rejected (left axis) and the Bouckaert's R (right axis).



(a) C45-cf vs. naive Bayes



(b) C45-cf vs. kNN



(c) Naive Bayes vs. kNN

Figure 4: Replicability of tests for comparison of two classifiers: variance-based $R(p)$ (left) and Bouckaert's $R(e)$ (right).

	c45	c45-m	c45-cf	tree	bayes	disc-bayes	knn
c45		154/.74	709/.59	818/.70	178/.71	0/1.00	151/.74
c45-m	.16/.96		307/.57	909/.83	300/.58	0/1.00	376/.53
c45-cf	.05/.99	.10/.98		758/.63	335/.55	0/1.00	167/.72
tree	.04/.98	.02/1.00	.05/.98		679/.56	162/.73	592/.52
bayes	.15/.96	.12/.97	.11/.97	.05/.99		0/1.00	2/1.00
disc-bayes	.41/.92	.20/.95	.28/.92	.18/.94	.20/.97		981/.96
knn	.16/.96	.10/.98	.14/.97	.06/.99	.35/.94	.01/1.00	

(a) Paired t-test

	c45	c45-m	c45-cf	tree	bayes	disc-bayes	knn
c45		75/.86	592/.52	809/.69	181/.70	0/1.00	184/.70
c45-m	.17/.96		238/.64	848/.74	314/.57	0/1.00	438/.51
c45-cf	.06/.99	.11/.98		729/.60	361/.54	0/1.00	216/.66
tree	.04/.99	.03/1.00	.06/.98		662/.55	79/.85	584/.51
bayes	.16/.95	.12/.97	.11/.97	.05/.99		0/1.00	1/1.00
disc-bayes	.36/.94	.20/.96	.27/.94	.19/.95	.24/.98		970/.94
knn	.14/.96	.09/.98	.13/.97	.06/.99	.35/.95	.01/1.00	

(b) Paired t-test on relative differences

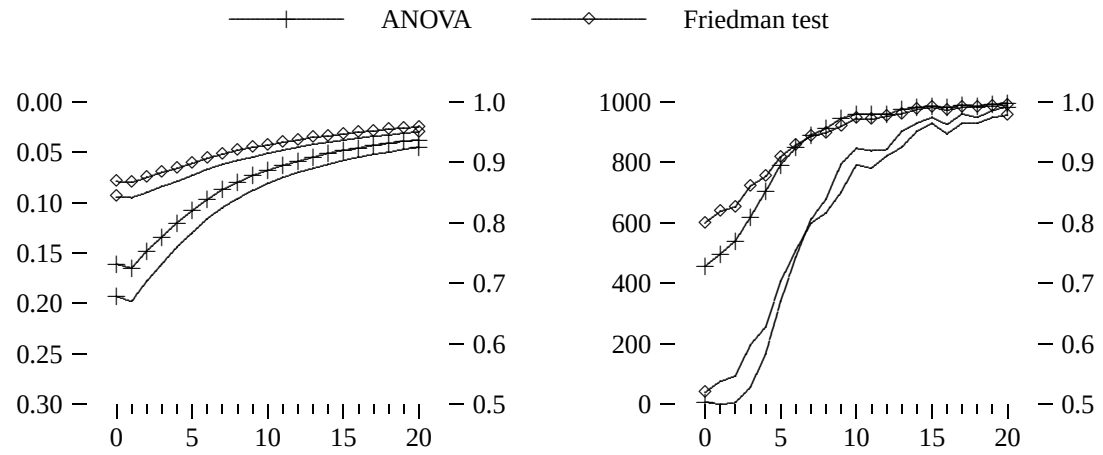
	c45	c45-m	c45-cf	tree	bayes	disc-bayes	knn
c45		521/.50	884/.79	897/.82	662/.55	81/.85	618/.53
c45-m	.08/.98		774/.65	983/.97	710/.59	351/.54	750/.62
c45-cf	.03/1.00	.04/.99		854/.75	804/.68	172/.71	720/.60
tree	.02/1.00	.01/1.00	.03/1.00		915/.84	521/.50	920/.85
bayes	.06/.99	.05/.99	.04/.99	.02/1.00		94/.83	102/.82
disc-bayes	.22/.96	.11/.98	.16/.97	.08/.98	.18/.97		999/1.00
knn	.07/.98	.04/.99	.05/.99	.02/1.00	.22/.96	.00/1.00	

(c) Wilcoxon signed-ranks test

	c45	c45-m	c45-cf	tree	bayes	disc-bayes	knn
c45		157/.74	323/.56	653/.55	171/.72	48/.91	110/.80
c45-m	.21/.90		205/.67	863/.76	299/.58	156/.74	256/.62
c45-cf	.10/.98	.16/.93		513/.50	423/.51	95/.83	229/.65
tree	.05/.99	.02/1.00	.09/.97		460/.50	210/.67	486/.50
bayes	.19/.89	.13/.94	.08/.97	.08/.97		0/1.00	1/1.00
disc-bayes	.29/.89	.18/.93	.25/.89	.18/.93	.52/.78		850/.74
knn	.25/.85	.14/.93	.15/.93	.07/.97	.45/.86	.01/1.00	

(d) Sign test

Table 7: Tests for comparisons of two classifiers: average p-values and $R(p)$ (below diagonal), and the number of null-hypothesis rejections and $R(e)$ (above diagonal).



(a) Average p values (left axis) and $R(p)$ (no symbols on lines, right axis)

(b) Number of experiments in which the null-hypothesis was rejected (left axis) and the corresponding $R(e)$ (no symbols on lines, right axis)

Figure 5: Comparison of ANOVA and Friedman test

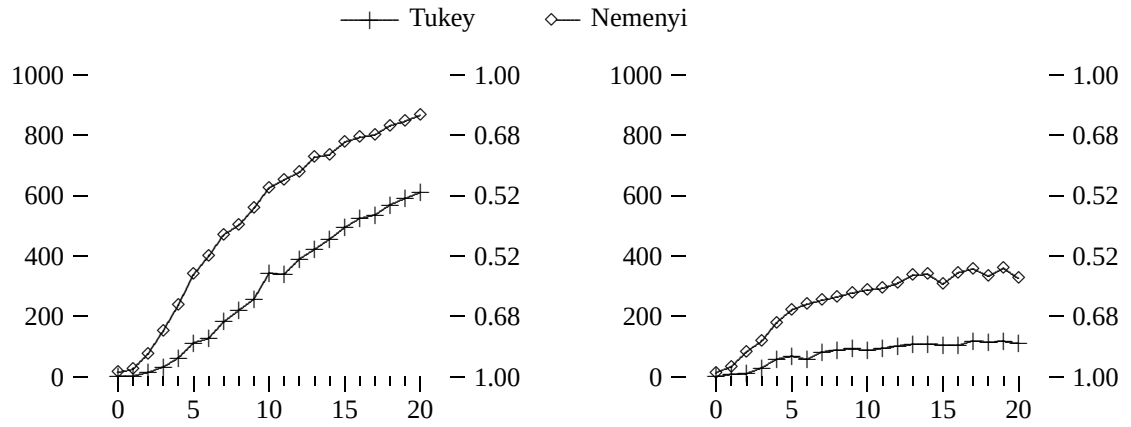
differences between the algorithms (k at around 10, in our experimental setup), ANOVA catches up and the two tests give similar results.

Replicability of the Friedman test is higher than that of ANOVA when measured by $R(p)$ and, due to the similar power of the tests, comparable when measured by $R(e)$. Altogether, replicability seems somewhat smaller than the replicability of the tests for comparisons of two classifiers. Therefore, as common sense would suggest, when comparing multiple classifiers, it is even more important to conduct the tests on as many data sets as possible.

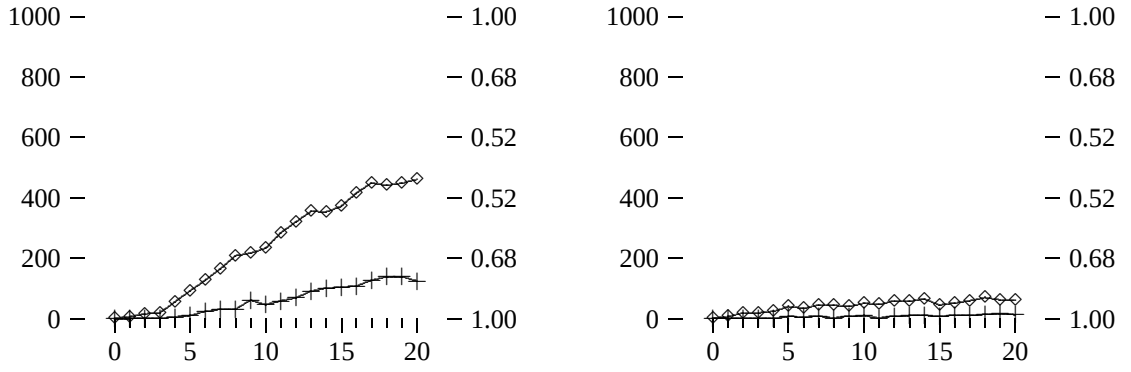
Figure 6 shows the comparison between the parametric Tukey and the non-parametric Nemenyi test. We counted the number of times they rejected the equivalence of C4.5-cf and naive Bayesian classifier and the equivalence of C4.5-cf and kNN (the comparison between the naive Bayesian classifier and kNN, which was included in previous tests, was less interesting since the null hypothesis was very seldom rejected). The two graphs on the left represent experiments in which the selection was based on the differences between the two algorithms compared on the graph, while for the right two graphs we used differences between the C4.5-cf and the average of the other six classifiers tested. In all cases, we have compared all seven algorithms, but presented only the number of rejections for the pair on the graph. The non-parametric test again more often rejects the null-hypothesis than the parametric one.

We do not show the p values and the corresponding replicabilities since they cannot always be computed or compared in all procedures due to different orders of testing.

Figure 7 compares post hoc tests for comparisons with a control classifier, using the same two ways of selecting data sets as in Figure 6. When the differences are large, the power of all tests is comparable, while when they are smaller the number of rejections for the parametric test seems to lag behind (we have observed this same pattern on other combinations of algorithms). The order of the non-parametric tests is as expected from the theory, although it is interesting to note that the Holm and Hochberg tests give practically equal results.



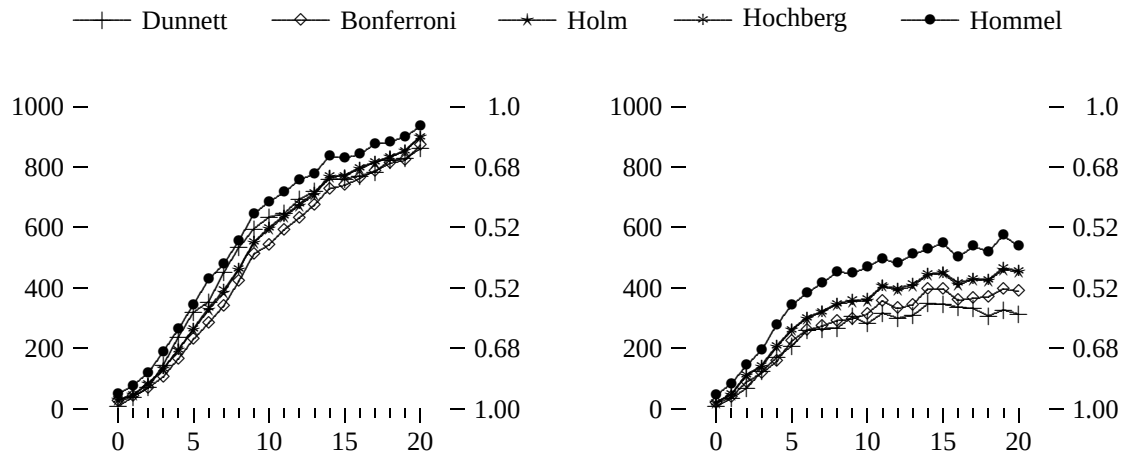
(a) C45-cf vs. naive Bayes



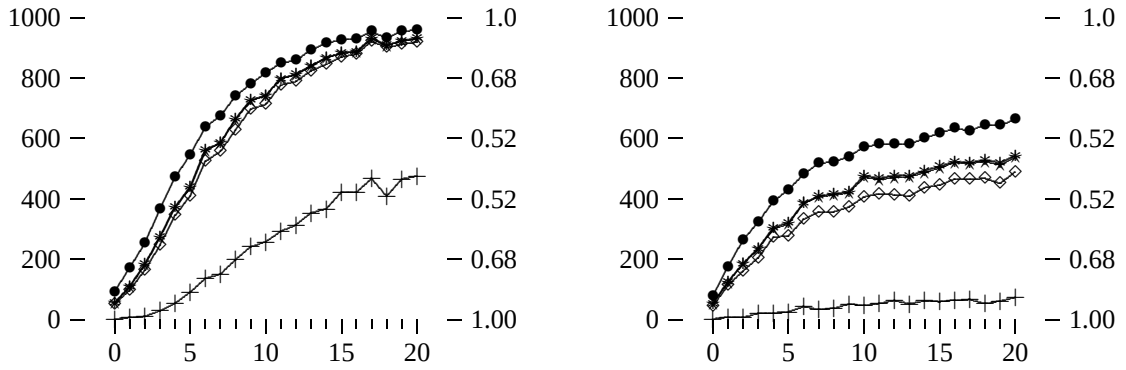
(b) C45-cf vs. kNN

Figure 6: Power of statistical tests for comparison of multiple classifiers. Bias is defined by the difference in performance of the two classifiers on the graph (left) or between the C4.5-cf and all other classifiers (right). The left scale on each graph gives the number of times the hypothesis was rejected and the right scale gives the corresponding $R(e)$.

These experiments again seem to favour the non-parametric tests over the parametric ones although not always as convincingly as in the case of comparisons of two classifiers. Due to the theoretical and practical advantages of the Friedman test (ease of computation and interpretation, the ability to present the overall performance of classifiers in form of ranks instead of the dubious averages), the Friedman test should be preferred over ANOVA. The corresponding non-parametric post-hoc tests give similar results, so it is upon the researcher to decide whether the slightly more powerful Hommel test is worth the complexity of its calculation as compared to the much simpler Holm test.



(a) C45-cf vs. naive Bayes



(b) C45-cf vs. kNN

Figure 7: Power of statistical tests for comparison of multiple classifiers with a control. Bias is defined by the difference in performance of the two classifiers on the graph (left) or between the C4.5-cf and the average of all other classifiers (right). The left scale on each graph gives the number of times the hypothesis was rejected and the right scale gives the corresponding Bouckaert's *R*.

5. Conclusion

Our analysis of the papers from the past International Conferences on Machine Learning has shown that many authors feel that the algorithms they propose should be compared over a set of problems and that the results can be used for drawing general conclusions. There is however no golden standard for making such comparisons and the tests performed often have dubious statistical foundations and lead to unwarranted and unverified conclusions.

While comparisons using a single data set are pestered by the biased variance estimations due to dependencies between the samples of examples drawn from the data set, in comparisons over multiple data set the variance comes from the differences between the data sets, which are usually

independent. Our setup is therefore free from the elevated Type 1 error that is common on the single data set testing procedures. The problems with the multiple data set tests are quite different, even in a sense complementary: the measurements from different data sets are usually incommensurate, and the normality of their distributions and the homogeneity of variance is questionable at best.

We theoretically and empirically analyzed three families of statistical tests that can be used for comparing two or more classifiers over multiple data sets: parametric tests (the paired t-test and ANOVA), non-parametric tests (the Wilcoxon and the Friedman test) and the non-parametric test that assumes no commensurability of the results (sign test). In the theoretical part, we specifically discussed the possible violations of the tests' assumptions by a typical machine learning data. Based on the well known statistical properties of the tests and our knowledge of the machine learning data, we concluded that the non-parametric tests should be preferred over the parametric ones.

We have observed the behaviour of the proposed statistics on several real-world classifiers and data sets. We varied the differences between the classifiers by biasing the selection of data sets, and measured the likelihood of rejection of the null-hypothesis and the replicability of the test. We have indeed found that the non-parametric tests are more likely to reject the null-hypothesis, which hints at the presence of outliers or violations of assumptions of the parametric tests and confirms our theoretical misgivings about them. The empirical analysis also shows that replicability of the tests might be a problem, thus the actual experiments should be conducted on as many data sets as possible.

In the empirical study we provided no analysis of Type 1/Type 2 error rates. The main reason for this is that the correct result—rejection or non-rejection of the null-hypothesis—is not well defined and depends upon the kind of difference between the algorithms we intend to measure. Besides, conducting the experiments in which we knew the true hypotheses would require artificial data sets and classifiers with the prescribed probabilities and distributions of errors. For this we would need to make some assumptions about the real-world distributions; these assumptions are, however, exactly what we were testing in the first place.

Overall, the non-parametric tests, namely the Wilcoxon and Friedman test are suitable for our problems. They are *appropriate* since they assume some, but limited commensurability. They are *safer than parametric tests* since they do not assume normal distributions or homogeneity of variance. As such, they can be applied to classification accuracies, error ratios or any other measure for evaluation of classifiers, including even model sizes and computation times. Empirical results suggest that they are also *stronger than the other tests studied*. The latter is particularly true when comparing a pair of classifiers.

We have proposed a visual representation of the post-hoc analysis when multiple classifiers are compared. CD diagrams are "space-friendly" and thus suitable when the length of the paper is an issue, yet they present the order of the algorithms, the magnitude of differences between them (in terms of ranks) and the significance of the observed differences much more clearly than it can be done in textual or in a pure numerical form.

There is an alternative opinion among statisticians that significance tests should not be performed at all since they are often misused, either due to misinterpretation or by putting too much stress on their results (Cohen, 1994; Schmidt, 1996; Harlow and Mulaik, 1997). Our stance is that statistical tests provide certain reassurance about the validity and non-randomness of the published results. For that to be true, they should be performed correctly and the resulting conclusions should be drawn cautiously. On the other hand, statistical tests should not be the deciding factor for or against publishing the work. Other merits of the proposed algorithm that are beyond the grasp of

statistical testing should also be considered and possibly even favoured over pure improvements in predictive power.

Acknowledgments

I wish to thank my colleagues from Artificial Intelligence Laboratory and Cognitive Modeling Laboratory at the Faculty of Computer and Information Science in Ljubljana, Slovenia, especially Blaž Zupan and Aleks Jakulin for their invaluable remarks and suggestions.

References

E. Alpaydın. Combined 5×2 F test for comparing supervised classification learning algorithms. *Neural Computation*, 11:1885–1892, 1999.

J. R. Beck and E. K. Schultz. The use of ROC curves in test performance evaluation. *Arch Pathol Lab Med*, 110:13–20, 1986.

R. Bellazzi and B. Zupan. Intelligent data analysis in medicine and pharmacology: a position statement. In *IDAMAP Workshop Notes at the 13th European Conference on Artificial Intelligence, ECAI-98*, Brighton, UK, 1998.

Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

R. R. Bouckaert. Choosing between two learning algorithms based on calibrated tests. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. AAAI Press, 2003.

R. R. Bouckaert. Estimating replicability of classifier learning experiments. In C Brodley, editor, *Machine Learning, Proceedings of the Twenty-First International Conference (ICML 2004)*. AAAI Press, 2004.

R. R. Bouckaert and E. Frank. Evaluating the replicability of significance tests for comparing learning algorithms. In D. Honghua, R. Srikant, and C. Zhang, editors, *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*. Springer, 2004.

P. B. Brazdil and C. Soares. A comparison of ranking methods for classification algorithm selection. In *Proceedings of 11th European Conference on Machine Learning*. Springer Verlag, 2000.

W. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:329–336, 1979.

J. Cohen. The earth is round ($p < .05$). *American Psychologist*, 49:997 1003, 1994.

J. Demšar and B. Zupan. *Orange: From Experimental Machine Learning to Interactive Data Mining, A White Paper*. Faculty of Computer and Information Science, Ljubljana, Slovenia, 2004.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1924, 1998.

O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

C. W. Dunnett. A multiple comparison procedure for comparing several treatments with a control. *Journal of American Statistical Association*, 50:1096–1121, 1980.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, Chambery, France, 1993. Morgan-Kaufmann.

R. A. Fisher. *Statistical methods and scientific inference (2nd edition)*. Hafner Publishing Co., New York, 1959.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

L. C. Hamilton. *Modern Data Analysis: A First Course in Applied Statistics*. Wadsworth, Belmont, California, 1990.

L. L. Harlow and S. A. Mulaik, editors. *What If There Were No Significance Tests?* Lawrence Erlbaum Associates, July 1997.

Y. Hochberg. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75: 800–803, 1988.

B. Holland. On the application of three modified Bonferroni procedures to pairwise multiple comparisons in balanced repeated measures designs. *Computational Statistics Quarterly*, 6:219–231, 1991.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

G. Hommel. A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, 75:383–386, 1988.

D. A. Hull. *Information Retrieval Using Statistical Classification*. PhD thesis, Stanford University, November 1994.

R. L. Iman and J. M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, pages 571–595, 1980.

P. Langley. Crafting papers on machine learning. In *Proc. of Seventeenth International Conference on Machine Learning (ICML-2000)*, 2000.

D. Mladenić and M. Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In I. Bratko and S. Džeroski, editors, *Machine Learning, Proceedings of the Sixteenth International Conference (ICML 1999), June 27-30, 2002, Bled, Slovenia*, pages 258–267. Morgan Kaufmann, 1999.

C. Nadeau and Y. Bengio. Inference for the generalization error. *Advances in Neural Information Processing Systems*, 12:239–281, 2000.

P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

J. Pizarro, E. Guerrero, and P. L. Galindo. Multiple comparison procedures applied to model selection. *Neurocomputing*, 48:155–173, 2002.

F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In J. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-1998)*, pages 445–453, San Francisco, CA, 1998. Morgan Kaufmann Publishers.

J. R. Quinlan. Bagging, boosting, and c4.5. In *Proc. Thirteenth National Conference on Artificial Intelligence*, pages 725–730, Portland, OR, 1996. AAAI Press.

S. L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328, 1997.

F. L. Schmidt. Statistical significance testing and cumulative knowledge in psychology. *Psychological Methods*, 1:115–129, 1996.

H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *SIGIR'95, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–237. ACM Press, 1995.

J. P. Shaffer. Multiple hypothesis testing. *Annual Review of Psychology*, 46:561–584, 1995.

D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC, 2000.

J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5:99–114, 1949.

E. G. Vázquez, A. Y. Escolano, and J. P. Junquera P. G. Riaño. Repeated measures multiple comparison procedures applied to model selection in neural networks. In *Proc. of the 6th Intl. Conf. On Artificial and Natural Neural Networks (IWANN 2001)*, pages 88–95, 2001.

G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40:159–197, 2000.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

J. H. Zar. *Biostatistical Analysis (4th Edition)*. Prentice Hall, Englewood Clifs, New Jersey, 1998.

Journal of Machine Learning Research 7 (2006) 31–54 Submitted 3/05; Revised 11/05; Published 1/06

Incremental Algorithms for Hierarchical Classification

Nicolò Cesa-Bianchi CESA-BIANCHI@DSI.UNIMI.IT
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39
20135 Milano, Italy

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
Dipartimento di Informatica e Comunicazione
Università dell'Insubria
via Mazzini 5
21100 Varese, Italy

Luca Zaniboni ZANIBONI@DTI.UNIMI.IT
Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
via Bramante 65
26013 Crema (CR), Italy

Editor: Michael Collins

Abstract

We study the problem of classifying data in a given taxonomy when classifications associated with multiple and/or partial paths are allowed. We introduce a new algorithm that incrementally learns a linear-threshold classifier for each node of the taxonomy. A hierarchical classification is obtained by evaluating the trained node classifiers in a top-down fashion. To evaluate classifiers in our multipath framework, we define a new hierarchical loss function, the H-loss, capturing the intuition that whenever a classification mistake is made on a node of the taxonomy, then no loss should be charged for any additional mistake occurring in the subtree of that node.

Making no assumptions on the mechanism generating the data instances, and assuming a linear noise model for the labels, we bound the H-loss of our on-line algorithm in terms of the H-loss of a reference classifier knowing the true parameters of the label-generating process. We show that, in expectation, the excess cumulative H-loss grows at most logarithmically in the length of the data sequence. Furthermore, our analysis reveals the precise dependence of the rate of convergence on the eigenstructure of the data each node observes.

Our theoretical results are complemented by a number of experiments on texual corpora. In these experiments we show that, after only one epoch of training, our algorithm performs much better than Perceptron-based hierarchical classifiers, and reasonably close to a hierarchical support vector machine.

Keywords: incremental algorithms, online learning, hierarchical classification, second order perceptron, support vector machines, regret bound, loss function

1. Introduction

In this paper, we investigate the problem of classifying data based on the knowledge that the graph of dependencies between the classes is a tree forest. The trees in this forest are collectively interpreted

as a taxonomy. That is, we assume that every data instance is labelled with a (possibly empty) set of class nodes and, whenever an instance is labelled with a certain node i, then it is also labelled with all the nodes on the path from the root of the tree where i occurs down to node i. A distinctive feature of our framework is that we also allow multiple-path labellings (instances can be labelled with nodes belonging to more than one path in the forest), and partial-path labellings (instances can be labelled with nodes belonging to a path that does not end on a leaf).

We introduce a new algorithm that incrementally learns a linear-threshold classifier for each node of the taxonomy. A hierarchical classification is then obtained by evaluating the node classifiers in a top-down fashion, so that the final labelling is consistent with the taxonomy.

The problem of hierarchical classification, especially of textual information, has been extensively investigated in past years (see, e.g., Dumais and Chen, 2000; Dekel et al., 2004, 2005; Granitzer, 2003; Hofmann et al., 2003; Koller and Sahami, 1997; McCallum et al., 1998; Mladenic, 1998; Ruiz and Srinivasan, 2002; Sun and Lim, 2001, and references therein). The on-line approach to hierarchical classification, which we analyze here, seems well suited when dealing with scenarios in which new data are produced frequently and in large amounts (e.g., data produced by newsfeeds—considered in this paper, or the speech data considered in Dekel et al., 2005).

An important ingredient in a hierarchical classification problem is the loss function used to evaluate the classifier's performance. In pattern classification the zero-one loss is traditionally used. In a hierarchical setting this loss would simply count one mistake each time, on a given data instance, the set of class labels output by the hierarchical classifier is not perfectly identical to the set of true labels associated to that instance. Loss functions able to reflect the taxonomy structure have been proposed in the past (e.g., Dekel et al., 2004; Hofmann et al., 2003; Sun and Lim, 2001), but none of these losses works well in our framework where multiple and partial paths are allowed. In this paper we define a new loss function, the H-loss (hierarchical loss), whose simple definition captures the following intuition: "if a mistake is made at node i of the taxonomy, then further mistakes made in the subtree rooted at i are unimportant". In other words, we do not require the algorithm be able to make fine-grained distinctions on tasks where it is unable to make coarse-grained ones. For example, if an algorithm failed to label a document with the class SPORTS, then the algorithm should not be charged more loss because it also failed to label the same document with the subclass SOCCER and the sub-subclass CHAMPIONS LEAGUE.

We bound the theoretical performance of our algorithm using the H-loss. In our analysis, we make no assumptions on the mechanism generating the data instances; that is, we bound the H-loss of the algorithm for any arbitrary sequence of data instances. The hierarchical labellings associated to the instances, instead, are assumed to be independently generated according to a parametric stochastic process defined on the taxonomy.

Following a standard approach in the analysis of on-line algorithms, we measure the predictive performance using the cumulative regret, a quantity measuring the difference between the cumulative H-loss of the classifiers incrementally generated by the on-line algorithm during its run and the cumulative H-loss of a fixed reference classifier. Our main theoretical result is a bound on the regret of our hierarchical learning algorithm with respect to a reference hierarchical classifier based on the true parameters of the label-generating process. More specifically, we bound the contribution to the cumulative regret of each node classifier in terms of quantities related to the position of the node in the taxonomy and the data process parameters. This interaction between node position and data process parameters captures the hierarchical nature of the classification problem since the contribution of each node to the overall cumulative regret decreases as we proceed downward from a root

in the forest. In general, the overall cumulative regret is seen to grow at most logarithmically in the length T of the data sequence.

From the theoretical point of view, the novelty of this line of research is twofold:

1. The use of hierarchically trained linear-threshold classifiers is common to several of the previous approaches to hierarchical classification (e.g., Dumais and Chen, 2000; Dekel et al., 2004, 2005; Granitzer, 2003; Hofmann et al., 2003; Koller and Sahami, 1997; McCallum et al., 1998; Mladenic, 1998; Ruiz and Srinivasan, 2002; Sun and Lim, 2001). However, to our knowledge, this research is the first one to provide a rigorous performance analysis of hierarchical classification algorithms in the presence of multiple and partial path classifications.

2. The core of our analysis is a local cumulative regret bound showing that the instantaneous regret of each node classifier vanishes at a rate $1/T$. The precise dependence of the rate of convergence on the eigenstructure of the data each node observes is a major contribution of this paper. This turns out to be similar in spirit to early (and classical) work in least-squares linear regression (e.g., Lai et al., 1979; Lai and Wei, 1982). But unlike these previous investigations, our analysis is not asymptotic in nature and studies a specific classification setting, instead of a regression one.

To support our theoretical findings we also describe some experiments concerning a more practical variant of the algorithm we actually analyze. These experiments use large corpora of textual data on which we test different batch and incremental classifiers. The experiments show that our on-line algorithm performs significantly better than Perceptron-based hierarchical classifiers. Furthermore, after only one epoch of training, our algorithm achieves a performance close to that of a hierarchical support vector machine, the popular batch learning algorithm for which, to the best of our knowledge, no theoretical performance bounds are known in hierarchical classification frameworks.

The paper is organized as follows. Section 2 defines the notation used throughout the paper. In Section 3 we introduce the H-loss function. Our hierarchical algorithm is described in Section 4. In Section 5 and 6 we define the data model, the learning model, and our theoretical performance measure: the cumulative regret. The analysis of our algorithm is carried out in Section 7, while in Section 8 we report on the experiments. Finally, in Section 9 we summarize our results and mention a few open questions.

2. Notation

We assume data elements are encoded as unit-norm vectors $x \in \mathbb{R}^d$, which we call *instances*. A *multilabel* for an instance x is any subset of the set $\{1,\ldots,N\}$ of all labels, including the empty set. We represent the multilabel of x with a vector $v = (v_1,\ldots,v_N) \in \{0,1\}^N$, where $i \in \{1,\ldots,N\}$ belongs to the multilabel of x if and only if $v_i = 1$.

A *taxonomy* G is a forest whose trees are defined over the set of labels. A multilabel $v \in \{0,1\}^N$ is said to *respect* a taxonomy G if and only if v is the union of one or more paths in G, where each path starts from a root but need not terminate on a leaf, see Figure 1. We assume the data-generating mechanism produces examples (x,v) such that v respects some fixed underlying taxonomy G with N nodes (see Section 5). The set of roots in G is denoted by $\mathrm{ROOT}(G)$. We use $\mathrm{PAR}(i)$ to denote the unique parent of node i, $\mathrm{ANC}(i)$ to denote the set of ancestors of i, $\mathrm{SUB}(i)$ to denote the set of nodes in the subtree rooted at i (including i), and $\mathrm{CHILD}(i)$ to denote the set of children of node i.

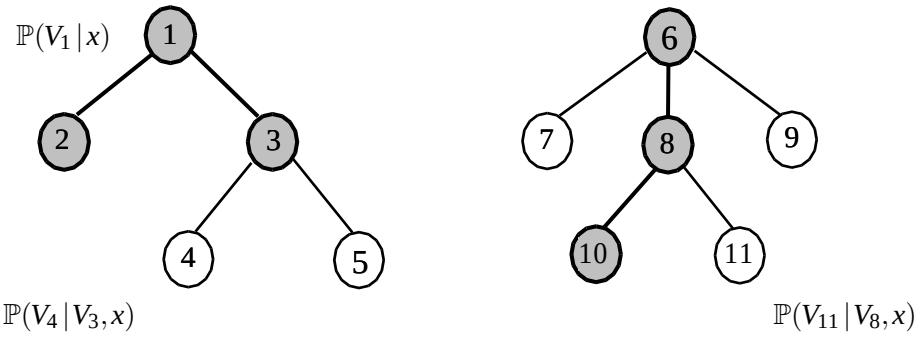


Figure 1: A forest made of two disjoint trees. The nodes are tagged with the name of the labels, so that in this case $N = 11$. According to our definition, the multilabel $v = (1,1,1,0,0,1,0,1,0,1,0)$ respects this taxonomy (since it is the union of paths $1 \to 2$, $1 \to 3$ and $6 \to 8 \to 10$), while the multilabel $v = (1,1,0,1,0,0,0,0,0,0,0)$ does not, since $1 \to 2 \to 4$ is not a path in the forest. Associated with each node i is a $\{0,1\}$-valued random variable V_i distributed according to a conditional probability function $\mathbb{P}(V_i \mid V_{\text{PAR}(i)}, x)$ —see Section 5.

We denote by $\{\phi\}$ the Bernoulli random variable which is 1 if and only if predicate ϕ is true. In our analysis, we repeatedly use simple facts such as $\{\phi \vee \psi\} = \{\phi\} + \{\psi \wedge \neg\phi\} \leq \{\phi\} + \{\psi\}$ and $\{\phi\} = \{\phi \wedge \psi\} + \{\phi \wedge \neg\psi\} \leq \{\phi \wedge \psi\} + \{\neg\psi\}$, where ψ is another predicate.

3. The H-Loss

Two very simple loss functions, measuring the discrepancy between the prediction multilabel $\widehat{y} = (\widehat{y}_1, \ldots, \widehat{y}_N)$ and the true multilabel $v = (v_1, \ldots, v_N)$, are the zero-one loss $\ell_{0/1}(\widehat{y}, v) = \{\exists i : \widehat{y}_i \neq v_i\}$ and the symmetric difference loss $\ell_\Delta(\widehat{y}, v) = \{\widehat{y}_1 \neq v_1\} + \ldots + \{\widehat{y}_N \neq v_N\}$. Note that the definition of these losses is based on the set $\{1, \ldots, N\}$ of labels without any additional structure. A loss function that takes into account a taxonomical structure defined over the set of labels is

$$\ell_H(\widehat{y}, v) = \sum_{i=1}^{N} \{\widehat{y}_i \neq v_i \wedge \widehat{y}_j = v_j,\ j \in \text{ANC}(i)\} \ .$$

This loss, which we call H-loss (hierarchical loss), can also be defined as follows: all paths in G from a root down to a leaf are examined and, whenever a node i is encountered such that $\widehat{y}_i \neq v_i$, then 1 is added to the loss, while all the loss contributions in the subtree rooted at i are discarded. Note that, with this definition, $\ell_{0/1} \leq \ell_H \leq \ell_\Delta$. A graphical representation of the H-loss and related concepts is given in Figure 2.

In the next lemma we show an important (and intuitive) property of the H-loss: when the multilabel v to be predicted respects a taxonomy G then there is no loss of generality in restricting to predictions which respect G. Formally, given a multilabel $\widehat{y} \in \{0,1\}^N$, we define the *$G$-truncation* of $\widehat{y}$ as the multilabel $\widehat{y}' = (\widehat{y}'_1, \ldots, \widehat{y}'_N) \in \{0,1\}^N$ where, for each $i = 1, \ldots, N$, $\widehat{y}'_i = 1$ if and only if $\widehat{y}_i = 1$ and $\widehat{y}_j = 1$ for all $j \in \text{ANC}(i)$. Note that the G-truncation of any multilabel always respects G. The next lemma states that if v respects G, then $\ell_H(\widehat{y}, v)$ cannot be smaller than $\ell_H(\widehat{y}', v)$.

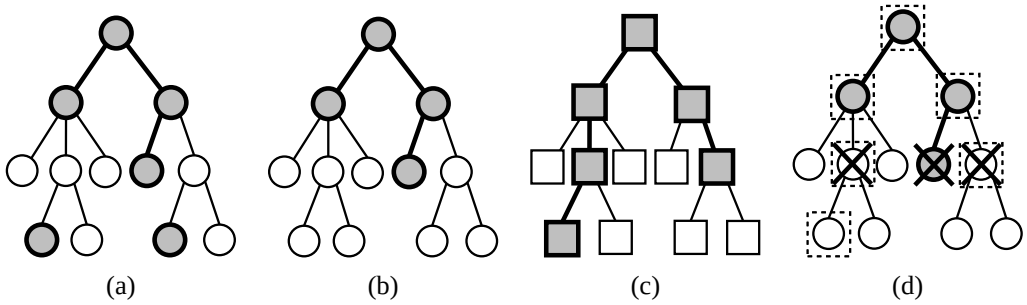


Figure 2: A one-tree forest (repeated four times). Each node corresponds to a class in the taxonomy G, hence in this case $N = 12$. Gray nodes are included in the multilabel under consideration, white nodes are not. (a) A generic multilabel which *does not* respect G; (b) its G-truncation. (c) A second multilabel that respects G. (d) Superposition of multilabel (b) on multilabel (c): Only the checked nodes contribute to the H-loss between (b) and (c). Hence the H-loss between multilabel (b) and multilabel (c) is 3. Here the zero-one loss between (b) and (c) is 1, while the symmetric difference loss equals 4.

Lemma 1 *Let G be a taxonomy, $v,\widehat{y} \in \{0,1\}^N$ be two multilabels such that v respects G, and $\widehat{y}'$ be the G-truncation of $\widehat{y}$. Then*

$$\ell_H(\widehat{y}',v) \le \ell_H(\widehat{y},v) \ .$$

Proof. Since $\ell_H(\widehat{y}',v) = \sum_{i=1}^N \{\widehat{y}'_i \neq v_i \wedge \widehat{y}'_j = v_j,\ j \in \text{ANC}(i)\}$ and $\ell_H(\widehat{y},v) = \sum_{i=1}^N \{\widehat{y}_i \neq v_i \wedge \widehat{y}_j = v_j,\ j \in \text{ANC}(i)\}$, it suffices to show that, for each $i = 1,\ldots,N$, $\widehat{y}'_i \neq v_i$ and $\widehat{y}'_j = v_j$ for all $j \in \text{ANC}(i)$ implies $\widehat{y}_i \neq v_i$ and $\widehat{y}_j = v_j$ for all $j \in \text{ANC}(i)$.

Pick some i and suppose $\widehat{y}'_i \neq v_i$ and $\widehat{y}'_j = v_j$ for all $j \in \text{ANC}(i)$. Now suppose $\widehat{y}'_j = 0$ (and thus $v_j = 0$) for some $j \in \text{ANC}(i)$. Then $v_i = 0$ since v respects G. But this implies $\widehat{y}'_i = 1$, contradicting the fact that the G-truncation $\widehat{y}'$ respects G. Therefore, it must be the case that $\widehat{y}'_j = v_j = 1$ for all $j \in \text{ANC}(i)$. Hence the G-truncation of $\widehat{y}$ left each node $j \in \text{ANC}(i)$ unchanged, implying $\widehat{y}_j = v_j$ for all $j \in \text{ANC}(i)$. But, since the G-truncation of $\widehat{y}$ does not change the value of a node i whose ancestors j are such that $\widehat{y}_j = 1$, this also implies $\widehat{y}_i = \widehat{y}'_i$. Therefore $\widehat{y}_i \neq v_i$ and the proof is concluded. □

4. A New Hierarchical Learning Algorithm

In this section we describe our on-line algorithm for hierarchical classification. Its theoretical performance is analyzed in Section 7.

The on-line learning model we consider is the following. In the generic time step $t = 1,2,\ldots$ instance x_t is revealed to the algorithm which outputs the prediction $\widehat{y}_t = (\hat{y}_{1,t},\ldots,\hat{y}_{N,t}) \in \{0,1\}^N$. This is viewed as a guess for the multilabel $v_t = (v_{1,t},v_{2,t},\ldots,v_{N,t})$ associated with the current instance x_t. After each prediction, the algorithm observes the true multilabel v_t and adjusts its parameters for the next prediction.

Our algorithm computes $\hat{y}_{1,t},\ldots,\hat{y}_{N,t}$ using N linear-threshold classifiers, one for each node in the taxonomy. These node classifiers are evaluated, starting from each root, in the following top-down fashion: the root is labelled by evaluating its node classifier; if a node has been labelled 1, then each child is labelled by evaluating its node classifier. On the other hand, if a node is labelled

Algorithm H-RLS.
Initialization: Weight vectors $w_{i,1} = (0,\ldots,0)$, $i = 1,\ldots,N$.
For $t = 1,2,\ldots$ do

1. Observe instance $x_t \in \{x \in \mathbb{R}^d : ||x|| = 1\}$;
2. For each $i = 1,\ldots,N$ compute predictions $\hat{y}_{i,t} \in \{0,1\}$ as follows:

$$\hat{y}_{i,t} = \begin{cases} \{w_{i,t}^\top x_t \geq 0\} & \text{if } i \text{ is a root node,} \\ \{w_{i,t}^\top x_t \geq 0\} & \text{if } i \text{ is not a root node and } \hat{y}_{j,t} = 1 \text{ for } j = \text{PAR}(i), \\ 0 & \text{if } i \text{ is not a root node and } \hat{y}_{j,t} = 0 \text{ for } j = \text{PAR}(i), \end{cases}$$

where

$$\begin{aligned} w_{i,t} &= (I + S_{i,Q(i,t-1)} S_{i,Q(i,t-1)}^\top + x_t x_t^\top)^{-1} \times \\ &\qquad \times S_{i,Q(i,t-1)} \left(v_{i,i_1}, v_{i,i_2}, \ldots, v_{i,i_{Q(i,t-1)}}\right)^\top \\ S_{i,Q(i,t-1)} &= [x_{i_1} x_{i_2} \ldots x_{i_{Q(i,t-1)}}] \qquad\qquad i = 1,\ldots,N. \end{aligned}$$

3. Observe multilabel v_t and update weights.

Figure 3: The hierarchical learning algorithm H-RLS.

0 then *all* of its descendants are labelled 0. Note that this evaluation scheme can only generate multilabels that respect the underlying taxonomy.

Let $w_1,\ldots,w_N$ be the weight vectors defining the linear-threshold classifiers used by the algorithm. A feature of the learning process, which is also important for its theoretical analysis, is that the classifier at node i is only trained on the examples that are positive for its parent node. In other words, w_i is considered for update only on those instances x_t such that $v_{\text{PAR}(i),t} = 1$.

Let $Q(i,t)$ denote the number of times the *parent* of node i observes a positive label up to time t, i.e., $Q(i,t) = |\{1 \leq s \leq t : v_{\text{PAR}(i),s} = 1\}|$. The weight vector $w_{i,t}$ stored at time t in node i is a (conditional) regularized least squares estimator given by

$$w_{i,t} = \left(I + S_{i,Q(i,t-1)} S_{i,Q(i,t-1)}^\top + x_t x_t^\top\right)^{-1} S_{i,Q(i,t-1)} \left(v_{i,i_1}, \ldots, v_{i,i_{Q(i,t-1)}}\right)^\top, \tag{1}$$

where I is the $d \times d$ identity matrix, $S_{i,Q(i,t-1)}$ is the $d \times Q(i,t-1)$ matrix whose columns are the instances $x_{i_1},\ldots,x_{i_{Q(i,t-1)}}$, and $\left(v_{i,i_1},\ldots,v_{i,i_{Q(i,t-1)}}\right)^\top$ is the $Q(i,t-1)$-dimensional (column) vector of the corresponding labels observed by node i.

The estimator in (1) is a slight variant of the regularized least squares estimator for classification (Cesa-Bianchi et al., 2002; Rifkin et al., 2003) where we include the current instance x_t in the computation of $w_{i,t}$ (see, e.g., Azoury and Warmuth, 2001; Vovk, 2001, for analyses of similar algorithms in different contexts). Efficient incremental computations of the inverse matrix and

dual variable formulations of the algorithm are extensively discussed by Cesa-Bianchi et al. (2002) and Rifkin et al. (2003).

The pseudocode of our algorithm, which we call H-RLS (Hierarchical Regularized Least Squares) is given in Figure 3.

5. A Stochastic Model for Generating Labels

While no assumptions are made on the mechanism generating the sequence $x_1, x_2, \ldots$ of instances, we base our analysis on the following stochastic model for generating the multilabel associated to an instance x_t.

A probability distribution f_G over the set of multilabels is associated to a taxonomy G as follows. Each node i of G is tagged with a $\{0,1\}$-valued random variable V_i distributed according to a conditional probability function $\mathbb{P}(V_i \mid V_{\text{PAR}(i)}, x)$. To model the dependency between the labels of nodes i and $j = \text{PAR}(i)$ we assume

$$\mathbb{P}(V_i = 1 \mid V_j = 0, x) = 0 \tag{2}$$

for all nonroot nodes i and all instances x. For example, in the taxonomy of Figure 1 we have $\mathbb{P}(V_4 = 1 \mid V_3 = 0, x) = 0$ for all $x \in \mathbb{R}^d$. The quantity

$$f_G(v \mid x) = \prod_{i=1}^{N} \mathbb{P}(V_i = v_i \mid V_j = v_j,\, j = \text{PAR}(i), x)$$

thus defines a joint probability distribution on $V_1, \ldots, V_N$ conditioned on x being the current instance. This joint distribution puts zero probability on all multilabels $v \in \{0,1\}^N$ which do not respect G.

Through f_G we specify an i.i.d. process $\{V_1, V_2, \ldots\}$ as follows. We assume that an arbitrary and unknown sequence of instance vectors $x_1, x_2, \ldots$ is fixed in advance, where $\|x_t\| = 1$ for all t. The multilabel V_t is distributed according to the joint distribution $f_G(\cdot \mid x_t)$. We call each pair (x_t, v_t), where v_t is a realization of V_t, an *example*.

Let us now introduce a parametric model for f_G. With each node i in the taxonomy, we associate a unit-norm weight vector $u_i \in \mathbb{R}^d$. Then, we define the conditional probabilities for a nonroot node i with parent j as follows:

$$\mathbb{P}(V_i = 1 \mid V_j = 1, x) = \frac{1 + u_i^\top x}{2}\,. \tag{3}$$

If i is a root node, the above simplifies to

$$\mathbb{P}(V_i = 1 \mid x) = \frac{1 + u_i^\top x}{2}\,.$$

Our choice of a linear model for Bernoulli random variables, as opposed to a more standard log-linear model, is mainly motivated by our intention of proving regret bounds with no assumptions on the way the sequence of instances is generated. Indeed, we are not aware of any analysis of logistic regression holding in a similar classification setup.

Note also that, in this model, the labels of the children of any given node are independent random variables. This is motivated by the fact that, unlike previous investigations, we are expliciteply modelling labellings involving multiple paths. A more sophisticated analysis could introduce arbitrary negative correlations among the labels of the children nodes. In this paper, however, we do not follow this route.

6. Regret and the Reference Classifier

Assuming the stochastic model described in Section 5, we compare the performance of our algorithm to the performance of the fixed hierarchical classifier built on the true parameters $u_1,\dots,u_N$ governing the label-generating process. This reference hierarchical classifier has the same form as the classifiers generated by H-RLS. More precisely, let the multilabel $y=(y_1,\dots,y_N)$ for an instance x be computed as follows:

$$y_i=\begin{cases}\{u_i^\top x\ge 0\} & \text{if } i \text{ is a root node,}\\ \{u_i^\top x\ge 0\} & \text{if } i \text{ is not a root and } y_j=1 \text{ for } j=\mathrm{PAR}(i),\\ 0 & \text{if } i \text{ is not a root and } y_j=0 \text{ for } j=\mathrm{PAR}(i).\end{cases} \tag{4}$$

To evaluate our algorithm against the reference hierarchical classifier defined in (4), we use the cumulative regret. Given any loss function ℓ (such as one of the three defined in Section 3), we define the (instantaneous) *regret* of a classifier assigning label $\widehat{y}_t$ to instance x_t as

$$\mathbb{E}\,\ell(\widehat{y}_t,V_t)-\mathbb{E}\,\ell(y_t,V_t)\ ,$$

where y_t is the multilabel assigned by classifier (4), and the expectation is with respect the random draw of V_t (as specified in Section 5). We measure the performance of H-RLS through its cumulative regret on a sequence of T examples:

$$\sum_{t=1}^{T}\big(\mathbb{E}\,\ell(\widehat{y}_t,V_t)-\mathbb{E}\,\ell(y_t,V_t)\big)\ . \tag{5}$$

The regret bound we prove in Section 7 holds when $\ell=\ell_H$, and is shown to depend on the interaction between the spectral structure of the data generating process and the structure of the taxonomy on which the process is applied.

7. Analysis

We now prove a bound on the cumulative regret of H-RLS with respect to the H-loss function ℓ_H. Our analysis hinges on proving that for any node i, the estimated margin $w_{i,t}^\top x_t$ is an asymptotically unbiased estimator of the true margin $u_i^\top x_t$, and then on using known large deviation arguments to obtain the stated bound. For this purpose, we bound the variance of the margin estimator at each node and prove a bound on the rate at which the bias vanishes.

Theorem 2 *Consider a taxonomy G with N nodes. Pick any set of model parameters $u_1,\dots,u_N\in\mathbb{R}^d$ such that $\|u_i\|=1$ for $i=1,\dots,N$, and pick any sequence of instance vectors $x_1,x_2,\dots\in\mathbb{R}^d$ such that $\|x_t\|=1$ for all t. Then the cumulative regret of the* H-RLS *algorithm (described in Figure 3) satisfies, for each $T\ge 1$,*

$$\sum_{t=1}^{T}\big(\mathbb{E}\,\ell_H(\widehat{y}_t,V_t)-\mathbb{E}\,\ell_H(y_t,V_t)\big)\le 16(1+1/e)\sum_{i=1}^{N}\frac{C_i}{\Delta_i^2}\,\mathbb{E}\left[\sum_{j=1}^{d}\log(1+\lambda_{i,j})\right]\ ,$$

where

$$\Delta_{i,t}=u_i^\top x_t,\qquad \Delta_i^2=\min_{t=1,\dots,T}\Delta_{i,t}^2,\qquad C_i=|\mathrm{SUB}(i)|,$$

$\lambda_{i,1},\dots,\lambda_{i,d}$ are the eigenvalues of matrix $S_{i,Q(i,T)}S_{i,Q(i,T)}^\top$, and e is the base of natural logarithms.

Before delving into the proof, it is worth making a few comments.

Remark 3 Since H-RLS can be cast in dual variables, we can run it in any reproducing kernel Hilbert space (e.g., Schölkopf and Smola, 2002). The regret bound contained in Theorem 2 remains true once we observe that the nonzero eigenvalues of $S_{i,Q(i,T)} S_{i,Q(i,T)}^\top$ coincide with the nonzero eigenvalues of the Gram matrix $S_{i,Q(i,T)}^\top S_{i,Q(i,T)}$, and we replace the sum over all input dimensions d with the sum over the (at most T) nonzero eigenvalues of $S_{i,Q(i,T)}^\top S_{i,Q(i,T)}$. We refer the reader to the work by Cesa-Bianchi et al. (2002) for additional details.

Remark 4 It is important to emphasize the interplay between the taxonomy structure and the process generating the examples, as expressed by the above regret bound. Recall that we denote by $\lambda_{i,1}, \ldots, \lambda_{i,d}$ the eigenvalues of matrix $S_{i,Q(i,T)} S_{i,Q(i,T)}^\top$. From the previous remark we have $\sum_{j=1}^d \lambda_{i,j} = \mathrm{trace}\big(S_{i,Q(i,T)}^\top S_{i,Q(i,T)}\big) = Q(i,T)$ since $\|x_t\| = 1 \; \forall t$, and

$$\sum_{j=1}^d \log(1+\lambda_{i,j}) \le \max\left\{ \sum_{j=1}^d \log(1+\mu_j) \,:\, \sum_{j=1}^d \mu_j = Q(i,T) \right\} = d \log\left(1 + \frac{Q(i,T)}{d}\right).$$

Moreover, $Q(i,T)$ is the sum of T Bernoulli random variables, where the t-th variable takes value 1 when the parent of the i-th node in the taxonomy observes label $V_{\mathrm{PAR}(i),t} = 1$ at time t. The probability of this event clearly equals

$$\prod_{j \in \mathrm{ANC}(i)} \left(\frac{1+\Delta_{j,t}}{2}\right).$$

Thus

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^d \log(1+\lambda_{i,j})\right] &\le d\,\mathbb{E}\left[\log\left(1+\frac{Q(i,T)}{d}\right)\right] && (6)\\
&\le d \log\left(1+\frac{\mathbb{E}\,Q(i,T)}{d}\right) \\
&\quad \text{(from Jensen's inequality)} \\
&= d \log\left(1+\frac{\sum_{t=1}^T \prod_{j \in \mathrm{ANC}(i)} \left(\frac{1+\Delta_{j,t}}{2}\right)}{d}\right). && (7)
\end{aligned}$$

Bound (6) is obviously a $\log T$ cumulative regret bound, since $Q(i,T) \le T$ anyway. It is important, however, to see how the regret bound depends on the taxonomy structure. Let us focus on (7). If i is a root node then $\mathbb{E}\,Q(i,T) = Q(i,T) = T$ (since a root node observes all labels). As we descend along a path, $\mathbb{E}\,Q(i,T)$ tends to decrease with a rate depending on the margins achieved by the ancestors of node i. Bound (7) thus makes explicit the contribution of node i to the overall regret. If i is a root node, then its contribution to the overall regret is roughly $\log T$. On the other hand, the deeper is node i within the taxonomy the smaller is the contribution of node i to the overall regret. A very deep leaf node observes a possibly small subset of the instances, but it is also required to produce only a small subset of linear-threshold predictions, i.e., the associated weight vector $w_{i,t}$ might be an unreliable estimator, but is also used less often. Therefore, the contribution of leaf node i is smaller than $\log T$ because the hierarchical nature of the problem (as expressed by the H-loss) lowers the relative importance of the accuracy of estimator $w_{i,t}$ when computing the overall regret.

Remark 5 Nothing prevents us from generalizing the H-loss by associating fixed cost coefficients to each taxonomy node:

$$\ell_H(\widehat{y}, v) = \sum_{i=1}^{N} c_i \left\{ \widehat{y}_i \neq v_i \wedge \widehat{y}_j = v_j,\, j \in \mathrm{ANC}(i) \right\} ,$$

where the cost coefficients c_i are positive real numbers. It is straightforward to see that with this definition of H-loss, the statement of Theorem 2 still holds, once we generalize the regret factors C_i as $C_i = \sum_{k \in \mathrm{SUB}(i)} c_k$. Note that this would involve changes neither in our learning algorithm nor in our reference predictor. In fact, we are measuring regret against a reference predictor that is not Bayes optimal for the data model at hand. This is not immediate to see when the cost coefficients c_i defining the H-loss are all set to 1 but, as we mentioned, it is generally evinced by the fact that both the reference predictor (4) and our learning algorithm do not depend on the c_i.

Remark 6 From the proof of Theorem 2 below, the reader can see that there are several ways one can improve the bounds. In fact, we made no special effort to minimize the main constant $16(1+1/e)$ and, in general, we disregarded quite a lot of constant factors throughout. Moreover, though we decided to cast the bounds in terms of the worst-case margin $\Delta_i^2 = \min_{t=1,\ldots,T} \Delta_{i,t}^2$, it is straighforward to modify the proof to obtain a bound depending on some sort of average squared margin. Since this sharper bound would hide the clean dependence on the eigenstructure of the data, we decided not to pursue this optimization any further.

We are now ready to prove Theorem 2.

Proof of Theorem 2. We fix a node i and upper bound its contribution to the total instantaneous regret. Since for any four predicates ϕ, ψ, χ, ζ we have $\{\phi \wedge \psi\} - \{\chi \wedge \zeta\} \leq \{\phi \wedge \psi \wedge \neg\chi\} + \{\phi \wedge \psi \wedge \chi \wedge \neg\zeta\}$, we see that

$$\begin{aligned}
&\{\widehat{y}_{i,t} \neq V_{i,t}, \forall j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} = V_{j,t}\} - \{y_{i,t} \neq V_{i,t}, \forall j \in \mathrm{ANC}(i) : y_{j,t} = V_{j,t}\} \\
&\quad \leq \left\{\widehat{y}_{i,t} \neq V_{i,t}, y_{i,t} = V_{i,t}, \forall j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} = V_{j,t}\right\} \qquad (8) \\
&\qquad + \left\{\widehat{y}_{i,t} \neq V_{i,t}, y_{i,t} \neq V_{i,t}, \forall j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} = V_{j,t}, \exists j \in \mathrm{ANC}(i) : y_{j,t} \neq V_{j,t}\right\} . \qquad (9)
\end{aligned}$$

We bound the two terms (8) and (9) separately. We can write:

$$\begin{aligned}
(8) &= \{\widehat{y}_{i,t} \neq V_{i,t}, y_{i,t} = V_{i,t}, \forall j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} = V_{j,t} = 1\} \\
&\qquad \text{(since } \widehat{y}_{j,t} = V_{j,t} = 0 \text{ for some ancestor } j \text{ implies } \widehat{y}_{i,t} = V_{i,t} = 0\text{)} \\
&\leq \{\widehat{y}_{i,t} \neq y_{i,t},\, \mathcal{K}_{i,t}\} ,
\end{aligned}$$

where we have introduced the short-hand $\mathcal{K}_{i,t} =$ "$\forall j \in \mathrm{ANC}(i) : V_{j,t} = 1$". By the same token, we have

$$\begin{aligned}
(9) &= \{\widehat{y}_{i,t} \neq V_{i,t}, y_{i,t} \neq V_{i,t}, \forall j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} = V_{j,t} = 1, \exists j \in \mathrm{ANC}(i) : y_{j,t} \neq V_{j,t}\} \\
&= \{\widehat{y}_{i,t} \neq V_{i,t}, y_{i,t} \neq V_{i,t}, \forall j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} = V_{j,t} = 1, \exists j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} \neq y_{j,t}\} \\
&\leq \{\exists j \in \mathrm{ANC}(i) : \widehat{y}_{j,t} \neq y_{j,t},\, \mathcal{K}_{i,t}\} \\
&\leq \sum_{j \in \mathrm{ANC}(i)} \{\widehat{y}_{j,t} \neq y_{j,t},\, \mathcal{K}_{i,t}\} \\
&\leq \sum_{j \in \mathrm{ANC}(i)} \{\widehat{y}_{j,t} \neq y_{j,t},\, \mathcal{K}_{j,t}\} ,
\end{aligned}$$

where the last inequality holds because $\mathcal{K}_{i,t}$ implies $\mathcal{K}_{j,t}$ for all $j \in \text{ANC}(i)$. Using our bounds for (8) and (9), and summing over i yields

$$\begin{aligned}
&\ell_H(\widehat{y}_t, V_t) - \ell_H(y_t, V_t) \\
&= \sum_{i=1}^{N} \Big(\{\widehat{y}_{i,t} \neq V_{i,t}, \forall j \in \text{ANC}(i) : \widehat{y}_{j,t} = V_{j,t}\} - \{y_{i,t} \neq V_{i,t}, \forall j \in \text{ANC}(i) : y_{j,t} = V_{j,t}\} \Big) \\
&\leq \sum_{i=1}^{N} \sum_{j \in \text{ANC}(i) \cup \{i\}} \{\widehat{y}_{j,t} \neq y_{j,t},\ \mathcal{K}_{j,t}\} \\
&= \sum_{i=1}^{N} \{\widehat{y}_{i,t} \neq y_{i,t},\ \mathcal{K}_{i,t}\} \sum_{j \in \text{SUB}(i)} 1 \\
&= \sum_{i=1}^{N} C_i \, \{\widehat{y}_{i,t} \neq y_{i,t},\ \mathcal{K}_{i,t}\} \ .
\end{aligned}$$

We then take expectations and sum over t:

$$\begin{aligned}
\sum_{t=1}^{T} \big(\mathbb{E}\ell_H(\widehat{y}_t, V_t) - \mathbb{E}\ell_H(y_t, V_t) \big) &\leq \sum_{t=1}^{T} \sum_{i=1}^{N} C_i \, \mathbb{P}(\widehat{y}_{i,t} \neq y_{i,t},\ \mathcal{K}_{i,t}) \\
&= \sum_{i=1}^{N} C_i \sum_{t=1}^{T} \mathbb{P}(\widehat{y}_{i,t} \neq y_{i,t},\ \mathcal{K}_{i,t}) \ . \qquad (10)
\end{aligned}$$

Equation (10) is a conveniently simple upper bound on the cumulative regret. This allows us to focus on bounding from above the one-node cumulative expectation $\sum_{t=1}^{T} \mathbb{P}(\widehat{y}_{i,t} \neq y_{i,t},\ \mathcal{K}_{i,t})$.

For brevity, in the rest of this proof we use the notations $\Delta_{i,t} = u_i^\top x_t$ (the target margin on x_t) and $\widehat{\Delta}_{i,t} = w_{i,t}^\top x_t$ (the algorithm margin on x_t). As we said earlier, our argument centers on proving that for any node i, $\widehat{\Delta}_{i,t}$ is an asymptotically unbiased estimator of $\Delta_{i,t}$, and then on using known large deviation techniques to obtain the stated bound. For this purpose, we need to study both the conditional bias and the conditional variance of $\widehat{\Delta}_{i,t}$.

Recall Figure 3. Since the sequence $x_1, x_2, \ldots$ is fixed, the multilabel vectors V_t are statistically independent. Also, for any $t = 1, 2, \ldots$ and for any node i with parent j, the child's labels $V_{i,i_1}, \ldots, V_{i,i_{Q(i,t-1)}}$ are independent when conditioned on the parent's labels $V_{j,1}, \ldots, V_{j,t-1}$. We use the notation

$$\mathbb{E}_{i,t} = \mathbb{E}[\cdot \mid V_{j,1}, \ldots, V_{j,t-1}] \ .$$

By definition of our parametric model (3) we have $\mathbb{E}_{i,t}[(V_{i,i_1}, \ldots, V_{i,i_{Q(i,t-1)}})^\top] = S_{i,Q(i,t-1)}^\top u_i$. Recalling the definition (1) of $w_{i,t}$, this implies (for conciseness we write Q instead of $Q(i,t-1)$)

$$\mathbb{E}_{i,t}[\widehat{\Delta}_{i,t}] = u_i^\top S_{i,Q} S_{i,Q}^\top (I + S_{i,Q} S_{i,Q}^\top + x_t x_t^\top)^{-1} x_t \ .$$

Note that

$$\Delta_{i,t} = \mathbb{E}_{i,t}[\widehat{\Delta}_{i,t}] + u_i^\top (I + x_t x_t^\top)(I + S_{i,Q} S_{i,Q}^\top + x_t x_t^\top)^{-1} x_t = \mathbb{E}_{i,t}[\widehat{\Delta}_{i,t}] + B_{i,t},$$

where $B_{i,t} = u_i^\top (I + x_t x_t^\top)(I + S_{i,Q} S_{i,Q}^\top + x_t x_t^\top)^{-1} x_t$ is the conditional bias of $w_{i,t}$. It is useful to introduce the short-hand notation

$$r_{i,t} = x_t^\top (I + S_{i,Q} S_{i,Q}^\top + x_t x_t^\top)^{-1} x_t \ .$$

Also, in order to stress the dependence[1] of $r_{i,t}$ on $Q = Q(i,t-1)$, we denote it by $r_{i,t,Q}$.

The conditional bias is bounded in the following lemma (proven in the appendix).

Lemma 7 *With the notation introduced so far, we have*

$$B_{i,t} \le \sqrt{r_{i,t,Q}} + |\Delta_{i,t}|\, r_{i,t,Q} \,.$$

As far as the conditional variance of $\widehat{\Delta}_{i,t}$ is concerned, from Figure 3 we see that

$$\widehat{\Delta}_{i,t} = \sum_{k=1}^{Q} V_{i,i_k} Z_{i,t,k} \,,$$

where

$$Z_{i,t}^\top = (Z_{i,t,1}, \ldots, Z_{i,t,Q}) = S_{i,Q}^\top \left(I + S_{i,Q} S_{i,Q}^\top + x_t x_t^\top \right)^{-1} x_t \,. \tag{11}$$

The next lemma (proven in the appendix) handles the conditional variance $\|Z_{i,t}\|^2$.

Lemma 8 *With the notation introduced so far, we have*

$$\|Z_{i,t}\|^2 \le r_{i,t,Q} \,.$$

Armed with these two lemmas, we proceed through our large deviation argument.

We can write

$$\begin{aligned}
&\{\hat{y}_{i,t} \neq y_{i,t}, \mathcal{K}_{i,t}\} \\
&\quad\le \left\{\widehat{\Delta}_{i,t}\,\Delta_{i,t} \le 0,\ \mathcal{K}_{i,t}\right\} \\
&\quad\le \left\{|\widehat{\Delta}_{i,t} - \Delta_{i,t}| \ge |\Delta_{i,t}|,\ \mathcal{K}_{i,t}\right\} \\
&\quad\le \left\{|\widehat{\Delta}_{i,t} + B_{i,t} - \Delta_{i,t}| \ge |\Delta_{i,t}| - |B_{i,t}|,\ \mathcal{K}_{i,t}\right\} \\
&\quad\le \left\{|\widehat{\Delta}_{i,t} + B_{i,t} - \Delta_{i,t}| \ge |\Delta_{i,t}|/2,\ \mathcal{K}_{i,t}\right\} + \{|B_{i,t}| \ge |\Delta_{i,t}|/2,\ \mathcal{K}_{i,t}\} \,.
\end{aligned} \tag{12}$$

We can further bound the second term of (12) by using Lemma 7. We obtain

$$\begin{aligned}
\left\{|B_{i,t}| \ge |\Delta_{i,t}|/2,\ \mathcal{K}_{i,t}\right\} &\le \{\sqrt{r_{i,t,Q}} + |\Delta_{i,t}|\, r_{i,t,Q} \ge |\Delta_{i,t}|/2,\ \mathcal{K}_{i,t}\} \\
&\le \left\{\left(r_{i,t,Q} \ge |\Delta_{i,t}|^2/16 \vee r_{i,t,Q} \ge 1/4\right),\ \mathcal{K}_{i,t}\right\} \\
&= \left\{r_{i,t,Q} \ge |\Delta_{i,t}|^2/16,\ \mathcal{K}_{i,t}\right\}
\end{aligned}$$

1. As it turns out, many of the quantities appearing in the present proof, including the bias term $B_{i,t}$ and the variance vector $Z_{i,t}$ defined later on, are algorithm-dependent, hence they do actually depend on $Q = Q(i,t-1)$. However, this dependence is made notationally explicit only for the quantity $r_{i,t} = r_{i,t,Q}$ since, we believe, this specific dependence is key to the proof.

the equality following from the fact that $|\Delta_{i,t}|^2/16 \leq 1/16 < 1/4$. We plug back into (12), take expectations, and sum over t. We have

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{T}\{\hat{y}_{i,t} \neq y_{i,t},\, \mathcal{K}_{i,t}\}\right] \\
&\leq \mathbb{E}\left[\sum_{t=1}^{T}\left(\left\{|\widehat{\Delta}_{i,t}+B_{i,t}-\Delta_{i,t}| \geq |\Delta_{i,t}|/2,\, \mathcal{K}_{i,t}\right\}+\left\{r_{i,t,Q} \geq |\Delta_{i,t}|^2/16,\, \mathcal{K}_{i,t}\right\}\right)\right] \\
&= \mathbb{E}\left[\sum_{t=1}^{T}\{\mathcal{K}_{i,t}\}\,\mathbb{E}_{i,t}\left\{|\widehat{\Delta}_{i,t}+B_{i,t}-\Delta_{i,t}| \geq |\Delta_{i,t}|/2\right\}\right] \qquad (13) \\
&\quad + \mathbb{E}\left[\sum_{t=1}^{T}\left\{r_{i,t,Q} \geq |\Delta_{i,t}|^2/16,\, \mathcal{K}_{i,t}\right\}\right], \qquad (14)
\end{aligned}$$

where in (13) we used the fact that $\mathcal{K}_{i,t}$ is determined given $V_{\mathrm{PAR}(i),1},\ldots,V_{\mathrm{PAR}(i),t-1}$.

We now bound the two expectations (13) and (14) separately. Let $j = \mathrm{PAR}(i)$. To bound the first expectation, we exploit the fact that $V_{i,i_1},\ldots,V_{i,i_Q}$ are independent under the law $P_{i,t} = \mathbb{P}\big(\cdot \mid V_{j,1},\ldots,V_{j,t-1}\big)$, and $Z_{i,t,1},\ldots,Z_{i,t,Q}$ defined in (11) are determined given $V_{j,1},\ldots,V_{j,t-1}$. Hence, we can apply Chernoff-Hoeffding inequality (Hoeffding, 1963) to the sum $\widehat{\Delta}_{i,t} = V_{i,i_1}Z_{i,t,1}+\ldots+V_{i,i_Q}Z_{i,t,Q}$ of independent random variables, where $\mathbb{E}_{i,t}[\widehat{\Delta}_{i,t}] = \Delta_{i,t} - B_{i,t}$ and $(V_{i,i_1}Z_{i,t,1})^2+\ldots+(V_{i,i_Q}Z_{i,t,Q})^2 \leq r_{i,t,Q}$ by Lemma 8. Recalling that $\Delta_i^2 = \min_{t=1,\ldots,T}\Delta_{i,t}^2$, we can write

$$\sum_{t=1}^{T}\{\mathcal{K}_{i,t}\}\mathbb{P}_{i,t}\left(|\widehat{\Delta}_{i,t}+B_{i,t}-\Delta_{i,t}| \geq |\Delta_{i,t}|/2\right) \leq 2\sum_{t=1}^{T}\{\mathcal{K}_{i,t}\}\exp\left(-\frac{\Delta_i^2}{8\,r_{i,t,Q}}\right).$$

This quantity can be further upper bounded using the following lemma (proven in the appendix).

Lemma 9 *Let α, M be positive constants. Then*

$$\max\left\{\sum_{t=1}^{n}e^{-\alpha/a_t} : a_1 \geq 0,\ldots,a_n \geq 0, \sum_{t=1}^{n}a_t = M\right\} \leq \frac{M}{e\alpha}.$$

If we let

$$M_i = \sum_{t=1}^{T}\{\mathcal{K}_{i,t}\}r_{i,t,Q} = \sum_{t:\{\mathcal{K}_{i,t}\}=1} r_{i,t,Q}$$

we immediately see that Lemma 9 implies

$$\sum_{t=1}^{T}\{\mathcal{K}_{i,t}\}\exp\left(-\frac{\Delta_i^2}{8\,r_{i,t,Q}}\right) = \sum_{t:\{\mathcal{K}_{i,t}\}=1}\exp\left(-\frac{\Delta_i^2}{8\,r_{i,t,Q}}\right) \leq \frac{8}{e\Delta_i^2}M_i.$$

Therefore,

$$(13) \leq \frac{16}{e\Delta_i^2}\,\mathbb{E}\,M_i.$$

To bound (14) we can argue as follows (note that, by definition, $r_{i,t,Q} \geq 0$, since it is the value of a quadratic form with a positive definite matrix):

$$\begin{aligned} M_i &= \sum_{t=1}^{T} \{\mathcal{K}_{i,t}\}\, r_{i,t,Q} \\ &= \sum_{t=1}^{T} \{r_{i,t,Q} \geq \Delta_i^2/16,\, \mathcal{K}_{i,t}\}\, r_{i,t,Q} + \sum_{t=1}^{T} \{r_{i,t,Q} < \Delta_i^2/16,\, \mathcal{K}_{i,t}\} r_{i,t,Q} \\ &\geq \sum_{t=1}^{T} \{r_{i,t,Q} \geq \Delta_i^2/16,\, \mathcal{K}_{i,t}\}\, \Delta_i^2/16 \,. \end{aligned}$$

Hence

$$(14) = \mathbb{E}\left[\sum_{t=1}^{T} \{r_{i,t,Q} \geq \Delta_i^2/16,\, \mathcal{K}_{i,t}\}\right] \leq \frac{16}{\Delta_i^2}\, \mathbb{E} M_i \,.$$

We have thus obtained the following bound

$$\sum_{t=1}^{T} \mathbb{P}(\hat{y}_{i,t} \neq y_{i,t},\, \mathcal{K}_{i,t}) \leq \frac{16(1+1/e)}{\Delta_i^2}\, \mathbb{E} M_i \,.$$

To conclude, we need to upper bound $\mathbb{E} M_i$. Observe that M_i is a sum only over time steps t such that $\{\mathcal{K}_{i,t}\} = 1$; i.e., over those t such that the weight vector $w_{i,t}$ gets actually updated. Therefore, since we would like to relate M_i to the spectral structure of the data correlation matrices $S_{i,Q(i,T)} S_{i,Q(i,T)}^\top$, we can proceed through the standard upper bounding argument (Azoury and Warmuth, 2001; Cesa-Bianchi et al., 2002) given below.

$$\begin{aligned} M_i &= \sum_{t=1}^{T} \{\mathcal{K}_{i,t}\} r_{i,t,Q} \\ &= \sum_{t=1}^{T} \left(1 - \frac{\det(I + S_{i,Q(i,t-1)} S_{i,Q(i,t-1)}^\top)}{\det(I + S_{i,Q(i,t)} S_{i,Q(i,t)}^\top)}\right) \\ &\quad \text{(using Lemma 2, part 1, in Lai and Wei, 1982)} \\ &\leq \sum_{t=1}^{T} \log \frac{\det(I + S_{i,Q(i,t)} S_{i,Q(i,t)}^\top)}{\det(I + S_{i,Q(i,t-1)} S_{i,Q(i,t-1)}^\top)} \quad (\text{since } 1 - x \leq -\log x \text{ for all } x > 0) \\ &= \log \frac{\det(I + S_{i,Q(i,T)} S_{i,Q(i,T)}^\top)}{\det(I)} \\ &= \sum_{j=1}^{d} \log(1 + \lambda_{i,j}) \,. \end{aligned}$$

Putting together as in (10) concludes the proof. □

Our analysis of Theorem 2 is similar in spirit to the work of Lai et al. (1979) on least-squares regression. In particular, they also assume the sequence $x_1, x_2, \ldots$ be arbitrary while the real-valued labels y_t are defined as $y_t = u^\top x_t + \varepsilon_t$, where ε_t are i.i.d. random variables with finite variance.

A regret bound similar to the one established by Theorem 2 can be proven for the zero-one loss using the fact that this loss can be crudely upper bounded by the H-loss (with all cost coefficients set to 1). Indeed, a more direct (and sharper) analysis could be performed for the zero-one

loss, following the same lines as the proof of Theorem 2. As far as the symmetric difference loss ℓ_Δ is concerned, a regret analysis might be obtained through a method we developed in earlier work (Cesa-Bianchi et al., 2004). As a matter of fact, the analysis by Cesa-Bianchi et al. (2004) rests on several side assumptions about the way data $x_1, \ldots, x_T$ are generated. We have been unable to apply the theoretical arguments employed in the present paper to ℓ_Δ. In any case, since these two loss functions are unable to capture the hierarchical nature of our classification problem, we believe the resulting bounds are less relevant to this paper.

8. Experimental Results

We tested the empirical performance of our on-line algorithm on data sets extracted from two popular corpora of free-text documents. The first data set consists of the first (in chronological order) 100,000 newswire stories from the Reuters Corpus Volume 1 (Reuters, 2000). The associated taxonomy of labels, which are the document topics, contains 101 nodes organized in a forest of 4 trees. The forest is shallow: the longest path has length 3 and the distribution of nodes, sorted by increasing path length, is $\{0.04, 0.53, 0.42, 0.01\}$. The average number of paths in the multilabel of an instance is 1.5. For this data set we used the bag-of-words vectorization performed by Xerox Research Center Europe within the EC project KerMIT (see Cesa-Bianchi et al., 2003, for details). The 100,000 documents were divided into 5 equally sized groups of chronologically consecutive documents. We then used each adjacent pair of groups as training and test set for an experiment (here the fifth and first group are considered adjacent), and then averaged the test set performance over the 5 experiments.

The second data set includes the documents classified in the nodes of the subtree rooted in "Quality of Health Care" (MeSH code N05.715) of the OHSUMED corpus of medical abstracts (Hersh, 1994). Since OHSUMED is not quite a tree but a directed acyclic graph, and since the H-loss is defined for trees only, we removed from this OHSUMED fragment the few nodes that did not have a unique path to the root. This produced a hierarchy with 94 classes and a data set with 55,503 documents. The choice of this specific subtree was motivated by its structure only; in particular: the subtree depth is 4, the distribution of nodes (sorted by increasing path length) is $\{0.26, 0.37, 0.22, 0.12, 0.03\}$, and there is a reasonable number of partial and multiple path multilabels (the average number of paths per instance is 1.53). The vectorization of the documents was carried out similarly to RCV1. After tokenization, we removed all stopwords and also those words that did not occur at least 3 times in the corpus. Then, we vectorized the documents using the BOW library (McCallum, 2004) with a $\log(1+\mathrm{TF})\log(\mathrm{IDF})$ encoding. We ran 5 experiments by randomly splitting the corpus in a training set of 40,000 documents and a test set of 15,503 documents. Test set performances are averages over these 5 experiments. In the training set we kept more documents than in the RCV1 splits since the OHSUMED corpus turned out to be a harder classification problem than RCV1. In both data sets instances have been normalized to unit length.

Since the space complexity of H-RLS grows linearly with training time, due to the need of storing each training instance in the matrices $S_{i,t}$ —see (1), we had to make some modifications to the algorithm in order to be able to carry out experiments on data sets of this size. For this purpose, we have developed SH-RLS, a space-efficient variant of H-RLS that we used in all of our experiments.

The performance of SH-RLS is compared against five baseline algorithms: a flat and a hierarchical version of the Perceptron algorithm (Novikov, 1962; Rosenblatt, 1958), a flat and a hierarchical

version of Vapnik's support vector machine (see, e.g., Vapnik, 1998; Schölkopf and Smola, 2002), and a flat version of SH-RLS. Note that support vector machines are not trained incrementally; we include them in our pool of baseline algorithms to show that on-line learners, processing each training example only once, can have a performance level close to that of batch learners.

Note also that, unlike our theoretical analysis based on cumulative regret, in the experiments we distinguish a training phase, where the hierarchical classifiers are built, and a test phase, where the performance of the hierarchical classifiers obtained in the training phase is measured on fresh data. This allows us to use a single measure, the test error, to compare both batch and incremental learners.

The first algorithm we consider, H-PERC, is a simple hierarchical version of the Perceptron. Its functioning differs from H-RLS described in Figure 3 only in the way weights are updated. In particular, H-PERC learns a hierarchical classifier by training a linear-threshold classifier at each node via the Perceptron algorithm. At the beginning, the weight vector of each node classifier is set to the zero vector, $w_{i,1} = (0,\ldots,0)$ for $i = 1,\ldots,N$. Upon receiving an example (x_t, v_t), H-PERC considers for an update only those classifiers sitting at nodes i satisfying either $i \in \text{ROOT}(G)$ or $v_{\text{PAR}(i),t} = 1$. If $\{w_{i,t}^\top x_t \geq 0\} \neq v_{i,t}$ for such a node i, then the weight vector $w_{i,t}$ is updated using the Perceptron rule $w_{i,t+1} = w_{i,t} + v_{i,t} x_t$; on the other hand, if $\{w_{i,t}^\top x_t \geq 0\} = v_{i,t}$, then $w_{i,t+1} = w_{i,t}$ (no update takes place at node i).

During the test phase, H-PERC computes the multilabel $\widehat{y} = (\widehat{y}_1,\ldots,\widehat{y}_N)$ of a test instance x using the same top-down process described in Figure 3,

$$\widehat{y}_i = \begin{cases} \{w_i^\top x \geq 0\} & \text{if } i \text{ is a root node,} \\ \{w_i^\top x \geq 0\} & \text{if } i \text{ is not a root node and } \widehat{y}_j = 1 \text{ for } j = \text{PAR}(i), \\ 0 & \text{if } i \text{ is not a root node and } \widehat{y}_j = 0 \text{ for } j = \text{PAR}(i). \end{cases} \tag{15}$$

The second incremental algorithm considered is SH-RLS, our sparse variant of H-RLS. The two algorithms, H-RLS and SH-RLS operate in the same way (see Figure 3) with the only difference that SH-RLS performs fewer updates in the training phase. In particular, given a training example (x_t, v_t), both algorithms consider for an update only those classifiers sitting at nodes i satisfying either $i \in \text{ROOT}(G)$ or $v_{\text{PAR}(i),t} = 1$. However, whereas H-RLS would update the weight $w_{i,t}$ of all such nodes i, SH-RLS also requires the margin condition $|w_{i,t}^\top x_t| \leq \sqrt{(5 \ln t)/N_{i,t}}$, where $N_{i,t}$ is the number of instances stored at node i up to time $t-1$. The choice of the margin threshold $\sqrt{(5 \ln t)/N_{i,t}}$ is motivated by Cesa-Bianchi et al. (2003) via a large deviation analysis.

We also tested a hierarchical version of SVM (denoted by H-SVM) in which each node is an SVM classifier trained using a batch version of our hierarchical learning protocol. More precisely, each node i was trained only on those examples (x_t, v_t) such that $v_{\text{PAR}(i),t} = 1$. The resulting set of linear-threshold functions was then evaluated on the test set using the hierarchical classification scheme (15). We tried both the C and ν parametrizations (Schölkopf et al., 2000) for SVM and found the setting $C = 1$ to work best[2] for our data (recall that all instances x_t are normalized, $\|x_t\| = 1$).

We finally tested the "flat" variants of H-PERC, SH-RLS and H-SVM, denoted by PERC, S-RLS and SVM, respectively. In these variants, each node is trained and evaluated independently of the others, disregarding all taxonomical information. All SVM experiments were carried out using the libSVM implementation (Chang and Lin, 2004) and all the algorithms ran with a linear kernel. The

2. It should be emphasized that this tuning of C was actually chosen in hindsight across the interval [0.1,10] with no cross-validation.

RCV1			
Algorithm	zero-one loss	uniform H-loss	Δ-loss
PERC	$0.702(\pm 0.045)$	$1.196(\pm 0.127)$	$1.695(\pm 0.182)$
H-PERC	$0.655(\pm 0.040)$	$1.224(\pm 0.114)$	$1.861(\pm 0.172)$
S-RLS	$0.559(\pm 0.005)$	$0.981(\pm 0.020)$	$1.413(\pm 0.033)$
SH-RLS	$\mathbf{0.456}(\pm\mathbf{0.010})$	$\mathbf{0.743}(\pm\mathbf{0.026})$	$\mathbf{1.086}(\pm\mathbf{0.036})$
SVM	$0.482(\pm 0.009)$	$0.790(\pm 0.023)$	$1.173(\pm 0.051)$
H-SVM	$0.440(\pm 0.008)$	$0.712(\pm 0.021)$	$1.050(\pm 0.027)$
OHSUMED			
Algorithm	zero-one loss	uniform H-loss	Δ-loss
PERC	$0.899(\pm 0.024)$	$1.938(\pm 0.219)$	$2.639(\pm 0.226)$
H-PERC	$0.846(\pm 0.024)$	$1.560(\pm 0.155)$	$2.528(\pm 0.251)$
S-RLS	$0.873(\pm 0.004)$	$1.814(\pm 0.024)$	$2.627(\pm 0.027)$
SH-RLS	$\mathbf{0.769}(\pm\mathbf{0.004})$	$\mathbf{1.200}(\pm\mathbf{0.007})$	$\mathbf{1.957}(\pm\mathbf{0.011})$
SVM	$0.784(\pm 0.003)$	$1.206(\pm 0.003)$	$1.872(\pm 0.005)$
H-SVM	$0.759(\pm 0.002)$	$1.170(\pm 0.005)$	$1.910(\pm 0.007)$

Table 1: Experimental results on two hierarchical text classification tasks under various loss functions. We report average test errors along with standard deviations (in parentheses). In bold are the best performance figures among the incremental algorithms (all incremental algorithms were run for one epoch over the training data).

performance of these algorithms was evaluated against three different loss measures (see Table 1). The first two losses are the zero-one loss and the H-loss with cost coefficients set to 1 (denoted by uniform H-loss in Table 1). The third loss is the symmetric difference loss (Δ-loss in Table 1).

A few remarks on Table 1 are in order at this point. As expected, H-SVM performs best, but the good performance of SVM (flat support vector machine) is surprising. As for the incremental algorithms, SH-RLS performs better than its flat variant SH-RLS, and far better than both H-PERC and PERC. In addition, and perhaps surprisingly, after a single epoch of training SH-RLS performs generally better than SVM and comes reasonably close to the performance of H-SVM. Finally, note that the running times of both S-RLS and SH-RLS scale quadratically in the number of stored instances, whereas the running time of Perceptrons scales only linearly. Thus, as usual, the performance benefit has to be traded-off against computational cost.

To give an idea of how flat and hierarchical algorithms compare in terms of running times, we mention that hierarchical algorithms turned out to be roughly four times faster than the corresponding flat algorithms running on the same data sets.

The (uniform) H-loss does not provide any information on the distribution of mistakes across the different hierarchy levels. Therefore, we counted the "H-loss mistakes" made at each level, distinguishing between false positive (FP) and false negative (FN) mistakes. Fix an example (x, v) and let $\widehat{y}$ be the guessed multilabel. Then node i makes an H-loss mistake on (x, v) if

$$\widehat{y}_i \neq v_i \wedge \widehat{y}_j = v_j = 1,\ j \in \text{ANC}(i)\ .$$

RCV1				
Depth		H-PERC	SH-RLS	H-SVM
0	FP	4144(±2431)	1449(±79)	1769(±163)
	FN	2690(±851)	2436(±112)	2513(±148)
1	FP	6769(±2509)	1361(±108)	1317(±81)
	FN	7961(±838)	8135(±476)	7260(±450)
2	FP	1161(±261)	413(±32)	380(±28)
	FN	1513(±833)	937(±51)	624(±23)
3	FP	161(±314)	14(±16)	20(±26)
	FN	88(±44)	115(±31)	94(±24)

OHSUMED				
Depth		H-PERC	SH-RLS	H-SVM
0	FP	7916(±2638)	3192(±88)	3062(±60)
	FN	12639(±1418)	12888(±64)	12587(±49)
1	FP	1816(±730)	828(±14)	839(±11)
	FN	1606(±373)	1594(±33)	1542(±25)
2	FP	88(±20)	30(±6)	37(±7)
	FN	86(±31)	54(±4)	55(±2)
3	FP	10(±5)	2(±1)	3(±1)
	FN	16(±11)	13(±3)	14(±1)
4	FP	3(±2)	1(±1)	4(±1)
	FN	5(±6)	1(±1)	2(±1)

Table 2: Distribution across the hierarchy levels of false positive (FP) and false negative (FN) H-loss mistakes on the two hierarchical text classification tasks RCV1 and OHSUMED. We report the average number of mistakes at each level of the hierarchy trees with standard deviation in parentheses (recall that we made 5 experiments on different splits of the two data set).

Thus, node i makes a false positive mistake if

$$\widehat{y}_i = 1 \wedge v_i = 0 \wedge \widehat{y}_j = v_j = 1,\ j \in \text{ANC}(i)$$

and makes a false negative mistake if

$$\widehat{y}_i = 0 \wedge v_i = 1 \wedge \widehat{y}_j = v_j = 1,\ j \in \text{ANC}(i)\ .$$

Table 2 shows the H-loss mistake distribution for RCV1 and OHSUMED over hierarchy levels.

The average values contained in Table 2 are also plotted in Figure 4. A quick visual comparison reveals the close similarity between the distributions obtained by SH-RLS and H-SVM, whereas the behavior of H-PERC looks quite different.

9. Conclusions, Ongoing Research, and Open Problems

We have introduced H-RLS, a new on-line algorithm for hierarchical classification that maintains and updates regularized least-squares estimators on the nodes of a taxonomy. The linear-threshold classifications, obtained from the estimators, are combined to produce a single hierarchical multilabel through a simple top-down evaluation model.

Our algorithm is suitable for learning multilabels that include multiple and/or partial paths on the taxonomy. To properly evaluate hierarchical classifiers in this framework we have defined the H-loss, a new hierarchical loss function, with cost coefficients possibly associated to each taxonomy node—see Remark 5.

Our main theoretical result states that, on any sequence of instances, the cumulative H-loss of H-RLS is never much bigger than the cumulative H-loss of a reference classifier tuned with the parameters of the stochastic process generating the multilabels for the given sequence of instances. Our theoretical findings are complemented by experiments on the hierarchical classification of textual data, in which we compare the performance of a sparsified variant of H-RLS to that of standard batch and incremental learners, such as simple hierarchical versions of the Perceptron algorithm and the SVM. The experiments show that one epoch of training of our algorithm is enough to achieve a performance close to that of the hierarchical SVM.

Our investigation leaves a number of open questions. The first open question is the derivation of a hierarchical algorithm especially designed to minimize the H-loss. We are currently exploring efficient ways to approximate the Bayes optimal classifier for the H-loss, given our data model. Since such optimal classifier turns out to be remarkably different from the hierarchical classifiers produced by H-RLS, a related theoretical question is to prove any reasonable bound on the regret with respect to the Bayes optimal classifier.

Additional open problems concern the data model. First, it would be useful to modify the label-generating model to introduce dependencies among the children's labels. This could allow a better fitting of data sets when the rate of multiple paths in multilabels is limited. Second, further investigation, both of empirical and theoretical nature, might be devoted to the issue of using regularized logistic regressors at each node.

Acknowledgments

The authors would like to thank Michael Collins for his timely editorial work, as well as the anonymous reviewers, whose comments and suggestions greatly improved the presentation of this paper.

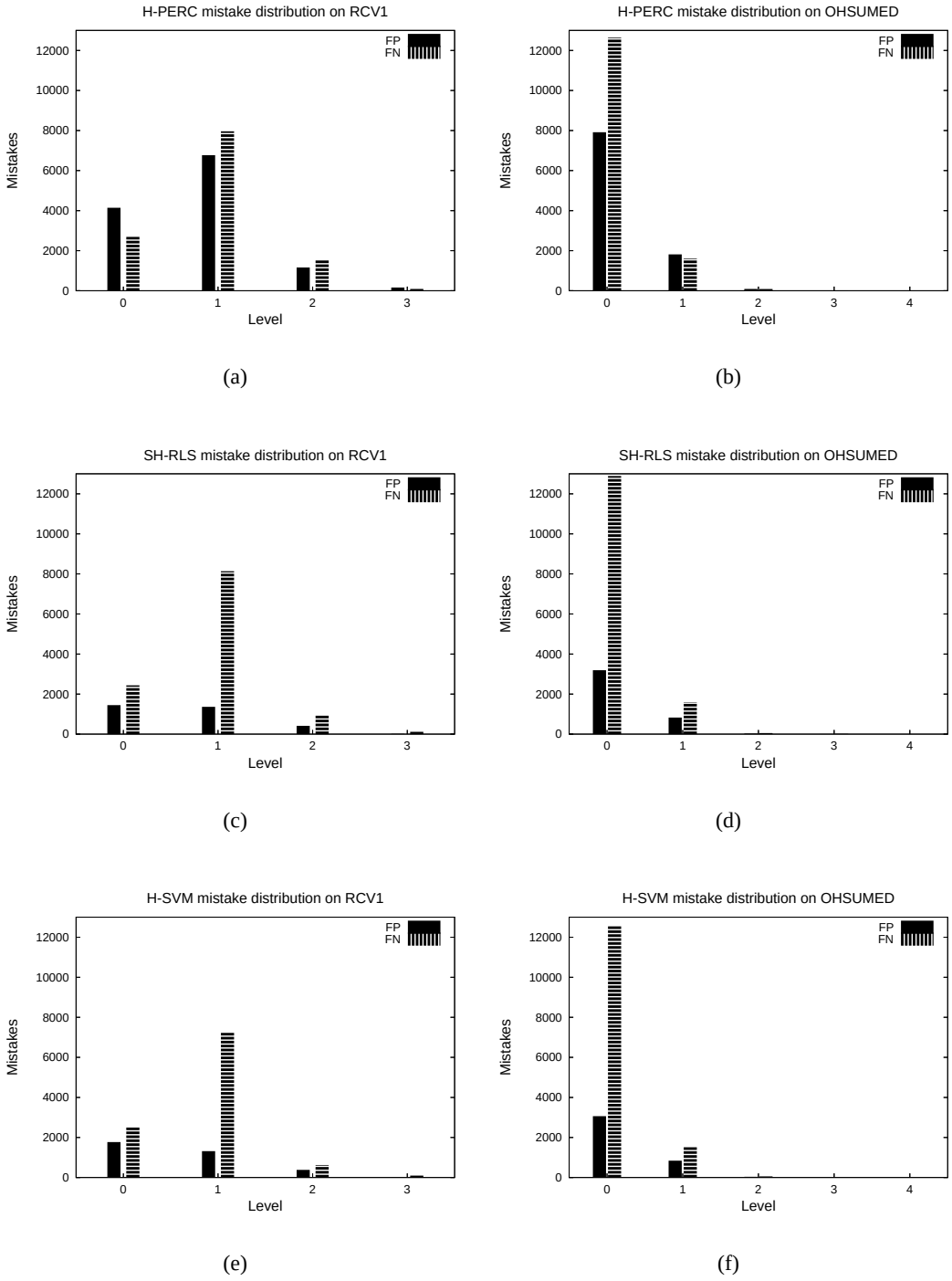


Figure 4: Plot of the average values contained in Table 2 for the H-loss mistake distribution over hierarchy levels.

This work was supported in part by the IST Programme of the European Community under the PASCAL Network of Excellence IST-2002-506778. This publication only reflects the authors' views.

Appendix A

This appendix contains the proofs of Lemmas 7, 8, and 9 mentioned in the main text. Throughout this appendix A denotes the positive definite matrix $I + S_{i,Q} S_{i,Q}^\top$, while r denotes the quadratic form $x_t^\top (A + x_t x_t^\top)^{-1} x_t$.

Proof of Lemma 7

We have

$$\begin{aligned} B_{i,t} &= u_i^\top (I + x_t x_t^\top)(A + x_t x_t^\top)^{-1} x_t \\ &= u_i^\top (A + x_t x_t^\top)^{-1} x_t + \Delta_{i,t}\, r \\ &\leq \sqrt{x_t^\top (A + x_t x_t^\top)^{-2} x_t} + |\Delta_{i,t}|\, r \\ &\leq \sqrt{r} + |\Delta_{i,t}|\, r \end{aligned}$$

where the first inequality follows from $u_i^\top z \leq \max_{\|u_i\|=1} u_i^\top z = \|z\|$, with $z = (A + x_t x_t^\top)^{-1} x_t$, and the second inequality follows from $x^\top (A + xx^\top)^{-2} x \leq x^\top (A + xx^\top)^{-1} x$, holding for any x and for any positive definite matrix A whose eigenvalues are not smaller than 1 (notice that this condition makes $(A + xx^\top)^{-1} - (A + xx^\top)^{-2}$ a positive semidefinite matrix). □

Proof of Lemma 8

Setting for brevity $H = S_{i,Q}^\top A^{-1} x_t$ and $a = x_t^\top A^{-1} x_t$ we can write

$$\begin{aligned} \|Z_{i,t}\|^2 &= x_t^\top \left(A + x_t x_t^\top\right)^{-1} S_{i,Q} S_{i,Q}^\top \left(A + x_t x_t^\top\right)^{-1} x_t \\ &= x_t^\top \left(A^{-1} - \frac{A^{-1} x_t x_t^\top A^{-1}}{1 + x_t^\top A^{-1} x_t}\right) S_{i,Q} S_{i,Q}^\top \left(A^{-1} - \frac{A^{-1} x_t x_t^\top A^{-1}}{1 + x_t^\top A^{-1} x_t}\right) x_t \\ &\quad \text{(by the Sherman-Morrison formula—e.g., Horn and Johnson, 1985, chap. 0)} \\ &= H^\top H - \frac{a}{1+a} H^\top H - \frac{a}{1+a} H^\top H + \frac{a^2}{(1+a)^2} H^\top H \\ &= \frac{H^\top H}{(1+a)^2} \\ &= \frac{x_t^\top A^{-1} S_{i,Q} S_{i,Q}^\top A^{-1} x_t}{(1+a)^2} \\ &= \frac{x_t^\top A^{-1/2} A^{-1/2} S_{i,Q} S_{i,Q}^\top A^{-1/2} A^{-1/2} x_t}{(1+a)^2} \\ &\leq \frac{\left\|A^{-1/2} x_t\right\| \left\|A^{-1/2} S_{i,Q} S_{i,Q}^\top A^{-1/2}\right\| \left\|x_t^\top A^{-1/2}\right\|}{(1+a)^2} \\ &= \frac{a}{(1+a)^2} \left\|A^{-1/2} S_{i,Q} S_{i,Q}^\top A^{-1/2}\right\|, \end{aligned} \tag{16}$$

where $\left\|A^{-1/2}S_{i,Q}S_{i,Q}^{\top}A^{-1/2}\right\|$ is the spectral norm of matrix $A^{-1/2}S_{i,Q}S_{i,Q}^{\top}A^{-1/2}$.

We continue by bounding the two factors in (16). Observe that

$$\frac{a}{(1+a)^2} \leq \frac{a}{1+a} = r$$

where the equality derives again from the Sherman-Morrison formula. As far as the second factor is concerned, we just note that the two matrices $A^{-1/2}$ and $S_{i,Q}S_{i,Q}^{\top}$ have the same eigenvectors. Furthermore, if λ_j is an eigenvalue of $S_{i,Q}S_{i,Q}^{\top}$, then $1/\sqrt{1+\lambda_j}$ is an eigenvalue of $A^{-1/2}$. Therefore

$$\left\|A^{-1/2}S_{i,Q}S_{i,Q}^{\top}A^{-1/2}\right\| = \max_j \frac{1}{\sqrt{1+\lambda_j}} \times \lambda_j \times \frac{1}{\sqrt{1+\lambda_j}} \leq 1\,.$$

Substituting into (16) yields $\|Z_{i,t}\|^2 \leq r$, as desired. □

Proof of Lemma 9

From a simple Kuhn-Tucker analysis[3] it follows that if a_t is larger than 0 at the maximum, then a_t takes some constant value $\beta > 0$ (independent of t). Hence the maximizing vector $(a_1, a_2, \ldots, a_n)$ has components which can take only two possible values: $a_t = 0$ or $a_t = \beta$. Let us denote by N^+ the number of $t : a_t = \beta$. At the maximum we can write

$$M = \sum_{t=1}^{n} a_t = \sum_{t\,:\,a_t=\beta} a_t + \sum_{t\,:\,a_t\to 0^+} a_t = \beta N^+$$

i.e., $\beta = M/N^+$. Hence, at the maximum

$$\begin{aligned}
\sum_{t=1}^{n} e^{-\alpha/a_t} &= \sum_{t\,:\,a_t=\beta} e^{-\alpha/a_t} + \sum_{t\,:\,a_t=0^+} e^{-\alpha/a_t} \\
&= \sum_{t\,:\,a_t=\beta} e^{-\alpha/\beta} \\
&= N^+ e^{-\alpha/\beta} \\
&= N^+ e^{-\alpha N^+/M}\,.
\end{aligned}$$

Since N^+ is not determined by this argument, we can write

$$\max\left\{\sum_{t=1}^{n} e^{-\alpha/a_t} : a_1 \geq 0, \ldots, a_n \geq 0, \sum_{t=1}^{n} a_t = M\right\} \leq \max_{x\geq 0} x e^{-\alpha x/M} = \frac{M}{e\alpha}$$

thereby concluding the proof. □

3. The function $f(a) = e^{-\alpha/a}$ is not defined when $a = 0$. However, it is clearly possible to extend f by defining $f(0) = 0$, preserving (one-sided) differentiability.

References

K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential familiy of distributions. *Machine Learning*, 43(3):211–246, 2001.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM Journal of Computing.*, 43(3):640–668, 2005.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. Learning probabilistic linear-threshold classifiers via selective sampling. In *Proceedings of the 16th Annual Conference on Computational Learning Theory*, pages 373–386. LNAI 2777, Springer, 2003.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. Regret bounds for hierarchical classification with linear-threshold functions. In *Proceedings of the 17th Annual Conference on Computational Learning Theory*, pages 93–108. LNAI 3120, Springer, 2004.

C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines, 2004.
Available electronically at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

O. Dekel, J. Keshet, and Y. Singer. An efficient online algorithm for hierarchical phoneme classification. In *Proceedings of the 1st International Workshop on Machine Learning for Multimodal Interaction*, pages 146-158. Springer LNAI 3361, 2005.

O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *Proceedings of the 21st International Conference on Machine Learning*. Omnipress, 2004.

S. T. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263. ACM Press, 2000.

M. Granitzer. *Hierarchical Text Classification using Methods from Machine Learning*. PhD thesis, Graz University of Technology, 2003.

W. R. Hersh. The OHSUMED test collection, 1994.
Available electronically at `http://medir.ohsu.edu/pub/ohsumed/`.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

T. Hofmann, L. Cai, and M. Ciaramita. Learning with taxonomies: classifying documents and words. In *NIPS 2003: Workshop on syntax, semantics, and statistics*, 2003.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning*, pages 170–178, 1997.

T. L. Lai, H. Robbins, and C. Z. Wei. Strong consistency of least squares estimates in multiple regression. *Proceedings of the National Academy of Sciences USA*, 75(7):3034–3036, 1979.

T. L. Lai and C. Z. Wei. Least squares estimates in stochastic regression models with applications to identification and control of dynamic systems. *The Annals of Statistics*, 10(1):154–166, 1982.

A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 2004.
Available electronically at `http://www-2.cs.cmu.edu/~mccallum/bow/`.

A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the 15th International Conference on Machine Learning*, pages 359–367. Morgan Kaufmann Publishers, 1998.

D. Mladenic. Turning yahoo into an automatic web-page classifier. In *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 473–474, 1998.

A. B. J. Novikov. On convergence proofs on Perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata, vol. XII*, pages 615–622, 1962.

Reuters. Reuters corpus volume 1, 2000. Available electronically at
`http://about.reuters.com/researchandstandards/corpus/`.

R. Rifkin, G. Yeo, and T. Poggio. Regularized least squares classification. *Advances in Learning Theory: Methods, Model and Applications. NATO Science Series III: Computer and Systems Sciences*, 190:131–153, 2003.

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

M. E. Ruiz and P. Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5(1):87–118, 2002.

B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

A. Sun and E. P. Lim. Hierarchical text classification and evaluation. In *Proceedings of the 2001 International Conference on Data Mining*, pages 521–528. IEEE Press, 2001.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

V. Vovk. Competitive on-line statistics. *International Statistical Review*, 69:213–248, 2001.

Journal of Machine Learning Research 7 (2006) 55–83 Submitted 4/05; Published 1/06

On the Complexity of Learning Lexicographic Strategies

Michael Schmitt MSCHMITTM@GOOGLEMAIL.COM
Ludwig-Marum-Gymnasium
Schlossgartenstraße 11
76327 Pfinztal, Germany

Laura Martignon MARTIGNON@PH-LUDWIGSBURG.DE
Institut für Mathematik und Informatik
Pädagogische Hochschule Ludwigsburg
Reuteallee 46
71634 Ludwigsburg, Germany

Editor: Dana Ron

Abstract

Fast and frugal heuristics are well studied models of bounded rationality. Psychological research has proposed the take-the-best heuristic as a successful strategy in decision making with limited resources. Take-the-best searches for a sufficiently good ordering of cues (or features) in a task where objects are to be compared lexicographically. We investigate the computational complexity of finding optimal cue permutations for lexicographic strategies and prove that the problem is NP-complete. It follows that no efficient (that is, polynomial-time) algorithm computes optimal solutions, unless P = NP. We further analyze the complexity of approximating optimal cue permutations for lexicographic strategies. We show that there is no efficient algorithm that approximates the optimum to within any constant factor, unless P = NP.

The results have implications for the complexity of learning lexicographic strategies from examples. They show that learning them in polynomial time within the model of agnostic probably approximately correct (PAC) learning is impossible, unless RP = NP. We further consider greedy approaches for building lexicographic strategies and determine upper and lower bounds for the performance ratio of simple algorithms. Moreover, we present a greedy algorithm that performs provably better than take-the-best. Tight bounds on the sample complexity for learning lexicographic strategies are also given in this article.

Keywords: bounded rationality, fast and frugal heuristic, PAC learning, NP-completeness, hardness of approximation, greedy method

1. Introduction

In many circumstances the human mind has to make decisions when time is scarce and knowledge is limited. Extensive reflections backed by deep reasoning are impossible in these situations. Cognitive psychology categorizes human judgments made under such constraints as being boundedly rational if they are "satisficing" (Simon, 1982) or, more generally, if they do not fall too far behind the rational standards. The modeling of bounded rationality has been considered essential for artificial intelligence. Russell and Wefald (1991), defining artificial intelligence as the problem of designing systems that "do the right thing", argue that intelligence seems linked with doing as well as possible given what resources one has.

A principal family of models for human reasoning that are studied within the context of bounded rationality are the probabilistic mental models proposed by Gigerenzer et al. (1991). To these belongs a kind of simple algorithms termed "fast and frugal heuristics" that were the topic of major research projects in psychology (Gigerenzer and Goldstein, 1996; Gigerenzer et al., 1999). Great efforts have been put into testing these heuristics by empirical means in experiments with human subjects on the one hand (Bröder, 2000; Bröder and Schiffer, 2003; Lee and Cummins, 2004; Newell and Shanks, 2003; Newell et al., 2003; Slegers et al., 2000) or in simulations on computers on the other (Bröder, 2002; Bullock and Todd, 1999; Hogarth and Karelaia, 2003; Nellen, 2003; Todd and Dieckmann, 2005). (See also the discussion and controversies documented in the open peer commentaries on Todd and Gigerenzer, 2000.) To a lesser extent, theoretical studies have been undertaken with analytical methods (Bröder, 2002; Martignon and Hoffrage, 1999, 2002; Martignon and Schmitt, 1999).

1.1 Take the Best

Among the fast and frugal heuristics there is an algorithm called "take-the-best"[1] (TTB) that during recent years has become one of the workhorses of research into models of bounded rationality. This algorithm is considered a process model for human judgments based on one-reason decision making. Which of the two cities has a larger population: (a) Düsseldorf, (b) Hamburg? This is the task originally studied by Gigerenzer and Goldstein (1996) where German cities with a population of more than 100,000 inhabitants have to be compared. The available information on each city consists of the values of nine binary cues, or attributes, indicating presence or absence of a feature. The cues being used are, for instance, whether the city is a state capital, whether it is indicated on car license plates by a single letter, or whether it has a soccer team in the national league.

The judgment which city is larger is made on the basis of the two binary vectors, or cue profiles, representing the two cities. TTB compares the cues one after the other and uses the first cue that discriminates as the one reason to yield the final decision. In other words, TTB performs a lexicographic strategy of comparison. For instance, if one city has a university and the other does not it would infer that the first city is larger than the second. If the cue values of both cities are equal, the algorithm passes on to the next cue.

TTB examines the cues in a certain order. Gigerenzer and Goldstein (1996) introduced ecological validity as a numerical measure for ranking the cues. (See Martignon and Hoffrage, 2002, for further criteria to order cues.) The validity of a cue is a real number in the interval $[0,1]$ that is computed in terms of the known outcomes of paired comparisons. It is defined as the number of pairs the cue discriminates correctly (i.e., where it makes a correct inference) divided by the number of pairs it discriminates (i.e., where it makes an inference, be it right or wrong). TTB always chooses a cue with the highest validity, that is, it "takes the best" among those cues not yet considered. Table 1 gives an example showing cue profiles and validities for three cities. The data are extracted from the appendix of Gigerenzer and Goldstein (1996). The ordering defined by the population size of the cities is given by

$$\{\langle \text{Düsseldorf}, \text{Essen} \rangle, \langle \text{Düsseldorf}, \text{Hamburg} \rangle, \langle \text{Essen}, \text{Hamburg} \rangle\},$$

1. "Take-the-best" is a shortening of "take the best, ignore the rest" (Gigerenzer and Goldstein, 1996).

	Soccer Team	State Capital	License Plate
Hamburg	1	1	0
Essen	0	0	1
Düsseldorf	0	1	1
Validity	1	$1/2$	0

Table 1: Part of the German cities task of Gigerenzer and Goldstein (1996). Shown are cue profiles and validities. Validities are computed from the cues of the three cities as given here. The original data has different validities but yields the same ranking for the cues. The meaning of the cues and the way how to calculate validities are explained in the text.

where a pair $\langle a, b\rangle$ indicates that a has less inhabitants than b. As an example for calculating the validity, the state-capital cue distinguishes the first and the third pair but is correct only on the latter. Hence, its validity has value $1/2$.

The order in which the cues are ranked is crucial for success or failure of TTB. In the example of Düsseldorf and Hamburg, the car-license-plate cue would yield that Düsseldorf (represented by the letter "D") is larger than Hamburg (represented by the two letters "HH"), whereas the soccer-team cue would favor Hamburg, which is correct. Thus, how successful a lexicographic strategy is in a comparison task consisting of a partial ordering of cue profiles depends on how well the cue ranking minimizes the number of incorrect comparisons. Specifically, the accuracy of TTB relies on the degree of optimality achieved by the ranking according to decreasing cue validities. For TTB and the German cities task, computer simulations have shown that TTB discriminates at least as accurate as other models (Gigerenzer and Goldstein, 1996; Gigerenzer et al., 1999; Todd and Dieckmann, 2005). TTB made as many correct inferences as standard algorithms proposed by cognitive psychology and even outperformed some of them.[2]

1.2 Accuracy and Complexity

Partial results concerning the accuracy of TTB compared to the accuracy of other strategies have been obtained analytically by Martignon and Hoffrage (2002). The intention of this article is to subject the problem of finding optimal cue orderings to a rigorous theoretical analysis. A conceivable approach would be to reveal conditions under which TTB performs better or worse. However, the analysis of TTB per se is not a major topic of this work. Instead, we take a different and more general road by employing methods from the theory of computational complexity (Garey and Johnson, 1979).

Obviously, TTB is an algorithm that runs in polynomial time. Given a list of ordered pairs, it computes all cue validities in a number of computing steps that is linear in the size of the list,

2. Gigerenzer and Goldstein (1996) introduced TTB with an additional feature, the recognition principle. The recognition cue indicates whether the city is recognized or not. A city that is recognized is preferred to an unrecognized one. The recognition cue is always queried first and, hence, not relevant for the problem of finding optimal cue permutations considered here.

assuming random access to the values of the cues. This observation directs our attention to studying the computational complexity of the problem of finding optimal cue permutations. Is there really an efficient algorithm that solves this problem? We define the decision problem LEXICOGRAPHIC STRATEGY as the task of determining whether for a given partial ordering, represented as a list of pairs of cue profiles, and a given threshold there exists a cue permutation such that the number of incorrect comparisons made by the lexicographic strategy does not exceed this threshold. As a fundamental result we prove that LEXICOGRAPHIC STRATEGY is NP-complete. It follows that TTB is not an algorithm for computing optimal cue permutations and, even more, that no polynomial-time algorithm exists for solving this task, unless the complexity classes P and NP are equal.

The fact that finding optimal cue permutations turns out to be practically intractable, however, does not exclude the possibility that the optimum can be efficiently approximated. The second main topic of this article is an optimization problem called MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY denoting the task of minimizing the number of incorrect inferences for the lexicographic strategy on a given list of pairs. Many computational problems are known to be NP-complete but have efficient approximation algorithms that are good in the sense that their solutions are never more than some constant factor away from the optimum. Problems in this class, which is denoted APX, are generally considered to be approximable well and efficiently (Ausiello et al., 1999). As the second major result of this article we show that, unless P = NP, no polynomial-time approximation algorithm exists that computes solutions for MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY that are only a constant factor worse than the optimum, unless P = NP. In other words, the approximating factor, also called performance ratio, must grow with the size of the problem.

As an extension of TTB we consider an algorithm for finding cue orderings that was called "TTB by Conditional Validity" in the context of bounded rationaility. This algorithm is based on the greedy method, a principle widely used in algorithm design. The greedy algorithm runs in polynomial time and we derive tight bounds for it, showing that it approximates the optimum with a performance ratio proportional to the number of cues. An important consequence of this result is a guarantee that for those instances which have a solution that discriminates all pairs correctly, the greedy algorithm always finds a permutation attaining this minimum. We are not aware that this quality has been established for any of the previously studied heuristics for paired comparison. Moreover, we show that TTB does not have this property, concluding that the greedy method of constructing cue permutations performs provably better than TTB.

While the results mentioned so far deal with lexicographic strategies based on cue permutations, we further consider the possibility to build them by also inverting cues. We present an algorithm that greedily constructs cue inversions that are always correct on a number of pairs that is at least half the optimum. In other words, this algorithm is a constant factor approximation algorithm for the problem of maximizing the number of correct inferences. Interestingly, this algorithm does not even need to permute any cues to approximate to within a constant factor the optimum taken even over all inversions and permutations.

1.3 Learning

LEXICOGRAPHIC STRATEGY is a decision problem that requires to minimize a disagreement. Given a set of pairs, the question is whether a cue permutation can be found that keeps the number of incorrect comparisons, or disagreements, of the lexicographic strategy below some prescribed value. Minimizing disagreement problems play a major role in the context of a computational model of

learning known as agnostic probably approximately correct (PAC) learning (see, e.g., Anthony and Bartlett, 1999). This model assumes that a learner receives a set of examples, the sample, drawn according to some unknown probability distribution. The learner is required to output a function from a so-called hypothesis class on the condition that, with high probability, the computed hypothesis is, with respect to the distribution, close to an optimal hypothesis within the class. A fundamental result is concerned with the question whether agnostic PAC learning with a given hypothesis class can be done efficiently, in particular, if there exists an algorithm that needs only a polynomial number of computation steps to find good hypotheses. The result states that no such learner can exist if the minimizing disagreement problem for the hypothesis class is NP-complete, given that the complexity classes RP and NP are different (see, e.g., Höffgen et al., 1995; Kearns et al., 1994).

The results in this paper have immediate consequences for the question whether lexicographic strategies can be learned. Adopting the framework of agnostic PAC learning, we assume that pairs of cue profiles are drawn randomly according to some unknown distribution. The task of the learner is to find a cue permutation that, with high probability, is close to an optimal one, where closeness means that the probability of differing inferences is small. This setting seems slightly different from the original PAC model as in the latter the sample consists of labeled examples, whereas the lexicographic strategy has to be learned from pairs. However, relevant in both cases is that a hypothesis can be correct or incorrect on a given example. Therefore, applying the above-mentioned result about agnostic PAC learning and assuming that RP $\neq$ NP, by showing that LEXICOGRAPHIC STRATEGY is NP-complete we may conclude that efficient learning of lexicographic strategies is impossible. Moreover, this evidence of impossibility is reinforced by our proving that the optimization problem MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY cannot be approximated in polynomial time to within any constant factor.

A further question that models of learning are involved in is the characterization of the ability to generalize, that is, to find a good hypothesis from only a small number of examples. A principal result in agnostic PAC learning has established a combinatorial parameter of a hypothesis class, its Vapnik-Chervonenkis (VC) dimension, as the relevant measure for this sample complexity (Vapnik and Chervonenkis, 1971). In particular, to come close to the minimal generalization error it is necessary and sufficient to draw a number of examples that is proportional to the VC dimension of the hypothesis class (see, e.g., Anthony and Bartlett, 1999). In this article we determine the VC dimension of the class of lexicographic strategies exactly. In detail, we show that the class of lexicographic strategies obtained by cue permutations and inversions has a VC dimension equal to the number of cues. As a consequence, the number of cues provides a tight bound on the sample complexity for learning lexicographic strategies.

1.4 Related Work

Research that approaches the investigation of simple heuristics for intelligent systems via the analysis of computational complexity traces back to Simon and Kadane (1975, 1976). They provided sufficient conditions under which so-called satisficing search strategies can be proved to be optimal. Their line of study was resumed by Greiner and Orponen (1996) who obtained estimates for the sample complexity of such strategies. Regarding the issue of ordering, Greiner (1999) raised a question relevant for inductive logic programming that is similar to the problems studied here. He asked whether it is possible to efficiently revise rule-based programs by rearranging the ordering of

the rules. His results include NP-completeness and nonapproximability statements for various types of logical theories.

Rivest (1987) introduced decision lists as a formalism for the representation of Boolean functions. The procedure for computing the output value of a decision list is similar to a lexicographic strategy in that both mechanisms are based on one-reason decision making. In fact, we shall show below that lexicographic strategies are a special case of so-called 2-decision lists. It will also follow from this result that the two function classes do not coincide. Thus, an algorithm that learns 2-decision lists does not necessarily learn lexicographic strategies. On the other hand, an algorithm that finds optimal cue permutations might not be good in constructing 2-decision lists.

Ordering problems have also been studied by Cohen et al. (1999). They considered the problem of putting a set of objects in a total order that maximally agrees with a specified preference function. They proved this problem to be NP-complete. We shall show later that the problem of finding cue permutations for the lexicographic strategy can be formulated as such an ordering problem. However, we shall also argue that the two problems are different, since the cue permutation problem requires the total order to be implemented as a lexicographic strategy and not every total order can be represented this way.

1.5 Outline

We introduce lexicographic strategies in Section 2 and provide there further definitions and properties. We then draw comparisons with decision lists and discuss the relationship of the problem of finding optimal cue permutations with the ordering problem studied by Cohen et al. (1999).

Section 3 establishes the NP-completeness of the problem LEXICOGRAPHIC STRATEGY. Additionally, we consider the complexity of this problem when the instances meet certain conditions. We obtain that the problem remains NP-complete under constraints that require the cue profiles to be sparse, impose a bound on the number of pairs, or suppose the pairs to satisfy some simple properties of orderings. In particular, we show NP-completeness to hold when each cue profile contains no more than one 0. In contrast, if the latter condition is met and the pairs are from some partial order, the problem can be solved in linear time.

The optimization problem MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY is considered in Section 4. As the main result we show that this problem cannot be approximated in polynomial time to within any constant factor, unless $\mathrm{P} = \mathrm{NP}$. It further emerges, that this result holds even when the instances satisfy some, albeit not all, of the restrictions considered in Section 3.

Section 5 introduces the greedy algorithm for constructing cue permutations. We tightly determine the performance ratio of this algorithm, showing that it is proportional to the number of cues. The result implies that the greedy method always finds a correct cue permutation if one exists. In contrast, we show that this does not hold for TTB. Restrictions under which the lower bound for the greedy method is still valid are also determined in this section.

In Section 6 we introduce the operation of inverting cues as a means for constructing lexicographic strategies. We show that a greedy method approximates the maximum number of correct inferences to within a constant factor.

The sample complexity for learning is studied in Section 7. We determine the number of cues as the exact value for the VC dimension of the class of lexicographic strategies obtained from cue permutations and inversions. Section 8 summarizes seven major open questions arising from this article and Section 9 concludes with final remarks.

We assume that the reader is acquainted with the theory of NP-completeness as propounded, for instance, by Garey and Johnson (1979). Familiarity with the theory of computational complexity for approximation problems is not required as we shall explicate the necessary details.

Bibliographic Note. The main result of Section 3 (Theorem 3) was mentioned by Martignon and Schmitt (1999), but its proof has been available only in an unpublished manuscript (Schmitt and Martignon, 1999). Parts of Sections 4 and 5 appear in a contribution to a conference (Schmitt and Martignon, 2006).

2. Lexicographic Strategies

In the following, we introduce lexicographic strategies and the computational problem that we study in this article. After giving formal definitions in Section 2.1, we compare in Section 2.2 lexicographic strategies with a related formalism known as decision lists. The optimization problem for lexicographic strategies bears some resemblance to ranking problems that have been studied earlier. In Section 2.3, we discuss the relationship between them and demonstrate that they are different problems.

2.1 Definitions

A *lexicographic strategy* is a method for comparing elements of a set $B \subseteq \{0,1\}^n$ of Boolean vectors. Each component $1,\ldots,n$ of these vectors is referred to as a *cue*. Given two elements $a,b \in B$, where $a = (a_1,\ldots,a_n)$ and $b = (b_1,\ldots,b_n)$, the lexicographic strategy searches for the smallest cue index $i \in \{1,\ldots,n\}$ such that a_i and b_i are different. The strategy then outputs one of " $<$ " or " $>$ " according to whether $a_i < b_i$ or $a_i > b_i$ assuming the usual order $0 < 1$ of the truth values. If no such cue exists, the strategy returns " $=$ ". Formally, let $\mathrm{diff} : B \times B \to \{1,\ldots,n+1\}$ be the function where $\mathrm{diff}(a,b)$ is the smallest cue index on which a and b are different, or $n+1$ if they are equal, that is,

$$\mathrm{diff}(a,b) \quad = \quad \min\{\{i : a_i \neq b_i\} \cup \{n+1\}\}.$$

Then, the function $S : B \times B \to \{\text{“}<\text{”},\text{“}=\text{”},\text{“}>\text{”}\}$ computed by the lexicographic strategy is

$$S(a,b) \quad = \quad \begin{cases} \text{“}<\text{”} & \text{if } \mathrm{diff}(a,b) \leq n \text{ and } a_{\mathrm{diff}(a,b)} < b_{\mathrm{diff}(a,b)}, \\ \text{“}>\text{”} & \text{if } \mathrm{diff}(a,b) \leq n \text{ and } a_{\mathrm{diff}(a,b)} > b_{\mathrm{diff}(a,b)}, \\ \text{“}=\text{”} & \text{otherwise.} \end{cases}$$

Considering a and b as binary encodings of natural numbers, $S(a,b)$ is nothing else than the result of the comparison of these two numbers.

Lexicographic strategies may take into account that the cues come in an order that is different from $1,\ldots,n$. Let $\pi : \{1,\ldots,n\} \to \{1,\ldots,n\}$ be a permutation of the cues. It gives rise to a mapping $\overline{\pi} : \{0,1\}^n \to \{0,1\}^n$ that permutes the components of Boolean vectors by $\overline{\pi}(a_1,\ldots,a_n) = (a_{\pi(1)},\ldots,a_{\pi(n)})$. As $\overline{\pi}$ is uniquely defined given π, we simplify the notation and write also π for $\overline{\pi}$. The *lexicographic strategy under cue permutation* π passes through the cues in the order $\pi(1),\ldots,\pi(n)$, that is, it computes the function $S_\pi : B \times B \to \{\text{“}<\text{”},\text{“}=\text{”},\text{“}>\text{”}\}$ defined as

$$S_\pi(a,b) \quad = \quad S(\pi(a),\pi(b)).$$

The problem we study is that of finding a cue permutation that minimizes the number of incorrect comparisons in a given list of element pairs using the lexicographic strategy. An instance of this problem consists of a set B of elements and a set of pairs $L \subseteq B \times B$. Each pair $\langle a,b\rangle \in L$ represents an inequality $a \leq b$. Given a cue permutation π, we say that the lexicographic strategy under π *infers* the pair $\langle a,b\rangle$ *correctly* if $S_\pi(a,b) \in \{\text{“}<\text{”},\text{“}=\text{”}\}$, otherwise the inference is incorrect. The task is to find a permutation π such that the number of incorrect inferences in L using S_π is minimal, that is, a permutation π that minimizes

$$\text{INCORRECT}(\pi, L) \quad = \quad |\{\langle a,b\rangle \in L : S_\pi(a,b) = \text{“}>\text{”}\}|.$$

We recall some definitions about orders on sets. A set $L \subseteq B \times B$ is a *partial order* if it is reflexive (that is, $\langle a,a\rangle \in L$ for every $a \in B$), antisymmetric (that is, $\langle a,b\rangle \in L$ and $\langle b,a\rangle \in L$ implies $a = b$), and transitive (that is, $\langle a,b\rangle \in L$ and $\langle b,c\rangle \in L$ implies $\langle a,c\rangle \in L$). Further, L is a *total order* if it is a partial order and satisfies $\langle a,b\rangle \in L$ or $\langle b,a\rangle \in L$ for every $a,b \in B$. Finally, L is *irreflexive* if $\langle a,a\rangle \notin L$ for every $a \in B$.

Given some cue permutation π, consider a relation that is satisfied by a pair $\langle a,b\rangle$ if and only if $S_\pi(a,b) \in \{\text{“}<\text{”},\text{“}=\text{”}\}$. Clearly, this relation defines a total order on any set $B \subseteq \{0,1\}^n$. A question that arises immediately is whether every total order has some cue permutation that represents this order using the lexicographic strategy. It is easy to see that this is not the case.

Proposition 1 *For every set $B \subseteq \{0,1\}^n$ and every cue permutation π, the lexicographic strategy under cue permutation π defines a total order on B. On the other hand, there are sets $B \subseteq \{0,1\}^n$ with a total order that cannot be represented by any cue permutation.*

Proof It is evident that the relation $\{(a,b) : S_\pi(a,b) \in \{\text{“}<\text{”},\text{“}=\text{”}\}\}$ is a total order. As a counterexample, consider a set B with $\{(0,\ldots,0),(1,\ldots,1)\} \subseteq B$. Clearly, under every cue permutation, $(0,\ldots,0)$ is less than $(1,\ldots,1)$. Thus, the reverse ordering of these two elements cannot be represented by the lexicographic strategy. ■

Obviously, the lexicographic strategy applied to a pair $\langle a,a\rangle$ is always correct, independently of the cue permutation. Therefore, the identical pairs of L pose no obstacle for the minimization problem. Also possible were an alternative setting where $\langle a,b\rangle$ is interpreted as a strict inequality. We admit identical pairs, however, to keep the definition more general and allow L to represent some “natural” relations such as partial or total orders or arbitrary subsets thereof. Nevertheless, all results presented in the following remain valid if the pairs are assumed to represent strict inequalities.

2.2 Lexicographic Strategies and Decision Lists

Decision lists are computing formalisms that operate quite similar to lexicographic strategies. A *decision list* represents a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and is given by a list of pairs

$$(m_1, r_1), \ldots, (m_\ell, r_\ell),$$

where each m_i is a Boolean monomial, that is, a conjunction of Boolean variables with or without negations (Rivest, 1987). Further, each r_i is 0 or 1, and m_ℓ is the constant function 1. The Boolean function computed by the decision list is defined as follows: Given some $a \in \{0,1\}^n$, the output

value is r_i where i is the smallest index such that m_i evaluates to 1 on a. A k-decision list is a decision list where every monomial has size at most k.

In the problem of minimizing the number of incorrect comparisons the relevant question is whether the output of the lexicographic strategy is correct, and not whether it is particularly one of "$<$", "$=$", or "$>$". In other words, we are interested in a binary and not a ternary classification. Thus, we may consider the lexicographic strategy S as a Boolean function f mapping a set L of pairs to $\{0,1\}$, where for every $\langle a,b\rangle \in L$ we have

$$f(a,b) = 1 \quad \text{if and only if} \quad S(a,b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\}.$$

Seen in this light, lexicographic strategies exhibit a similarity to decision lists. The following statement, which is easy to derive, makes this relationship precise.

Proposition 2 *Let $f : \{0,1\}^{2n} \to \{0,1\}$ be a Boolean function with variables $x_1,\ldots,x_n$ and $y_1,\ldots,y_n$. Then f is computed by the lexicographic strategy if and only if f is computed by the 2-decision list*

$$(x_1\overline{y}_1,0),(\overline{x}_1y_1,1),\ldots,(x_n\overline{y}_n,0),(\overline{x}_ny_n,1),(1,1).$$

Proof Let $a,b \in \{0,1\}^n$. Clearly, if $a = b$, all monomials of the decision list evaluate to 0, except for the constant function 1. If $a \neq b$, let $i = \mathrm{diff}(a,b)$. In the case that $a_i < b_i$, the monomial $\overline{x}_iy_i$ is the first one that evaluates to 1, and the output of the decision list is 1. Similarly, if $a_i > b_i$, this is first detected by the monomial $x_i\overline{y}_i$, and the decision list yields 0. ■

The proposition shows that the lexicographic strategy has a unique characterization as a 2-decision list. Thus, finding a cue permutation for the lexicographic strategy amounts to constructing a 2-decision list with some restrictions concerning the structure of the monomials, the pattern of the output values, and the length of the list. It is also obvious from Proposition 2, however, that 2-decision lists compute a much richer class of Boolean functions than lexicographic strategies do. We conclude that cue permutations are not necessarily found using algorithms for constructing 2-decision lists. Further, an optimal cue permutation might not be an optimal 2-decision list.

2.3 Ranking Problems

The problem of minimizing the number of incorrect comparisons in a list of pairs exhibits some similarity with an optimization problem that occurs in the context of ordering problems and was studied by Cohen et al. (1999). In this problem, which we here call ranking problem, one receives a set X, a collection of functions $R_1,\ldots,R_N$ mapping $X \times X$ to the real interval $[0,1]$, and rational numbers $w_1,\ldots,w_N \in [0,1]$ whose sum is equal to 1. A solution of the problem is a total order ρ of X that maximally agrees with the so-called preference function $\mathrm{PREF} : X \times X \to [0,1]$. The closer the value of $\mathrm{PREF}(a,b)$ is to 1, the more a is to be ranked above b. The preference function is defined as

$$\mathrm{PREF}(a,b) \quad = \quad \sum_{i=1}^{N} w_i R_i(a,b)$$

The agreement of the total order ρ with the preference function PREF is quantified by the value of

$$\sum_{\{(a,b):\rho(a)>\rho(b)\}} \mathrm{PREF}(a,b) \tag{1}$$

and a desired total order ρ is one that maximizes this value.

It is not hard to see that the instances of the cue permutation problem are particular instances of the above problem. Specifically, introduce for each pair $\langle a,b\rangle$ a function $R_{\langle a,b\rangle} : B \times B \to \{0,1\}$ that outputs 1 on (b,a), and 0 otherwise. Further, let $w_{\langle a,b\rangle} = 1/|L|$. Then, a total order ρ that maximizes the value of the expression (1) is one that minimizes the number of incorrect inferences in L.

Cohen et al. (1999) have shown that the ranking problem is NP-complete. The question is, therefore, whether this hardness result has any implications on the complexity of finding a cue permutation that minimizes the number of incorrect inferences. However, the ranking problem is different from the cue permutation problem not only in that its instances are more general. The two problems also disagree in the type of solutions that are sought. While the ranking problem accepts any total order that maximizes the agreement with the preference function, the cue permutation problem requires that the total order can be implemented by a lexicographic strategy. Proposition 1 demonstrates, though, that not every total order can be represented as a cue permutation. Thus, the space taken by the solutions of the cue permutation problem is narrower than the solution space for the ranking problem described above. Moreover, we show in Section 3 that the cue permutation problem remains NP-complete even when the instances are known to have a total order. In contrast, imposing this restriction on the ranking problem results in a problem that is trivially solvable.

A further difference emerges if one considers the problem of approximating optimal solutions as we do in Section 4. Then the cue permutation problem is a minimization problem while the ranking problem is a maximization problem. Among the complexity classes of approximation problems several examples are known where the minimization and the maximization problem have different degrees of approximability (see, e.g., Amaldi and Kann, 1995, 1998). Consequently, despite the apparent similarity of the cue permutation problem and the ranking problem, the complexities of the two problems are obviously not related.

3. Complexity of Finding Optimal Cue Permutations

We consider the complexity of the problem to minimize the number of incorrect inferences under the lexicographic strategy. To show that it is computationally intractable, we formulate this search problem as a decision problem. The decision problem has as input a set of binary vectors, an ordering defined on this set in terms of a list of vector pairs, and a bound given as a natural number. The question is to decide whether the cues can be permuted such that the number of incorrect inferences made by the lexicographic strategy when applied with this cue permutation to the list of pairs is not larger than the given bound. We call this decision problem LEXICOGRAPHIC STRATEGY.

LEXICOGRAPHIC STRATEGY
Instance: A set $B \subseteq \{0,1\}^n$, a set $L \subseteq B \times B$, and a natural number k.
Question: Is there a permutation of the cues of B such that the number of incorrect inferences in L under the lexicographic strategy is at most k?

Clearly, any polynomial-time algorithm for finding a permutation with a minimal number of incorrect inferences can be turned into a polynomial-time algorithm that solves LEXICOGRAPHIC STRATEGY. However, we show that this problem is NP-hard. Hence, if P ≠ NP, no polynomial-time algorithm for the decision problem and, a fortiori, for the search problem exists. The NP-hardness proof provides a polynomial-time reduction from a problem dealing with graphs and known as VERTEX COVER (Garey and Johnson, 1979).

VERTEX COVER
Instance: An undirected graph $G = (V,E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges, and a natural number k.
Question: Is there a vertex cover of cardinality k or less for G, that is, a subset $V' \subseteq V$ with $|V'| \leq k$ such that for each edge $\{u,v\} \in E$ at least one of u and v belongs to V'?

Theorem 3 LEXICOGRAPHIC STRATEGY *is NP-complete.*

Proof Obviously, a nondeterministic algorithm can generate a permutation of the cues and verify in polynomial time whether the number of incorrect inferences is at most k. Thus, the problem is a member of NP. To establish its NP-hardness, we construct a reduction from VERTEX COVER. Let 1_i ($1_{i,j}$) denote the n-bit vector with a 1 in every position except for position i (positions i and j) where it has a 0. Further, 1 is the n-bit vector with a 1 everywhere. Given the graph $G = (V,E)$, where the set of vertices is $V = \{v_1, \ldots, v_n\}$, we define a set B of Boolean vectors with $n+1$ cues, that is $B \subseteq \{0,1\}^{n+1}$, in three steps:

1. Let $(1,0) \in B$.
2. For $i = 1, \ldots, n$, let $(1_i, 1) \in B$.
3. For every $\{v_i, v_j\} \in E$, let $(1_{i,j}, 1) \in B$.

The set $L \subseteq B \times B$ of pairs that represents the element ordering is defined such that the element from step 1 is less than each element constructed in step 2, and each element arising from step 3 is less than the element from step 1. Formally,

$$L \quad = \quad \{\langle (1,0), (1_i,1)\rangle : i = 1, \ldots, n\} \cup \{\langle (1_{i,j},1), (1,0)\rangle : \{v_i, v_j\} \in E\}. \tag{2}$$

Finally, we let the number k in the instance of LEXICOGRAPHIC STRATEGY be the same as in the instance of VERTEX COVER. Clearly, the reduction is computable in polynomial time.

We establish the correctness of the reduction by proving that the graph G has a vertex cover of cardinality at most k if and only if the associated instance of LEXICOGRAPHIC STRATEGY has a cue permutation that results in no more than k incorrect inferences. For simplicity, let us call a pair from the first and second set on the right-hand side of equation (2) a vertex pair and an edge pair, respectively.

($\Rightarrow$) Assume that G has a vertex cover V' of cardinality at most k and, without loss of generality, let its cardinality be exactly k, so that $V' = \{v_{i_1}, \ldots, v_{i_k}\}$. Further, let $V \setminus V' = \{v_{i_{k+1}}, \ldots, v_{i_n}\}$. Define the permutation of the cues as

$$i_1, \ldots, i_k, n+1, i_{k+1}, \ldots, i_n.$$

We claim that this cue ranking causes no more than k incorrect inferences in L. Consider an arbitrary edge pair $\langle (1_{i,j},1), (1,0)\rangle$. As V' is a vertex cover, at least one of i and j occurs in $i_1, \ldots, i_k$. This implies that the first cue that distinguishes this pair will have value 0 in $(1_{i,j},1)$ and value 1 in $(1,0)$. Thus, the result of the lexicographic comparison is correct. Next, let $\langle (1,0), (1_i,1)\rangle$ be a vertex pair with $v_i \notin V'$. In this case, cue $n+1$ distinguishes this pair with the correct outcome. Finally, each vertex pair $\langle (1,0), (1_i,1)\rangle$ with $v_i \in V'$ is distinguished by cue i with a result different from

the ordering given by L. In summary, the only incorrect comparisons arise from vertex pairs with $v_i \in V'$. As V' has cardinality k, we thus have no more than k incorrect inferences.

($\Leftarrow$) Now, let π be a permutation of the cues that produces at most k incorrect inferences in L. Define the set V' of vertices as follows:

1. For every incorrect vertex pair $\langle (1,0),(1_i,1)\rangle$, let $v_i \in V'$.

2. For every incorrect edge pair $\langle (1_{i,j},1),(1,0)\rangle$, let one of $v_i, v_j \in V'$.

Clearly, V' has cardinality at most k. It remains to show that V' is a vertex cover. For the sake of a contradiction, assume that there is an edge in E, say $\{v_i, v_j\}$, not covered. This means that neither of v_i, v_j is in V', implying that we have correct comparisons for the vertex pairs corresponding to v_i and v_j and for the edge pair corresponding to $\{v_i, v_j\}$. The fact that the edge pair is inferred correctly implies that π must rank cue i or j before cue $n+1$. But then we have that at least one of the vertex pairs for v_i and v_j results in an incorrect comparison. This contradicts the assertion made above that both vertex pairs have correct comparisons. We conclude that V' is a vertex cover. ■

The reduction constructed in the previous proof has some properties that we exploit in the following statement to establish the NP-completeness of restricted versions of LEXICOGRAPHIC STRATEGY. First, it shows that the set B can be sparse in a certain sense, that is, has elements that exhibit only very constrained bit patterns. Moreover, the NP-completeness holds even when L is not much larger than B. Finally, the problem remains intractable even if L does not contain identical pairs or has some properties of a partial or total order.

Corollary 4 LEXICOGRAPHIC STRATEGY *is NP-complete even when the instances satisfy any (or all) of the following constraints:*

1. *Each element of B contains at most two* 0*s.*

2. *The cardinality of L is linearly bounded from above by the cardinality of B, that is, $|L|$ is $O(|B|)$.*

3. *L is irreflexive.*

4. *L is a subset of some partial order.*

5. *L is a subset of some total order.*

Proof We show that all constraints are satisfied by the instances defined in the reduction for the proof of Theorem 3. That the first condition holds is obvious from the definition of B. Further, the instances of LEXICOGRAPHIC STRATEGY in this reduction all satisfy $|B| = |E| + n + 1$ and $|L| = n + |E|$. Thus, $|B| = |L| - 1$ and the second constraint is met. Moreover, L does not contain any pair $\langle a,a\rangle$ which implies that the third constraint holds. We establish the fourth condition by checking that L does not violate any of the requirements for a partial order: Clearly, each $a \neq b$ does not have both $\langle a,b\rangle$ and $\langle b,a\rangle$ in L, and there are no three pairs $\langle a,b\rangle, \langle b,c\rangle, \langle c,a\rangle$ in L. Finally, it is easy to see that L is consistent with the total order resulting from the following ascending arrangement of B: We begin with the elements $(1_{i,j},1)$, where $\{v_i, v_j\} \in E$, in lexicographic order, followed by the element $(1,0)$, and complete this sequence at the end by the elements $(1_i,1)$, for

$i = 1, \ldots, n$, again in lexicographic order. Thus, we have an ordering where any two elements of B are comparable, implying that also the last constraint is satisfied. ■

The first constraint of Corollary 4 gives rise to the question whether the problem is still NP-complete if each element of B has no more than one 0. The following two results treat this issue. First, we show that the problem in general remains NP-complete under this restriction. To establish this we provide a reduction from the NP-complete problem FEEDBACK ARC SET (Garey and Johnson, 1979).

FEEDBACK ARC SET
Instance: A directed graph $G = (V,E)$, where V is the set of vertices and $A \subseteq V \times V$ is the set of arcs, and a natural number k.
Question: Is there a subset $A' \subseteq A$ with $|A'| \leq k$ such that A' contains at least one arc from every directed cycle in G?

Theorem 5 LEXICOGRAPHIC STRATEGY *is NP-complete even when restricted to instances where each element of B contains at most one* 0*.*

Proof Clearly, as LEXICOGRAPHIC STRATEGY is in NP, any subproblem of it is in NP as well. We establish the NP-hardness of the problem by giving a reduction that is a simple rewriting of FEEDBACK ARC SET. Given the graph $G = (V,A)$ with $V = \{v_1, \ldots, v_n\}$ and using the notation from the proof of Theorem 3, we let

$$\begin{aligned} B &= \{1_i : i = 1, \ldots, n\}, \\ L &= \{\langle 1_i, 1_j \rangle : (v_i, v_j) \in A\}, \end{aligned}$$

and define k to have the same value as in the instance of FEEDBACK ARC SET.

Obviously, $A' \subseteq A$ contains at least one arc from every directed cycle in G if and only if the graph $G' = (V, A \setminus A')$ is acyclic. Further, G' is acyclic if and only if V has a total ordering in which v_i is less than v_j for each $(v_i, v_j) \in A \setminus A'$. Finally, the existence of such a total ordering is equivalent to the assertion that B has a cue permutation with no incorrect comparisons in $L' = \{\langle 1_i, 1_j \rangle : (v_i, v_j) \in A \setminus A'\}$. With this chain of equivalences, the correctness of the reduction follows from the fact that $|L'| = |L| - |A'|$. ■

We may also add to the assumption of Theorem 5 the restriction that $|L|$ is linearly bounded in $|B|$, so that the problem is still NP-complete. In this case, the NP-hardness follows from the fact that FEEDBACK ARC SET remains NP-hard for directed graphs in which the degree of the vertices is bounded by some constant (Garey and Johnson, 1979). However, if we include the constraint that L is a subset of some partial order, the complexity of the problem changes drastically, as we see in the following statement.

Corollary 6 *The problem of finding a cue permutation with a minimal number of incorrect comparisons under the lexicographic strategy is solvable in linear time for instances where B contains at most one* 0 *and L is a subset of some partial order.*

Proof As was argued in the proof of Theorem 5, the problem is the same as the problem of finding a total order that is consistent with the partial order given by L (which is always possible). Such a total order can be constructed by topological sorting. Algorithms for this sorting problem exist that run in linear time (see, e.g., Skiena, 1997). ■

It is not difficult—and we leave it to the reader—to establish dual formulations of Theorem 5 and Corollary 6 where it is assumed that each element of B contains at most one 1.

4. Approximability of Optimal Cue Permutations

In the previous section, we have shown that there is no polynomial-time algorithm that computes optimal cue permutations for the lexicographic strategy, unless P = NP. While it follows that this problem is as difficult as all other optimization problems that have an NP-complete decision problem, we cannot draw any conclusions for the case where we are interested in solutions that are not equal to the optimum but somehow close to it. In fact, there is a large class of optimization problems that have NP-complete decision problems, but can be solved efficiently if the solution is required to be only a constant factor worse than the optimal solution. This class of problems is denoted APX (Ausiello et al., 1999).

In this section, we show that the problem of approximating the optimal cue permutation is harder than any problem in the class APX. In particular, we prove that, if P $\neq$ NP, there is no polynomial-time algorithm whose solutions yield a number of incorrect comparisons that is by at most a constant factor larger than the minimal number possible. First, however, we state the problem as an optimization problem and introduce some definitions from the complexity theory of approximation problems (Ausiello et al., 1999).

MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY
Instance: A set $B \subseteq \{0,1\}^n$ and a set $L \subseteq B \times B$.
Solution: A permutation π of the cues of B.
Measure: The number of incorrect inferences in L for the lexicographic strategy under cue permutation π, that is, INCORRECT(π, L).

Given a real number $r > 0$, an algorithm is said to approximate MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY to within a factor of r if for every instance (B, L) the algorithm returns a permutation π such that

$$\text{INCORRECT}(\pi, L) \quad \leq \quad r \cdot \text{opt}(L),$$

where opt(L) is the minimal number of incorrect comparisons achievable on L by any permutation. The factor r is also known as the performance ratio of the algorithm. The following optimization problem plays a crucial role in the derivation of the lower bound for the approximability of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY.

MINIMUM HITTING SET
Instance: A collection C of subsets of a finite set U.
Solution: A hitting set for C, that is, a subset $U' \subseteq U$ such that U' contains at least one element from each subset in C.
Measure: The cardinality of the hitting set, that is, $|U'|$.

Similarly as above, we say that an algorithm approximates MINIMUM HITTING SET to within a factor of r if for every instance C the algorithm outputs a hitting set U' that satisfies

$$|U'| \quad \leq \quad r \cdot \mathrm{opt}(C),$$

where $\mathrm{opt}(C)$ denotes the minimal cardinality of a hitting set for C. (For simplicity, we use $\mathrm{opt}(\cdot)$ to represent the value of an optimal solution in both problems. It shall be clear from the context to which problem it refers.)

MINIMUM HITTING SET is equivalent to a problem called MINIMUM SET COVER in the sense that every polynomial-time algorithm that approximates MINIMUM HITTING SET to within a certain factor can be turned into a polynomial-time algorithm that approximates MINIMUM SET COVER to within the same factor, and vice versa (Ausiello et al., 1980). Bellare et al. (1993) have shown that MINIMUM SET COVER cannot be approximated in polynomial time to within any constant factor, unless P = NP. Thus, if P $\neq$ NP, MINIMUM HITTING SET cannot be approximated in polynomial time to within any constant factor as well. We make use of this fact when we establish the lower bound for the approximability of the optimal cue permutation.

Theorem 7 *For every r, there is no polynomial-time algorithm that approximates* MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY *to within a factor of r, unless* P = NP.

Proof We use the main ideas from the proof of Theorem 3 to establish an approximation preserving reduction, or AP-reduction, from MINIMUM HITTING SET to MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY.[3] (See Ausiello et al., 1999, for a definition of the AP-reduction.) This reduction entails that every polynomial-time algorithm that approximates MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY to within some constant factor can be turned into a polynomial-time algorithm that approximates MINIMUM HITTING SET to within the same constant factor. Then the statement follows from the equivalence of MINIMUM HITTING SET to MINIMUM SET COVER and the lower bound on the approximability of the latter (Bellare et al., 1993).

We first define a function f that is computable in polynomial time and maps each instance of MINIMUM HITTING SET to an instance of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY. Let 1 denote the n-bit vector with a 1 everywhere and $1_{i_1,\ldots,i_\ell}$ the vector with 0 in positions $i_1,\ldots,i_\ell$ and 1 elsewhere. Given the collection C of subsets of the set $U = \{u_1,\ldots,u_n\}$, the function f maps C to (B,L), where $B \subseteq \{0,1\}^{n+1}$ is defined as follows:

1. Let $(1,0) \in B$.
2. For $i = 1,\ldots,n$, let $(1_i,1) \in B$.
3. For every $\{u_{i_1},\ldots,u_{i_\ell}\} \in C$, let $(1_{i_1,\ldots,i_\ell},1) \in B$.

Further, the set L is constructed as

$$L \quad = \quad \{\langle(1,0),(1_i,1)\rangle : i = 1,\ldots,n\} \cup \{\langle(1_{i_1,\ldots,i_\ell},1),(1,0)\rangle : \{u_{i_1},\ldots,u_{i_\ell}\} \in C\}. \qquad (3)$$

3. A proof of Theorem 3 can be obtained by employing this reduction as a reduction between decision problems, from the NP-complete HITTING SET to LEXICOGRAPHIC STRATEGY. However, the reduction used in the proof of Theorem 3 is more powerful since Corollary 4 cannot be inferred when reducing from HITTING SET.

In the following, a pair from the first and second set on the right-hand side of equation (3) is referred to as an element pair and a subset pair, respectively. Obviously, the function f is computable in polynomial time. It has the following property.

Claim 1. *Let $f(C) = (B,L)$. If C has a hitting set of cardinality k or less then $f(C)$ has a cue permutation π where* INCORRECT$(\pi, L) \leq k$.

To prove this, assume without loss of generality that C has a hitting set U' of cardinality exactly k, say $U' = \{u_{j_1}, \ldots, u_{j_k}\}$, and let $U \setminus U' = \{u_{j_{k+1}}, \ldots, u_{j_n}\}$. Then the cue permutation

$$j_1, \ldots, j_k, n+1, j_{k+1}, \ldots, j_n.$$

results in no more than k incorrect inferences in L. Indeed, consider an arbitrary subset pair $\langle (1_{i_1,\ldots,i_\ell}, 1), (1,0) \rangle$. To not be an error, one of $i_1, \ldots, i_\ell$ must occur in the hitting set $j_1, \ldots, j_k$. Hence, the first cue that distinguishes this pair has value 0 in $(1_{i_1,\ldots,i_\ell}, 1)$ and value 1 in $(1,0)$, resulting in a correct comparison. Further, let $\langle (1,0), (1_i, 1) \rangle$ be an element pair with $u_i \notin U'$. This pair is distinguished correctly by cue $n+1$. Finally, each element pair $\langle (1,0), (1_i, 1) \rangle$ with $u_i \in U'$ is distinguished by cue i with a result that disagrees with the ordering given by L. Thus, only element pairs with $u_i \in U'$ yield incorrect comparisons and subset pairs are inferred correctly. Hence, the number of incorrect inferences is not larger than $|U'|$.

Next, we define a polynomial-time computable function g that maps each collection C of subsets of a finite set U and each cue permutation π for $f(C)$ to a subset of U. Given that $f(C) = (B,L)$, the set $g(C,\pi) \subseteq U$ is defined as follows:

1. For every element pair $\langle (1,0), (1_i, 1) \rangle \in L$ that is compared incorrectly by π, let $u_i \in g(C,\pi)$.
2. For every subset pair $\langle (1_{i_1,\ldots,i_\ell}, 1), (1,0) \rangle \in L$ that is compared incorrectly by π, let one of the elements $u_{i_1}, \ldots, u_{i_\ell} \in g(C,\pi)$.

Clearly, the function g is computable in polynomial time. It satisfies the following condition.

Claim 2. *Let $f(C) = (B,L)$. If* INCORRECT$(\pi, L) \leq k$ *then $g(C,\pi)$ is a hitting set of cardinality k or less for C.*

Obviously, if INCORRECT$(\pi, L) \leq k$ then $g(C,\pi)$ has cardinality at most k. To show that it is a hitting set, assume the subset $\{u_{i_1}, \ldots, u_{i_\ell}\} \in C$ is not hit by $g(C,\pi)$. Then neither of $u_{i_1}, \ldots, u_{i_\ell}$ is in $g(C,\pi)$. Hence, we have correct comparisons for the element pairs corresponding to $u_{i_1}, \ldots, u_{i_\ell}$ and for the subset pair corresponding to $\{u_{i_1}, \ldots, u_{i_\ell}\}$. As the subset pair is distinguished correctly, one of the cues $i_1, \ldots, i_\ell$ must be ranked before cue $n+1$. But then at least one of the element pairs for $u_{i_1}, \ldots, u_{i_\ell}$ yields an incorrect comparison. This contradicts the assertion that the comparisons for these element pairs are all correct. Thus, $g(C,\pi)$ is a hitting set and the claim is established.

Assume now that there exists a polynomial-time algorithm A that approximates MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY to within a factor of r. Consider the algorithm that, for a given instance C of MINIMUM HITTING SET as input, calls algorithm A with input $(B,L) = f(C)$, and returns $g(C,\pi)$ where π is the output provided by A. Clearly, this new algorithm runs in polynomial time. We show that it approximates MINIMUM HITTING SET to within a factor of r. By the assumed approximation property of algorithm A, we have

$$\text{INCORRECT}(\pi, L) \quad \leq \quad r \cdot \text{opt}(L).$$

Together with Claim 2, this implies that $g(\pi, C)$ is a hitting set for C satisfying

$$|g(C,\pi)| \quad \leq \quad r \cdot \mathrm{opt}(L).$$

From Claim 1 we obtain $\mathrm{opt}(L) \leq \mathrm{opt}(C)$ and, thus,

$$|g(C,\pi)| \quad \leq \quad r \cdot \mathrm{opt}(C).$$

Thus, the proposed algorithm for MINIMUM HITTING SET violates the approximation lower bound that holds for this problem under the assumption P $\neq$ NP. This proves the statement of the theorem. ■

Similarly as in Corollary 4 we can state a stronger version of Theorem 7 that takes restrictions into account that may hold for the instances of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY. The proof is obtained in the same way as the proof of Corollary 4 and not given here.

Corollary 8 *If* P $\neq$ NP, *then for every r there is no polynomial-time algorithm that approximates* MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY *to within a factor of r, even when the instances satisfy any (or all) of the following constraints:*

1. *The cardinality of L is linearly bounded from above by the cardinality of B, that is, $|L|$ is $O(|B|)$.*
2. *L is irreflexive.*
3. *L is a subset of some partial order.*
4. *L is a subset of some total order.*

The reader may have noticed that the constraint of Corollary 4 that imposes a bound on the number of 0s in the elements of B is missing here. In fact, there is some evidence, that the construction of an approximation preserving reduction from MINIMUM HITTING SET to this subproblem of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY is difficult or even impossible. The case where the number of 0s is bounded by some constant corresponds to the subproblem of MINIMUM HITTING SET where the cardinality of each subset is not larger than a constant. This restricted version of MINIMUM HITTING SET is known to be approximable to within some constant factor (Bar-Yehuda and Even, 1981; Hochbaum, 1982). Of course, this apparent relationship does not prove anything about the complexity of approximating the subproblem of MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY. However, it gives reason to the conjecture that this subproblem might have a constant-factor approximation algorithm.

5. Greedy Approximation of Optimal Cue Permutations

The so-called greedy approach to the solution of a computation or approximation problem is helpful when it is not known which algorithm performs best. This simple heuristic often provides satisfactory solutions in many situations in practice. The algorithm GREEDY CUE PERMUTATION that we introduce here is based on the greedy method. The idea is to select the first cue according to which single cue makes a minimum number of incorrect inferences (choosing one arbitrarily if there are

Algorithm 1 GREEDY CUE PERMUTATION

Input: a set $B \subseteq \{0,1\}^n$ and a set $L \subseteq B \times B$
Output: a cue permutation π for n cues
  $I := \{1,\ldots,n\}$;
  for $i = 1,\ldots,n$ **do**
    let $j \in I$ be a cue where $\text{INCORRECT}(j,L) = \min_{j' \in I} \text{INCORRECT}(j',L)$;
    $\pi(i) := j$;
    $I := I \setminus \{j\}$;
    $L := L \setminus \{\langle a,b\rangle : a_j \neq b_j\}$
  end for.

two or more). After that the algorithm removes those pairs that are distinguished by the selected cue, which is reasonable as the distinctions drawn by this cue cannot be undone by later cues. This procedure is then repeated on the set of pairs left. The description of GREEDY CUE PERMUTATION is given as Algorithm 1. It employs an extension of the function INCORRECT, first defined in Section 2.1, applicable also to single cues, such that for a cue i we say

$$\text{INCORRECT}(i,L) \quad = \quad |\{\langle a,b\rangle \in L : a_i > b_i\}|.$$

It is evident that Algorithm 1 runs in polynomial time, but how good is it? The least one should demand from a good heuristic is that, whenever a minimum of zero is attainable, it finds such a solution. This is indeed the case with GREEDY CUE PERMUTATION as we show in the following result. Moreover, a general performance ratio for the approximation of the optimum is asserted here.

Theorem 9 *The algorithm* GREEDY CUE PERMUTATION *approximates* MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY *to within a factor of n, where n is the number of cues. In particular, it always finds a cue permutation with no incorrect inferences if one exists.*

Proof We show by induction on n that the permutation returned by the algorithm makes a number of incorrect inferences no larger than $n \cdot \text{opt}(L)$. If $n = 1$, the optimal cue permutation is definitely found.

Let $n > 1$. Clearly, as the incorrect inferences of a cue cannot be reversed by other cues, there is a cue j with

$$\text{INCORRECT}(j,L) \leq \text{opt}(L).$$

The algorithm selects such a cue in the first round of the loop. During the rest of the rounds, a permutation of $n-1$ cues is constructed for the set of remaining pairs. Let j be the cue that is chosen in the first round, $I' = \{1,\ldots,j-1,j+1,\ldots,n\}$, and $L' = L \setminus \{\langle a,b\rangle : a_j \neq b_j\}$. Further, let $\text{opt}_{I'}(L')$ denote the minimum number of incorrect inferences taken over the permutations of I' on the set L'. Then, we observe that

$$\text{opt}(L) \;\geq\; \text{opt}(L') \;=\; \text{opt}_{I'}(L').$$

The inequality is valid because of $L \supseteq L'$. (Note that $\text{opt}(L')$ refers to the minimum taken over the permutations of all cues.) The equality holds as cue j does not distinguish any pair in L'. By the induction hypothesis, rounds 2 to n of the loop determine a cue permutation π' with

$$\text{INCORRECT}(\pi',L') \leq (n-1) \cdot \text{opt}_{I'}(L').$$

$\langle$ 001 , 010 $\rangle$
$\langle$ 010 , 100 $\rangle$
$\langle$ 010 , 101 $\rangle$
$\langle$ 100 , 111 $\rangle$

Figure 1: A set of lexicographically ordered pairs with nondecreasing cue validities ($1, 1/2$, and $2/3$). The cue ordering of TTB $(1,3,2)$ causes an incorrect inference on the first pair. By Theorem 9, GREEDY CUE PERMUTATION finds the lexicographic ordering.

Thus, the number of incorrect inferences made by the permutation π finally returned by the algorithm satisfies

$$\text{INCORRECT}(\pi, L) \quad \leq \quad \text{INCORRECT}(j, L) + (n-1) \cdot \text{opt}_{I'}(L'),$$

which is, by the inequalities derived above, not larger than $\text{opt}(L) + (n-1) \cdot \text{opt}(L)$ as stated. ■

The special property of GREEDY CUE PERMUTATION that it always finds the minimum if this has value zero is not owned by TTB as demonstrated by the following result.

Corollary 10 *On inputs that have a cue ordering without incorrect comparisons under the lexicographic strategy,* GREEDY CUE PERMUTATION *can be better than TTB.*

Proof Figure 1 shows a set of four lexicographically ordered pairs. According to Theorem 9, GREEDY CUE PERMUTATION comes up with the given permutation of the cues. The validities are $1, 1/2$, and $2/3$. Thus, TTB ranks the cues as $1,3,2$ whereupon the first pair is inferred incorrectly. ■

Next, we consider lower bounds on the performance ratio of GREEDY CUE PERMUTATION. We obtain bounds in terms of n and $|L|$. It emerges in particular that the upper bound obtained in Theorem 9 is optimal up to the factor 2.

Theorem 11 *The performance ratio of* GREEDY CUE PERMUTATION *is at least*

$$\max\{n/2, |L|/2\}.$$

Proof We show how to construct for every n an instance on which GREEDY CUE PERMUTATION has the claimed performance ratio. Let $B = \{a^{(0)}, \ldots, a^{(n)}, b\} \subseteq \{0,1\}^n$ be the set where $a^{(0)} = (0, \ldots, 0)$, $b = (1, 0, \ldots, 0, 1)$, and $a^{(i)}$, for $i = 1, \ldots, n$, is the vector with a 1 in position i and 0 elsewhere. The set $L \subseteq B \times B$ is defined as

$$L \quad = \quad \{\langle a^{(n)}, a^{(0)}\rangle, \langle b, a^{(1)}\rangle\} \cup \{\langle a^{(i)}, a^{(n)}\rangle : i = 2, \ldots, n-1\}.$$

Figure 2 shows the set L for the case $n = 6$. As can be seen, cue 1 is correct on all pairs, cue n is incorrect on two pairs, and every cue $j \in \{2, \ldots, n-1\}$ satisfies $\text{INCORRECT}(j, L) = 1$. Hence,

$\langle$ 000001 , 000000 $\rangle$
$\langle$ 100001 , 100000 $\rangle$
$\langle$ 010000 , 000001 $\rangle$
$\langle$ 001000 , 000001 $\rangle$
$\langle$ 000100 , 000001 $\rangle$
$\langle$ 000010 , 000001 $\rangle$

Figure 2: A set of pairs providing a lower bound on the performance ratio of GREEDY CUE PERMUTATION (Theorem 11).

GREEDY CUE PERMUTATION selects cue 1 as the first cue. As this cue does not distinguish any pair, L is left unchanged. Then, one of the cues $2,\ldots,n-1$ is selected as the second cue. After removal of the pair distinguished by this cue, the remaining cues make the same incorrect inferences as before. Thus, the algorithm keeps on choosing cues from $\{2,\ldots,n-1\}$ during rounds $2,\ldots,n-1$ of the loop until cue n is selected in the last round. The resulting permutation π has cue 1 in its first position, cues from $\{2,\ldots,n-1\}$ in positions $2,\ldots,n-1$, and cue n in the last position. This implies that $\text{INCORRECT}(\pi,L) = |L|$.

On the other hand, the optimal value is 2, which is attained by any permutation that has cue n as the first cue. This yields a performance ratio for GREEDY CUE PERMUTATION of at least $|L|/2$. The lower bound $n/2$ is obtained by observing that $|L| = n$. ■

We conclude this section by examining the performance of GREEDY CUE PERMUTATION on subproblems, that is, when the instances are not arbitrary but meet certain constraints. It plainly arises from the proof of Theorem 11 that the lower bound holds under restrictions of the instances similar to those considered in Sections 3 and 4.

Corollary 12 *The lower bound* $\max\{n/2,|L|/2\}$ *for the performance ratio of* GREEDY CUE PERMUTATION *holds even when the instances satisfy any (or all) of the following constraints:*

1. *Each element of B contains at most two* 1*s.*
2. *The set L is smaller than the set B.*
3. *L is irreflexive.*
4. *L is a subset of some partial order.*
5. *L is a subset of some total order.*

6. Lexicographic Strategies With Cue Inversion

While in the previous sections the problem was to optimize lexicographic strategies by permuting the cues, we now introduce an additional degree of freedom for building lexicographic strategies. Here, the method of construction is allowed not only to permute but also to invert cues. A *cue*

Algorithm 2 GREEDY CUE INVERSION

Input: a set $B \subseteq \{0,1\}^n$ and a set $L \subseteq B \times B$
Output: a cue inversion q for n cues
  for $i = 1,\ldots,n$ **do**
    if $|\{\langle a,b\rangle \in L : a_i < b_i\}| \geq |\{\langle a,b\rangle \in L : a_i > b_i\}|$ **then**
      $q(i) := 0$
    else
      $q(i) := 1$
    end if
    $L := L \setminus \{\langle a,b\rangle : a_i \neq b_i\}$
  end for.

inversion is a mapping $q : \{1,\ldots,n\} \to \{0,1\}$, where n is the number of cues. It uniquely defines a function $\overline{q} : \{0,1\}^n \to \{0,1\}^n$ such that for every $a \in \{0,1\}^n$,

$$\overline{q}(a_i) \quad = \quad \begin{cases} a_i & \text{if } q(i) = 0, \\ 1 - a_i & \text{otherwise.} \end{cases}$$

In other words, a value of $q(i) = 1$ indicates that the i-th position of every Boolean vector a is to be inverted, whereas the cues with $q(i) = 0$ are left unchanged by $\overline{q}$. As the meaning is clear, we shall use q also to denote $\overline{q}$. Given a set $B \subseteq \{0,1\}^n$, the *lexicographic strategy under cue inversion* q is the function $S^q : B \times B \to \{\text{“}<\text{”}, \text{“}=\text{”}, \text{“}>\text{”}\}$ with

$$S^q(a,b) \quad = \quad S(q(a), q(b)).$$

Combining permutation and inversion, we obtain the *lexicographic strategy under cue permutation* π *and cue inversion* q denoted by S^q_π and defined as

$$S^q_\pi(a,b) \quad = \quad S(\pi(q(a)), \pi(q(b))).$$

In particular, we require that the cue inversion is applied before the permutation.

A simple greedy method for inverting the cues is described as Algorithm 2. The idea is to pass through the cues and to select either the cue or its inverse, depending on which makes a larger number of correct inferences. The pairs that are distinguished by this cue are then removed. It is evident that GREEDY CUE INVERSION runs in polynomial time. We show that the cue inversion returned by this algorithm yields a number of correct inferences that is at least half the maximum over all cue inversions and permutations.

Theorem 13 *The algorithm* GREEDY CUE INVERSION *always returns a cue inversion q such that S^q is correct on at least $\mathrm{opt}(L)/2$ pairs, where $\mathrm{opt}(L)$ is the maximum number of correct inferences achievable by the lexicographic strategy under any cue permutation and any cue inversion.*

Proof Let L_i be the set of pairs that the algorithm removes from L in round i of the for-loop and let L_{n+1} be the set of pairs that remains after completion of the last round. Clearly, $L_1,\ldots,L_{n+1}$ is a partition of L. Obviously, by the construction of q, S^q is correct on at least half of each L_i, for $i = 1,\ldots,n$. Further, it is correct on all of L_{n+1}, as this set consists solely of identical pairs. Thus, S^q

correctly distinguishes at least half of all pairs in L. Since $\mathrm{opt}(L) \leq |L|$, it follows that S^q is correct on at least $\mathrm{opt}(L)/2$ pairs. ■

One remarkable aspect of this algorithm is the fact that it retains the order of the cues, while its performance guarantee is valid even over all cue permutations. It seems, at first glance, that the method of cue inversion leads much easier to a good performance guarantee than the permutation of the cues. However, the result of Theorem 13 cannot directly compared with those of the previous sections, as these apply to the problem of minimizing the number of incorrect inferences, whereas here we are concerned with the maximization of the number of correct inferences. A constant performance ratio for the one problem does not necessarily imply a constant performance ratio for the other, as can easily be seen. Assume, for instance, that the maximum number of correct inferences is $|L|-1$. Then the algorithm that is correct on exactly $\lceil |L|/2 \rceil$ pairs has a constant performance ratio for the maximization problem, while with regard to the minimization problem its performance ratio grows linearly in $|L|$.

7. Sample Complexity for Learning Lexicographic Strategies

A central notion for characterizing the sample complexity of a learning problem is the VC dimension (Vapnik and Chervonenkis, 1971; Anthony and Bartlett, 1999). In the following, we calculate the VC dimension of lexicographic strategies. The definition of the VC dimension relies on the notion of shattering. A class $\mathcal{F}$ of Boolean functions is said *to shatter* a set $L \subseteq \{0,1\}^n$ if $\mathcal{F}$ induces every dichotomy of L, that is, if for every (L_0, L_1) such that $L_0 \cap L_1 = \emptyset$ and $L_0 \cup L_1 = L$, there is some function $f \in \mathcal{F}$ satisfying $f(L_0) \subseteq \{0\}$ and $f(L_1) \subseteq \{1\}$. The *Vapnik-Chervonenkis (VC) dimension* of a class $\mathcal{F}$ of Boolean functions is the cardinality of the largest set that is shattered by $\mathcal{F}$.

We recall from Section 2.2 that we identify the lexicographic strategy S with a Boolean function $f : \{0,1\}^{2n} \to \{0,1\}$ such that for every $\langle a, b\rangle \in \{0,1\}^{2n}$,

$$f(a,b) = 1 \quad \text{if and only if} \quad S(a,b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\}.$$

In this sense, we can investigate the VC dimension of the function class

$$\mathcal{S}_n \;=\; \{S^q_\pi : \pi \text{ is a permutation and } q \text{ an inversion of } n \text{ cues}\},$$

that is, we ask what is the largest cardinality of a set L of pairs that is shattered by the lexicographic strategy under all possible cue permutations and inversions.

It is evident from the definition that the VC dimension of a finite function class $\mathcal{F}$ cannot be larger than $\log|\mathcal{F}|$. Since the number of permutations is equal to $n!$ and the number of inversions is equal to 2^n, it follows that the VC dimension of $\mathcal{S}_n$ is not larger than $n + n\log n$. We show, however, that this VC dimension is linear. Moreover, we provide the exact value.

Theorem 14 *The VC dimension of the class $\mathcal{S}_n$ of lexicographic strategies is equal to n.*

Proof We first establish n as upper bound. Given a cue inversion q, consider the lexicographic strategy $S^q \in \mathcal{S}_n$ (that is, the strategy S^q_π where π is the identity function). We claim that every $a, b \in \{0,1\}^n$ satisfies

$$S^q(a,b) \in \{\text{“}<\text{”}, \text{“}=\text{”}\} \quad \text{if and only if} \quad \sum_{i=1}^{n} (-1)^{q(i)} 2^{n+1-i}(b_i - a_i) \;\geq\; -1. \tag{4}$$

To show this, we consider the absolute value of first term on the left-hand side of the inequality, where $i = 1$, that is,

$$|2^n(b_1 - a_1)|. \tag{5}$$

If $a_1 \neq b_1$, the value of (5) is 2^n, whereas the absolute value of the remaining sum is not larger than $2^n - 2$. Then, the inequality in (4) is satisfied if and only if $\overline{q}(a_1) < \overline{q}(b_1)$. On the other hand, if $a_1 = b_1$, the term (5) is equal to 0, and the validity of the equivalence (4) follows by induction.

Obviously, by permuting the coefficients, every lexicographic strategy $S^q_\pi \in \mathcal{S}_n$ can be written as an inequality such as in (4). Such inequalities are evaluated by Boolean linear threshold functions. A Boolean linear threshold function $f : \{0,1\}^n \to \{0,1\}$ is a function for which there exist real numbers $w_1, \ldots, w_n$ and t (the parameters of this function class) such that for every $z \in \{0,1\}^n$,

$$f(z) = 1 \quad \text{if and only if} \quad w_1 z_1 + \cdots + w_n z_n \geq t.$$

It follows that every $S^q_\pi \in \mathcal{S}_n$ can be expressed as a Boolean linear threshold function with input variables $(y_1 - x_1), \ldots, (y_n - x_n)$ and a fixed parameter $t = -1$.

Therefore, every set $L \subseteq \{0,1\}^{2n}$ that can be shattered by $\mathcal{S}_n$ is also shattered by this class of linear threshold functions. The class of linear threshold functions in n variables with n parameters (that is, where t is fixed) is known to have VC dimension equal to n (see, e.g., Anthony and Bartlett, 1999). Thus, the VC dimension of $\mathcal{S}_n$ does not exceed n.

For deriving the lower bound, we show that the set $L \subseteq \{0,1\}^{2n}$ defined as

$$L \quad = \quad \{\langle 1_i, 1\rangle : i = 1, \ldots, n\},$$

where 1 is the vector with a 1 in every position and 1_i has a 0 in position i and 1 elsewhere, is shattered by $\mathcal{S}_n$.

Let (L_0, L_1) be an arbitrary dichotomy of L. Define the cue inversion $q : \{1, \ldots, n\} \to \{0,1\}$ such that $q(i) = 0$ if and only if $\langle 1_i, 1\rangle \in L_1$. Obviously then, the lexicographic strategy $\mathcal{S}^q$ (without permuting the cues) yields a correct comparison for every pair in L_1, while the pairs from L_0 are inferred incorrectly. Thus, the dichotomy (L_0, L_1) is induced by S^q. ■

The lower bound in the previous result was obtained by choosing a suitable cue inversion and leaving the order of the cues unchanged. We can also obtain an almost optimal lower bound when the cues are not allowed to be inverted but only permuted. In fact, the $(n-1)$-element set

$$L \quad = \quad \{\langle 1_1, 1_i\rangle : i = 2, \ldots, n\}$$

can be shattered as follows. Given the dichotomy (L_0, L_1), we define the permutation π such that for $i = 2, \ldots, n$, $\pi(1) < \pi(i)$ if and only if $\langle 1_1, 1_i\rangle \in L_1$. Obviously, the dichotomy (L_0, L_1) is induced by S_π.

It is easy to see that there are values of n for which this lower bound of $n-1$ cannot be improved. For $n = 1, 2$, and 3, the number of permutations of n elements is $1, 2$, and 6, respectively; to shatter sets of these cardinalities, however, requires $2, 4$, and 8 functions.

8. Open Questions

In the following we summarize the major open questions that arise from this work hoping that they might provide fertile soil for future research. The main result of Section 3 is the NP-completeness

of the decision problem LEXICOGRAPHIC STRATEGY. In that section, we have established further that the problem remains NP-complete under several restrictions. Moreover, one of the subproblems originating from such restrictions was shown to be efficiently solvable. Probably, the restrictions considered there may not be those that are "natural", that is, met in practice. It is therefore reasonable to study more subproblems and to delineate the intractable ones from those that can be solved efficiently.

- What are natural restrictions for LEXICOGRAPHIC STRATEGY under which the problem is NP-complete or efficiently solvable?

Of course, similar considerations are appropriate for MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY. In Section 4 we obtained a lower bound for the performance ratio that is still valid for various subproblems. A promising task is, therefore, to find restrictions relevant in practice under which the problem has a constant performance ratio.

- What are natural restrictions for MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY under which the problem belongs to APX?

Work by Raz and Safra (1997) implies that MINIMUM HITTING SET cannot be approximated in polynomial time to within some factor that grows logarithmically in $|C|$, the number of subsets. The reduction defined in the proof of Theorem 7 does not seem to allow to exploit this fact.

- Does MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY have a lower bound on the performance ratio for polynomial-time algorithms that is not bounded by some constant?

The results in Sections 4 and 5 have left a gap. While we have shown that there cannot be a polynomial-time algorithm for MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY with a performance ratio bounded by some constant (if P $\neq$ NP), the algorithm GREEDY CUE PERMUTATION has a lower bound of $\max\{n/2, |L|/2\}$.

- Are there polynomial-time algorithms for MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY that have a better performance ratio than GREEDY CUE PERMUTATION?

The algorithm GREEDY CUE PERMUTATION is a simple and obvious heuristic that has not been studied before in the context of lexicographic strategies. In Section 5 we have derived tight bounds on the performance ratio of this algorithm. Various other procedures have been studied in the literature and become known as fast and frugal heuristics, but nothing seems to have been proven about their performance ratio.

- Which are the performance ratios of other (fast and frugal) heuristics for lexicographic strategies?

In Section 6 we have introduced cue inversion as an additional feature to build lexicographic strategies. The algorithm GREEDY CUE INVERSION was shown to approximate the maximum number of correct inferences to within a constant factor. While the problems of minimizing the number of incorrect inferences and maximizing the number of correct inferences give rise to equivalent decision problems, there might well be a difference with regard to the approximation problem. There seems to be no immediate way to derive a lower bound for the maximization problem from a lower bound for the minimization problem. Thus, similar questions as considered here can be raised for the problem MAXIMUM CORRECT LEXICOGRAPHIC STRATEGY which is defined analogously.

- Which is the performance ratio of polynomial-time algorithms for approximating MAXIMUM CORRECT LEXICOGRAPHIC STRATEGY?

While this question is meant to consider only cue permutations and not inversions for constructing lexicographic strategies, the objective of minimization is combined with both these features in a second approximation problem emerging from Section 6.

- Which is the performance ratio of polynomial-time algorithms for approximating MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY UNDER CUE PERMUTATIONS AND CUE INVERSIONS?

We can ask further what happens if the problems studied here are generalized in a certain way. One obvious possibility of generalizing is to allow cues that have more than two values. It is evident that the reductions provided in Sections 3 and 4 remain valid also in this multiple-valued case. In other words, the problem with binary cues is a subproblem of the problem with multiple-valued cues. Hence, NP-completeness and the lower bound for the approximability hold for learning lexicographic strategies on multiple-valued cues, too. Moreover, we observe that the algorithm GREEDY CUE PERMUTATION and the proof of the upper bound on its performance ratio (Theorem 9) do not make use of the two-valuedness of the cues. Thus, this algorithm has the claimed approximation property for multiple-valued cues as well. One could also generalize lexicographic strategies to the effect that more than two outcomes, correct or incorrect, of a lexicographic comparison are possible. The results of this article do not seem to yield a statement for such cases in general.

9. Conclusions

Computational problems that arise in learning lexicographic strategies from examples are the topic of this article. In particular, we considered the model of agnostic PAC learning. We have introduced the minimizing disagreement problem LEXICOGRAPHIC STRATEGY and shown that it is NP-complete. Thus, it has become very unlikely that lexicographic strategies can be efficiently learned. This statement was strengthened by our proving that the optimization problem MINIMUM INCORRECT LEXICOGRAPHIC STRATEGY cannot be approximated in polynomial time to within any constant factor.

These results answer a question raised by psychological research into models of bounded rationality: How accurate are fast and frugal heuristics? We have shown that no fast, that is, polynomial-time, algorithm can compute the optimum and, moreover, not even approximate it well, under the widely accepted assumption that $P \neq NP$.

This answers also a second question concerning a specific fast and frugal heuristic: How accurate is TTB? We have introduced a greedy algorithm that provably performs better than TTB. In particular, we have shown that the greedy method always finds accurate solutions when they exist, whereas this is not the case with TTB. Tight bounds for the factor with which the greedy method approximates the optimum have also been obtained.

The lower bounds derived in this article have mostly been shown to hold even for subproblems obtained from various restrictions. We interpret this as revealing to a high degree that lexicographic strategies cannot be learned efficiently and that it might be very difficult to find satisfactory algorithms.

For the learning of lexicographic strategies using cue inversions we have provided a simple and efficient algorithm that approximates the maximum number of correct inferences to within a con-

stant factor. Thus, it seems that cue inversions lead much easier to good performance bounds than cue permutations. However, one cannot directly compare a bound for the maximization problem with a bound for the minimization problem. This result should more be considered as a stimulating impetus for further research.

We have calculated the exact values of the VC dimension of lexicographic strategies. This result is one of the few examples where the VC dimension of a function class has been determined precisely.

While we have already presented in the previous section a couple of formal open questions for theoretical investigation, a challenge to experimental research is also given by this article: to study the relevance of the greedy method as a model for bounded rationality in psychology.

Acknowledgments

We thank the anonymous reviewers for the detailed and helpful comments that helped to improve the article.

Parts of this research were carried out while M. S. was with the Ruhr-Universität Bochum, and while he was visiting the Max Planck Institute for Psychological Research, Munich, the Max Planck Institute for Human Development, Berlin, and the Pädagogische Hochschule Ludwigsburg, all in Germany. The authors are grateful to these institutions for their support.

References

E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147:181–210, 1995.

E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, 1999.

G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Problems and Their Approximability Properties*. Springer-Verlag, Berlin, 1999.

G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980.

R. Bar-Yehuda and S. Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.

M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 294–304. ACM Press, New York, NY, 1993.

Arndt Bröder. Assessing the empirical validity of the "take-the-best" heuristic as a model of human probabilistic inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26:1332–1346, 2000.

Arndt Bröder. Take the best, Dawes' rule, and compensatory decision strategies: A regression-based classification method. *Quality & Quantity*, 36:219–238, 2002.

Arndt Bröder and Stefanie Schiffer. Take the best versus simultaneous feature matching: Probabilistic inferences from memory and effects of representation format. *Journal of Experimental Psychology: General*, 132:277–293, 2003.

Seth Bullock and Peter M. Todd. Made to measure: Ecological rationality in structured environments. *Minds and Machines*, 9:497–541, 1999.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA, 1979.

Gerd Gigerenzer and Daniel G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103:650–669, 1996.

Gerd Gigerenzer, Ulrich Hoffrage, and Heinz Kleinbölting. Probabilistic mental models: A Brunswikian theory of confidence. *Psychological Review*, 98:506–528, 1991.

Gerd Gigerenzer, Peter M. Todd, and the ABC Research Group. *Simple Heuristics That Make Us Smart*. Oxford University Press, New York, NY, 1999.

Russell Greiner. The complexity of revising logic programs. *The Journal of Logic Programming*, 40:273–298, 1999.

Russell Greiner and Pekka Orponen. Probably approximately optimal satisficing strategies. *Artificial Intelligence*, 82:21–44, 1996.

Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11:555–556, 1982.

Klaus-U. Höffgen, Hans-U. Simon, and Kevin S. Van Horn. Robust trainability of single neurons. *Journal of Computer and System Sciences*, 50:114–125, 1995.

Robin M. Hogarth and Natalia Karelaia. "Take-the-best" and other simple strategies: Why and when they work "well" in binary choice. DEE Working Paper 709, Universitat Pompeu Fabra, Barcelona, October 2003.

Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17:115–141, 1994.

Michael D. Lee and Tarrant D. R. Cummins. Evidence accumulation in decision making: Unifying the "take the best" and the "rational" models. *Psychonomic Bulletin & Review*, 11:343–352, 2004.

Laura Martignon and Ulrich Hoffrage. Why does one-reason decision making work? A case study in ecological rationality. In Gigerenzer, G., Todd, P. M., and the ABC Research Group, *Simple Heuristics That Make Us Smart*, pages 119–140. Oxford University Press, New York, NY, 1999.

Laura Martignon and Ulrich Hoffrage. Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory and Decision*, 52:29–71, 2002.

Laura Martignon and Michael Schmitt. Simplicity and robustness of fast and frugal heuristics. *Minds and Machines*, 9:565–593, 1999.

Stefani Nellen. The use of the "take the best" heuristic under different conditions, modeled with ACT-R. In F. Detje, D. Dörner, and H. Schaub, editors, *Proceedings of the Fifth International Conference on Cognitive Modeling*, pages 171–176, Universitätsverlag Bamberg, Bamberg, 2003.

Ben R. Newell and David R. Shanks. Take the best or look at the rest? Factors influencing "One-Reason" decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29:53–65, 2003.

Ben R. Newell, Nicola J. Weston, and David R. Shanks. Empirical tests of a fast-and-frugal heuristic: Not everyone "takes-the-best". *Organizational Behavior and Human Decision Processes*, 91: 82–96, 2003.

Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM Press, New York, NY, 1997.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

Stuart Russell and Eric Wefald. *Do The Right Thing: Studies in Limited Rationality*. MIT Press, Cambridge, MA, 1991.

Michael Schmitt and Laura Martignon. Complexity of lexicographic strategies on binary cues. Preprint, 1999.

Michael Schmitt and Laura Martignon. On the accuracy of bounded rationality: How far from optimal is fast and frugal? In Yair Weiss, Bernhard Schölkopf, and John C. Platt, editors, *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006. To appear.

Herbert A. Simon. *Models of Bounded Rationality, Volume 2*. MIT Press, Cambridge, MA, 1982.

Herbert A. Simon and Joseph B. Kadane. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence*, 6:235–247, 1975.

Herbert A. Simon and Joseph B. Kadane. Problems of computational complexity in artificial intelligence. In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 281–299. Academic Press, New York, NY, 1976.

Steven S. Skiena. *The Algorithm Design Manual*. Springer-Verlag, New York, NY, 1997.

D. W. Slegers, G. L. Brake, and M. E. Doherty. Probabilistic mental models with continuous predictors. *Organizational Behavior and Human Decision Processes*, 81:98–114, 2000.

Peter M. Todd and Anja Dieckmann. Heuristics for ordering cue search in decision making. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1393–1400. MIT Press, Cambridge, MA, 2005.

Peter M. Todd and Gerd Gigerenzer. Précis of "Simple Heuristics That Make Us Smart". *Behavioral and Brain Sciences*, 23:727–741, 2000.

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Journal of Machine Learning Research 7 (2006) 85–115 Submitted 5/05; Revised 10/05; Published 1/06

Generalized Bradley-Terry Models and Multi-Class Probability Estimates

Tzu-Kuo Huang R93002@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Ruby C. Weng CHWENG@NCCU.EDU.TW
Department of Statistics
National Chengchi University
Taipei 116, Taiwan

Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: Greg Ridgeway

Abstract

The Bradley-Terry model for obtaining individual skill from paired comparisons has been popular in many areas. In machine learning, this model is related to multi-class probability estimates by coupling all pairwise classification results. Error correcting output codes (ECOC) are a general framework to decompose a multi-class problem to several binary problems. To obtain probability estimates under this framework, this paper introduces a generalized Bradley-Terry model in which paired individual comparisons are extended to paired team comparisons. We propose a simple algorithm with convergence proofs to solve the model and obtain individual skill. Experiments on synthetic and real data demonstrate that the algorithm is useful for obtaining multi-class probability estimates. Moreover, we discuss four extensions of the proposed model: 1) weighted individual skill, 2) home-field advantage, 3) ties, and 4) comparisons with more than two teams.

Keywords: Bradley-Terry model, probability estimates, error correcting output codes, support vector machines

1. Introduction

The Bradley-Terry model (Bradley and Terry, 1952) for paired comparisons has been broadly applied in many areas such as statistics, sports, and machine learning. It considers a set of k individuals for which

$$P(\text{individual } i \text{ beats individual } j) = \frac{p_i}{p_i + p_j}, \tag{1}$$

and $p_i > 0$ is the overall skill of individual i. Suppose that the outcomes of all comparisons are independent and denote r_{ij} as the number of times that i beats j. Then the negative

log-likelihood takes the form

$$l(\mathbf{p}) = -\sum_{i<j}\left(r_{ij}\log\frac{p_i}{p_i+p_j} + r_{ji}\log\frac{p_j}{p_i+p_j}\right). \tag{2}$$

Since $l(\mathbf{p}) = l(\alpha\mathbf{p})$ for any $\alpha > 0$, $l(\mathbf{p})$ is scale invariant. Therefore, it is convenient to assume that $\sum_{i=1}^k p_i = 1$ for the sake of identifiability. One can then estimate p_i by

$$\begin{aligned} \min_{\mathbf{p}} \quad & l(\mathbf{p}) \\ \text{subject to} \quad & 0 \le p_j, j = 1,\ldots,k, \sum_{j=1}^k p_j = 1. \end{aligned} \tag{3}$$

This approach dates back to Zermelo (1929) and has been extended to more general settings. For instance, in sports scenario, extensions to account for the home-field advantage and ties have been proposed. Some reviews are, for example, (David, 1988; Davidson and Farquhar, 1976; Hunter, 2004; Simons and Yao, 1999). The solution of (3) can be solved by a simple iterative procedure:

Algorithm 1:

1. Start with any initial $p_j^0 > 0$, $j = 1,\ldots,k$.
2. Repeat $(t = 0, 1, \ldots)$
   (a) Let $s = (t \mod k) + 1$. Define

$$\mathbf{p}^{t+1} \equiv \left[p_1^t, \ldots, p_{s-1}^t, \frac{\sum_{i:i\neq s} r_{si}}{\sum_{i:i\neq s}\frac{r_{si}+r_{is}}{p_s^t+p_i^t}}, p_{s+1}^t, \ldots, p_k^t\right]^T. \tag{4}$$

   (b) Normalize $\mathbf{p}^{t+1}$.

   until $\partial l(\mathbf{p}^t)/\partial p_j = 0, j = 1,\ldots,k$ are satisfied.

This algorithm is so simple that there is no need to use sophisticated optimization techniques. If r_{ij} $\forall i, j$ satisfy some mild conditions, Algorithm 1 globally converges to the unique minimum of (3). A systematic study on the convergence of Algorithm 1 is in Hunter (2004).

An earlier work (Hastie and Tibshirani, 1998) in statistics and machine learning considered the problem of obtaining multi-class probability estimates by coupling results from pairwise comparisons. Assume

$$\bar{r}_{ij} \equiv P(\mathbf{x} \text{ in class } i \mid \mathbf{x} \text{ in class } i \text{ or } j)$$

is known. This work estimates $p_i = P(\mathbf{x} \text{ in class } i)$ by minimizing the (weighted) Kullback-Leibler (KL) distance between $\bar{r}_{ij}$ and $\mu_{ij} \equiv p_i/(p_i + p_j)$:

$$\begin{aligned} \min_{\mathbf{p}} \quad & \sum_{i<j} n_{ij} \left(\bar{r}_{ij} \log \frac{\bar{r}_{ij}}{\mu_{ij}} + \bar{r}_{ji} \log \frac{\bar{r}_{ji}}{\mu_{ji}} \right) \\ \text{subject to} \quad & 0 \le p_j, j = 1, \ldots, k, \sum_{j=1}^{k} p_j = 1, \end{aligned} \tag{5}$$

where n_{ij} is the number of training data in class i or j. By defining $r_{ij} \equiv n_{ij}\bar{r}_{ij}$ and removing constant terms, (5) reduces to the same form as (2), and hence Algorithm 1 can be used to find $\mathbf{p}$. Although one might interpret this as a Bradley-Terry model by treating classes as individuals and r_{ij} as the number that the ith class beats the jth class, it is not indeed. First, r_{ij} (now defined as $n_{ij}\bar{r}_{ij}$) may not be an integer any more. Secondly, r_{ij} are dependent as they share the same training set. However, the closeness between the two motivates us to propose more general models in this paper.

The above approach involving comparisons for each pair of classes is referred to as the "one-against-one" setting in multi-class classification. It is a special case of the framework *error correcting output codes* (ECOC) to decompose a multi-class problem into a number of binary problems (Dietterich and Bakiri, 1995; Allwein et al., 2001). Some classification techniques are two-class based, so this framework extends them to multi-class scenarios. Zadrozny (2002) generalizes the results in (Hastie and Tibshirani, 1998) to obtain probability estimates under ECOC settings. The author proposed an algorithm analogous to Algorithm 1 and demonstrated some experimental results. However, the convergence issue was not discussed. Though the author intended to minimize the KL distance as Hastie and Tibshirani (1998) did, in Section 4.2 we show that their algorithm may not converge to a point with the smallest KL distance.

Motivated from multi-class classification with ECOC settings, this paper presents a generalized Bradley-Terry model where each competition is between two teams (two disjoint subsets of subjects) and team size/members can vary from competition to competition. Then from the outcomes of all comparisons, we fit this general model to estimate the individual skill. Here we propose a simple iterative method to solve the generalized model. The convergence is proved under mild conditions.

The proposed model has some potential applications. For example, in tennis or badminton, if a player participates in many singles and doubles, this general model can combine all outcomes to yield the estimated skill of all individuals. More importantly, for multi-class problems by combining binary classification results, we can also minimize the KL distance and obtain the same optimization problem. Hence the proposed iterative method can be directly applied to obtain the probability estimate under ECOC settings.

This paper is organized as follows. Section 2 introduces a generalized Bradley-Terry model and a simple algorithm to maximize the log-likelihood. The convergence of the proposed algorithm is in Section 3. Section 4 discusses multi-class probability estimates and experiments are in Sections 5 and 6. In Section 7 we discuss four extensions of the proposed model: 1) weighted individual skill, 2) home-field advantage, 3) ties, and 4) comparisons with more than two teams. Discussion and conclusions are in Section 8. A short and

preliminary version of this paper appeared in an earlier conference NIPS 2004 (Huang et al., 2005).[1]

2. Generalized Bradley-Terry Model

In this section we study a generalized Bradley-Terry model for approximating individual skill. Consider a group of k individuals: $\{1,\dots,k\}$. Each time two disjoint subsets I_i^+ and I_i^- form teams for a series of games and $r_i \geq 0$ ($r_i' \geq 0$) is the number of times that I_i^+ beats I_i^- (I_i^- beats I_i^+). Thus, we have $I_i \subset \{1,\dots,k\}, i = 1,\dots,m$ so that

$$I_i = I_i^+ \cup I_i^-, \qquad I_i^+ \neq \emptyset, I_i^- \neq \emptyset, \text{ and } I_i^+ \cap I_i^- = \emptyset.$$

If the game is designed so that each member is equally important, we can assume that a team's skill is the sum of all its members'. This leads to the following model:

$$P(I_i^+ \text{ beats } I_i^-) = \frac{\sum_{j\in I_i^+} p_j}{\sum_{j\in I_i} p_j}.$$

If the outcomes of all comparisons are independent, then estimated individual skill can be obtained by defining

$$q_i \equiv \sum_{j\in I_i} p_j, \qquad q_i^+ \equiv \sum_{j\in I_i^+} p_j, \qquad q_i^- \equiv \sum_{j\in I_i^-} p_j$$

and minimizing the negative log-likelihood

$$\begin{aligned} \min_{\mathbf{p}} \quad & l(\mathbf{p}) = -\sum_{i=1}^{m} \left(r_i \log(q_i^+/q_i) + r_i' \log(q_i^-/q_i) \right) \\ \text{subject to} \quad & \sum_{j=1}^{k} p_j = 1, 0 \leq p_j, j = 1,\dots,k. \end{aligned} \tag{6}$$

Note that (6) reduces to (3) in the pairwise approach, where $m = k(k-1)/2$ and $I_i, i = 1,\dots,m$ are as the following:

I_i^+	I_i^-	r_i	r_i'
$\{1\}$	$\{2\}$	r_{12}	r_{21}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\{1\}$	$\{k\}$	r_{1k}	r_{k1}
$\{2\}$	$\{3\}$	r_{23}	r_{32}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\{k-1\}$	$\{k\}$	$r_{k-1,k}$	$r_{k,k-1}$

In the rest of this section we discuss how to solve the optimization problem (6).

1. Programs used are at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/libsvm-errorcode`.

2.1 A Simple Procedure to Maximize the Likelihood

The difficulty of solving (6) over (3) is that now $l(\mathbf{p})$ is expressed in terms of q_i^+, q_i^-, q_i but the real variable is $\mathbf{p}$. We propose the following algorithm to solve (6).

Algorithm 2:

1. Start with initial $p_j^0 > 0, j = 1, \ldots, k$ and obtain corresponding $q_i^{0,+}, q_i^{0,-}, q_i^0, i = 1, \ldots, m$.
2. Repeat $(t = 0, 1, \ldots)$
   (a) Let $s = (t \mod k) + 1$. Define $\mathbf{p}^{t+1}$ by $p_j^{t+1} = p_j^t,\ \forall j \neq s$, and

$$p_s^{t+1} = \frac{\sum_{i:s\in I_i^+} \frac{r_i}{q_i^{t,+}} + \sum_{i:s\in I_i^-} \frac{r_i'}{q_i^{t,-}}}{\sum_{i:s\in I_i} \frac{r_i+r_i'}{q_i^t}} p_s^t. \tag{7}$$

   (b) Normalize $\mathbf{p}^{t+1}$.
   (c) Update $q_i^{t,+}, q_i^{t,-}, q_i^t$ to $q_i^{t+1,+}, q_i^{t+1,-}, q_i^{t+1}, i = 1, \ldots, m$.

   until $\partial l(\mathbf{p}^t)/\partial p_j = 0, j = 1, \ldots, k$ are satisfied.

The gradient of $l(\mathbf{p})$, used in the stopping criterion, is:

$$\begin{aligned}\frac{\partial l(\mathbf{p})}{\partial p_s} &= -\sum_{i=1}^{m}\left(r_i\frac{\partial \log q_i^+}{\partial p_s} + r_i'\frac{\partial \log q_i^-}{\partial p_s} - (r_i + r_i')\frac{\partial \log q_i}{\partial p_s}\right) \\ &= -\sum_{i:s\in I_i^+}\frac{r_i}{q_i^+} - \sum_{i:s\in I_i^-}\frac{r_i'}{q_i^-} + \sum_{i:s\in I_i}\frac{r_i + r_i'}{q_i}, \qquad s = 1, \ldots, k. \end{aligned} \tag{8}$$

In Algorithm 2, for the multiplicative factor in (7) to be well defined (i.e., non-zero denominator), we need Assumption 1, which will be discussed in Section 3. Eq. (7) is a simple fixed-point type update; in each iteration, only one component (i.e., p_s^t) is modified while the others remain the same. If we apply the updating rule (7) to the pairwise model,

$$p_s^{t+1} = \frac{\sum_{i:s<i} \frac{r_{si}}{p_s^t} + \sum_{i:i<s} \frac{r_{si}}{p_s^t}}{\sum_{i:s<i} \frac{r_{si}+r_{is}}{p_s^t+p_i^t} + \sum_{i:i<s} \frac{r_{is}+r_{si}}{p_s^t+p_i^t}} p_s^t = \frac{\sum_{i:i\neq s} r_{si}}{\sum_{i:i\neq s} \frac{r_{si}+r_{is}}{p_s^t+p_i^t}}$$

reduces to (4).

The updating rule (7) is motivated from using a descent direction to strictly decrease $l(\mathbf{p})$: If $\partial l(\mathbf{p}^t)/\partial p_s \neq 0$ and $p_s^t > 0$, then under suitable assumptions on r_i, r_i',

$$\begin{aligned}\frac{\partial l(\mathbf{p}^t)}{\partial p_s}(p_s^{t+1} - p_s^t) &= \frac{\partial l(\mathbf{p}^t)}{\partial p_s}\left(\frac{\sum_{i:s\in I_i^+}\frac{r_i}{q_i^+} + \sum_{i:s\in I_i^-}\frac{r_i'}{q_i^-} - \sum_{i:s\in I_i}\frac{r_i+r_i'}{q_i}}{\sum_{i:s\in I_i}\frac{r_i+r_i'}{q_i^t}}\right)p_s^t \\ &= \left(-\left(\frac{\partial l(\mathbf{p}^t)}{\partial p_s}\right)^2 p_s^t\right) \Big/ \left(\sum_{i:s\in I_i}\frac{r_i+r_i'}{q_i^t}\right) < 0. \end{aligned} \tag{9}$$

Thus, $p_s^{t+1} - p_s^t$ is a descent direction in optimization terminology since a sufficiently small step along this direction guarantees the strict decrease of the function value. As now we take the whole direction without searching for the step size, more efforts are needed to prove the strict decrease in the following Theorem 1. However, (9) does hint that (7) is a reasonable update.

Theorem 1 *Let s be the index to be updated at $\mathbf{p}^t$. If*

1. $p_s^t > 0$,
2. $\partial l(\mathbf{p}^t)/\partial p_s \neq 0$, *and*
3. $\sum_{i:s\in I_i}(r_i + r_i') > 0$,

then

$$l(\mathbf{p}^{t+1}) < l(\mathbf{p}^t).$$

The proof is in Appendix A. Note that $\sum_{i:s\in I_i}(r_i + r_i') > 0$ is a reasonable assumption. It means that individual s participates in at least one game.

2.2 Other Methods to Maximize the Likelihood

We briefly discuss other methods to solve (6). For the original Bradley-Terry model, Hunter (2004) discussed how to transform (3) to a logistic regression form: Under certain assumptions,[2] the optimal $p_i > 0, \forall i$. Using this property and the constraints $p_j \geq 0, \sum_{j=1}^k p_j = 1$ of (3), we can reparameterize the function (2) by

$$p_s = \frac{e^{\beta_s}}{\sum_{j=1}^k e^{\beta_j}}, \tag{10}$$

and obtain

$$-\sum_{i<j}\left(r_{ij}\log\frac{1}{1+e^{\beta_j-\beta_i}} + r_{ji}\log\frac{e^{\beta_j-\beta_i}}{1+e^{\beta_j-\beta_i}}\right). \tag{11}$$

This is the negative log-likelihood of a logistic regression model. Hence, methods such as iterative weighted least squares (IWLS) (McCullagh and Nelder, 1990) can be used to fit

2. They will be described in the next section.

the model. In addition, $\boldsymbol{\beta}$ is now unrestricted, so (3) is transformed to an unconstrained optimization problem. Then conventional optimization techniques such as Newton or Quasi Newton can also be applied.

Now for the generalized model, (6) can still be re-parameterized as an unconstrained problem with the variable $\boldsymbol{\beta}$. However, the negative log-likelihood

$$-\sum_{i=1}^{m}\left(r_i \log \frac{\sum_{j\in I_i^+} e^{\beta_j}}{\sum_{j\in I_i} e^{\beta_j}} + r_i' \log \frac{\sum_{j\in I_i^-} e^{\beta_j}}{\sum_{j\in I_i} e^{\beta_j}}\right) \tag{12}$$

is not in a form similar to (11), so methods for logistic regression may not be used. Of course Newton or Quasi Newton is still applicable but their implementations are not simpler than Algorithm 2.

3. Convergence of Algorithm 2

Though Theorem 1 has shown the strict decrease of $l(\mathbf{p})$, we must further prove that Algorithm 2 converges to a stationary point of (6). Thus if $l(\mathbf{p})$ is convex, a global optimum is obtained. A vector $\mathbf{p}$ is a stationary (Karash-Kuhn-Tucker) point of (6) if and only if there is a scalar δ and two nonnegative vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\xi}$ such that

$$\begin{aligned}&\nabla f(\mathbf{p})_j = \delta + \lambda_j - \xi_j,\\ &\lambda_j p_j = 0, \xi_j(1-p_j) = 0, j = 1,\ldots,k.\end{aligned}$$

In the following we will prove that under certain conditions Algorithm 2 converges to a point satisfying

$$0 < p_j < 1, \nabla f(\mathbf{p})_j = 0, j = 1,\ldots,k. \tag{13}$$

That is, $\delta = \lambda_j = \xi_j = 0, \forall j$. Problem (6) is quite special as through the convergence proof of Algorithm 2 we show that its optimality condition reduces to (13), the condition without considering constraints. Furthermore, an interesting side-result is that from $\sum_{j=1}^{k} p_j = 1$ and (13), we obtain a point in R^k satisfying $(k+1)$ equations.

If Algorithm 2 stops in a finite number of iterations, then $\partial l(\mathbf{p})/\partial p_j = 0, j = 1,\ldots,k$, which means a stationary point of (6) is already obtained. Thus, we only need to handle the case where $\{\mathbf{p}^t\}$ is an infinite sequence. As $\{\mathbf{p}^t\}_{t=0}^{\infty}$ is in a compact set

$$\{\mathbf{p} \mid 0 \le p_s \le 1, \sum_{j=1}^{k} p_j = 1\},$$

there is at least one convergent subsequence. Assume that $\{\mathbf{p}^t\}, t \in K$ is any such sequence and it converges to $\mathbf{p}^*$. In the following we will show that $\partial l(\mathbf{p}^*)/\partial p_j = 0, j = 1,\ldots,k$.

To prove the convergence of a fixed-point type algorithm (i.e., Lyapunov's theorem), we require $p_s^* > 0, \forall s$. Then if $\partial l(\mathbf{p}^*)/\partial p_s \neq 0$ (i.e., $\mathbf{p}^*$ is not optimal), we can use (7) to find $\mathbf{p}^{*+1} \neq \mathbf{p}^*$, and, as a result of Theorem 1, $l(\mathbf{p}^{*+1}) < l(\mathbf{p}^*)$. This property further leads to a contradiction. To have $p_s^* > 0, \forall s$, for the original Bradley-Terry model, Ford (1957) and Hunter (2004) assume that for any pair of individuals s and j, there is a "path" from s to j; that is, $r_{s,s_1} > 0, r_{s_1,s_2} > 0, \ldots, r_{s_t,j} > 0$. The idea behind this assumption is simple:

Since $\sum_{r=1}^{k} p_r^* = 1$, there is at least one $p_j^* > 0$. If in certain games s beats s_1, s_1 beats $s_2, \ldots$, and s_t beats j, then p_s^*, the skill of individual s, should not be as bad as zero. For the generalized model, we make a similar assumption:

Assumption 1 *For any two different individuals s and j, there are $I_{s_0}, I_{s_1}, \ldots, I_{s_t}$, such that either*

1. $r_{s_0} > 0, r_{s_1} > 0, \ldots, r_{s_t} > 0$,
2. $I_{s_0}^+ = \{s\}; I_{s_r}^+ \subset I_{s_{r-1}}, r = 1, \ldots, t; j \in I_{s_t}^-$,

or

1. $r'_{s_0} > 0, r'_{s_1} > 0, \ldots, r'_{s_t} > 0$,
2. $I_{s_0}^- = \{s\}; I_{s_r}^- \subset I_{s_{r-1}}, r = 1, \ldots, t; j \in I_{s_t}^+$.

The idea is that if $p_j^* > 0$, and s beats $I_{s_0}^-$, a subset of I_{s_0} beats $I_{s_1}^-$, a subset of I_{s_1} beats $I_{s_2}^-, \ldots$, and a subset of $I_{s_{t-1}}$ beats $I_{s_t}^-$, which includes j, then p_s^* should not be zero. How this assumption is exactly used is in Appendix B for proving Lemma 2.

Assumption 1 is weaker than that made earlier in (Huang et al., 2005). However, even with the above explanation, this assumption seems to be very strong. Whether the generalized model satisfies Assumption 1 or not, an easy way to fulfill it is to add an additional term

$$-\mu \sum_{s=1}^{k} \log \left(\frac{p_s}{\sum_{j=1}^{k} p_j} \right) \tag{14}$$

to $l(\mathbf{p})$, where μ is a small positive number. That is, for each s, we make an $I_i = \{1, \ldots, k\}$ with $I_i^+ = \{s\}, r_i = \mu$, and $r'_i = 0$. As $\sum_{j=1}^{k} p_j = 1$ is one of the constraints, (14) reduces to $-\mu \sum_{s=1}^{k} \log p_s$, which is usually used as a barrier term in optimization to ensure that p_s does not go to zero.

An issue left in Section 2 is whether the multiplicative factor in (7) is well defined. With Assumption 1 and initial $p_j^0 > 0, j = 1, \ldots, k$, one can show by induction that $p_j^t > 0, \forall t$ and hence the denominator of (7) is never zero: If $p_j^t > 0$, Assumption 1 implies that there is some i such that $I_i^+ = \{j\}$ or $I_i^- = \{j\}$. Then either $\sum_{i:j \in I_i^+} r_i / q_i^{t,+}$ or $\sum_{i:j \in I_i^-} r'_i / q_i^{t,-}$ is positive. Thus, both numerator and denominator in the multiplicative factor are positive, and so is p_j^{t+1}.

The result $p_s^* > 0$ is proved in the following lemma.

Lemma 2 *If Assumption 1 holds, $p_s^* > 0, s = 1, \ldots, k$.*

The proof is in Appendix B.

As the convergence proof will use the strictly decreasing result, we note that Assumption 1 implies the condition $\sum_{i:s \in I_i} (r_i + r'_i) > 0, \forall s$, required by Theorem 1. Finally, the convergence is established:

Theorem 3 *Under Assumption 1, any convergent point of Algorithm 2 is a stationary point of* (6).

The proof is in Appendix C. Though r_i in the Bradley-Terry model is an integer indicating the number of times that team I_i^+ beats I_i^-, in the convergence proof we do not use such a property. Hence later for multi-class probability estimates, where r_i is a real number, the convergence result still holds.

Note that a stationary point may be only a saddle point. If (6) is a convex programming problem, then a stationary point is a global minimum. Unfortunately, $l(\mathbf{p})$ may not be convex, so it is not clear whether Algorithm 2 converges to a global minimum or not. The following theorem states that in some cases including the original Bradley-Terry model, any convergent point is a global minimum, and hence a maximum likelihood estimator:

Theorem 4 *Under Assumption 1, if*

1. $|I_i^+| = |I_i^-| = 1, i = 1, \ldots, m$ *or*

2. $|I_i| = k, i = 1, \ldots, m,$

then (6) *has a unique global minimum and Algorithm 2 globally converges to it.*

The proof is in Appendix D. The first case corresponds to the original Bradley-Terry model. Later we will show that under Assumption 1, the second case is related to "one-against-the rest" for multi-class probability estimates. Thus though the theorem seems to be rather restricted, it corresponds to useful situations.

4. Multi-class Probability Estimates

A classification problem is to train a model from data with known class labels and then predict labels of new data. Many classification methods are two-class based approaches and there are different ways to extend them for multi-class cases. Most existing studies focus on predicting class labels but not probability estimates. In this section, we discuss how the generalized Bradley-Terry model can be applied to multi-class probability estimates.

As mentioned in Section 1, there are various ways to decompose a multi-class problem into a number of binary classification problems. Among them, the most commonly used are "one-against-one" and "one-against-the rest." Recently Allwein et al. (2001) proposed a more general framework for the decomposition. Their idea, extended from that of Dietterich and Bakiri (1995), is to associate each class with a row of a $k \times m$ "coding matrix" with all entries from $\{-1, 0, +1\}$. Here m is the number of binary classification problems to be constructed. Each column of the matrix represents a comparison between classes with "-1" and "$+1$," ignoring classes with "0." Note that the classes with "-1" and "$+1$" correspond to our I_i^- and I_i^+, respectively. Then the binary learning method is run for each column of the matrix to obtain m binary decision rules. For a given example, one predicts the class label to be j if the results of the m binary decision rules are "closest" to labels of row j in the coding matrix. Since this coding method can correct errors made by some individual decision rules, it is referred to as *error correcting output codes* (ECOC). Clearly the commonly used "one-against-one" and "one-against-the rest" settings are special cases of this framework.

Given n_i, the number of training data with classes in $I_i = I_i^+ \cup I_i^-$, we assume here that for any given data $\mathbf{x}$,

$$\bar{r}_i = P(\mathbf{x} \text{ in classes of } I_i^+ \mid \mathbf{x} \text{ in classes of } I_i) \tag{15}$$

is available, and the task is to estimate $P(\mathbf{x} \text{ in class } s), s = 1, \ldots, k$. We minimize the (weighted) KL distance between $\bar{r}_i$ and q_i^+/q_i^- similar to (Hastie and Tibshirani, 1998):

$$\min_{\mathbf{p}} \sum_{i=1}^{m} n_i \left(\bar{r}_i \log \frac{\bar{r}_i}{(q_i^+/q_i)} + (1 - \bar{r}_i) \log \frac{1 - \bar{r}_i}{(q_i^-/q_i)} \right). \tag{16}$$

By defining

$$r_i \equiv n_i \bar{r}_i \text{ and } r_i' \equiv n_i(1 - \bar{r}_i), \tag{17}$$

and removing constant terms, (16) reduces to (6), the negative log-likelihood of the generalized Bradley-Terry model. It is explained in Section 1 that one cannot directly interpret this setting as a generalized Bradley-Terry model. Instead, we minimize the KL distance and obtain the same optimization problem.

We show in Section 5 that many practical "error correcting codes" have the same $|I_i|$, i.e., each binary problem involves the same number of classes. Thus, if data is balanced (all classes have about the same number of instances), then $n_1 \approx \cdots \approx n_m$ and we can remove n_i in (16) without affecting the minimization of $l(\mathbf{p})$. As a result, $r_i = \bar{r}_i$ and $r_i' = 1 - \bar{r}_i$.

In the rest of this section we discuss the case of "one-against-the rest" in detail and the earlier result in (Zadrozny, 2002).

4.1 Properties of the "One-against-the rest" Approach

For this approach, $m = k$ and $I_i, i = 1, \ldots, m$ are

I_i^+	I_i^-	r_i	r_i'
$\{1\}$	$\{2, \ldots, k\}$	r_1	$1 - r_1$
$\{2\}$	$\{1, 3, \ldots, k\}$	r_2	$1 - r_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\{k\}$	$\{1, \ldots, k-1\}$	r_k	$1 - r_k$

Clearly, $|I_i| = k \ \forall i$, so every game involves all classes. Then, $n_1 = \cdots = n_m =$ the total number of training data and the solution of (16) is not affected by n_i. This and (17) suggest that we can solve the problem by simply taking $n_i = 1$ and $r_i + r_i' = 1$, $\forall i$. Thus, (8) can be simplified as

$$\frac{\partial l(\mathbf{p})}{\partial p_s} = -\frac{r_s}{p_s} - \sum_{j:j \neq s} \frac{r_j'}{1 - p_j} + k.$$

Setting $\partial l(\mathbf{p})/\partial p_s = 0 \ \forall s$, we have

$$\frac{r_s}{p_s} - \frac{1 - r_s}{1 - p_s} = k - \sum_{j=1}^{k} \frac{r_j'}{1 - p_j}. \tag{18}$$

Since the right-hand side of (18) is the same for all s, we can denote it by δ. If $\delta = 0$, then $p_i = r_i$. This happens only if $\sum_{i=1}^k r_i = 1$. If $\delta \neq 0$, (18) implies

$$p_s = \frac{(1+\delta) - \sqrt{(1+\delta)^2 - 4r_s\delta}}{2\delta}. \tag{19}$$

In Appendix E we show that p_s defined in (19) satisfies $0 \leq p_s \leq 1$. Note that

$$((1+\delta) + \sqrt{(1+\delta)^2 - 4r_s\delta})/2\delta$$

also satisfies (18), but when $\delta < 0$, it is negative and when $\delta > 0$, it is greater than 1. Then the solution procedure is as the following:

If $\sum_{i=1}^k r_i = 1$,
    optimal $\mathbf{p} = [r_1, \ldots, r_k]^T$.
else
    find the root of $\sum_{s=1}^k \frac{(1+\delta)-\sqrt{(1+\delta)^2-4r_s\delta}}{2\delta} - 1 = 0$.
    optimal $p_s =$ (19).

If $\sum_{i=1}^k r_i = 1$, $\mathbf{p} = [r_1, \ldots, r_k]^T$ satisfies $\partial l(\mathbf{p})/\partial p_s = 0 \ \forall s$, and thus is the unique optimal solution in light of Theorem 4. For the else part, Appendix E proves that the above equation of δ has a unique root. Therefore, instead of using Algorithm 2, one can easily solve a one-variable nonlinear equation and obtain the optimal $\mathbf{p}$. This "one-against-the rest" setting is special as we can directly prove the existence of a solution satisfying $k+1$ equations: $\sum_{s=1}^k p_s = 1$ and $\partial l(\mathbf{p})/\partial p_s = 0, s = 1, \ldots, k$. Earlier for general models we rely on the convergence proof of Algorithm 2 to show the existence (see the discussion in the beginning of Section 3).

From (19), if $\delta > 0$, larger p_s implies smaller $(1+\delta)^2 - 4r_s\delta$ and hence larger r_s. The situation for $\delta < 0$ is similar. Therefore, the order of $p_1, \ldots, p_k$ is the same as that of $r_1, \ldots, r_k$:

Theorem 5 *If $r_s \geq r_t$, then $p_s \geq p_t$.*

This theorem indicates that results from the generalized Bradley-Terry model are reasonable estimates.

4.2 An Earlier Approach

Zadrozny (2002) was the first to address the probability estimates using error-correcting codes. By considering the same optimization problem (16), she proposes a heuristic updating rule

$$p_s^{t+1} \equiv \frac{\sum_{i:s\in I_i^+} r_i + \sum_{i:s\in I_i^-} r_i'}{\sum_{i:s\in I_i^+} \frac{n_i q_i^{t,+}}{q_i^t} + \sum_{i:s\in I_i^-} \frac{n_i q_i^{t,-}}{q_i^t}} p_s^t, \tag{20}$$

but does not provide a convergence proof. For the "one-against-one" setting, (20) reduces to (4) in Algorithm 1. However, we will show that under other ECOC settings, the algorithm using (20) may not converge to a point with the smallest KL distance. Taking the "one-against-the rest" approach, if $k = 3$ and $r_1 = r_2 = 3/4, r_3 = 1/2$, for our approach Theorem

5 implies $p_1 = p_2$. Then (18) and $p_1 + p_2 + p_3 = 1$ give

$$\frac{3}{4p_1} - \frac{1}{4(1-p_1)} = \frac{1}{2p_3} - \frac{1}{2(1-p_3)} = \frac{1}{2(1-2p_1)} - \frac{1}{4p_1}.$$

This leads to a solution

$$\mathbf{p} = [15 - \sqrt{33}, 15 - \sqrt{33}, 2\sqrt{33} - 6]^T/24, \tag{21}$$

which is also unique according to Theorem 4. If this is a convergent point by using (20), then a further update from it should lead to the same point (after normalization). Thus, the three multiplicative factors must be the same. Since we keep $\sum_{i=1}^k p_i^t = 1$ in the algorithm, with the property $r_i + r_i' = 1$, for this example the factor in the updating rule (20) is

$$\frac{r_s + \sum_{i:i\neq s} r_i'}{p_s^t + \sum_{i:i\neq s}(1-p_i^t)} = \frac{k - 1 + 2r_s - \sum_{i=1}^k r_i}{k - 2 + 2p_s^t} = \frac{2r_s}{1 + 2p_s^t}. \tag{22}$$

Clearly the $\mathbf{p}$ obtained earlier in (21) by our approach of minimizing the KL distance does not result in the same value for (22). Thus, in this case Zadrozny (2002)'s approach fails to converge to the unique solution of (16) and hence lacks a clear interpretation.

5. Experiments: Simulated Examples

In the following two sections, we present experiments on multi-class probability estimates using synthetic and real-world data. In implementing Algorithm 2, we use the following stopping condition:

$$\max_{s:s\in\{1,\ldots,k\}} \left| \frac{\sum_{i:s\in I_i^+} \frac{r_i}{q_i^{t,+}} + \sum_{i:s\in I_i^-} \frac{r_i'}{q_i^{t,-}}}{\sum_{i:s\in I_i} \frac{r_i + r_i'}{q_i^t}} - 1 \right| < 0.001,$$

which implies that $\partial l(\mathbf{p}^t)/\partial p_s, s = 1, \ldots, k$ are all close to zero.

5.1 Data Generation

We consider the same setting in (Hastie and Tibshirani, 1998; Wu et al., 2004) by defining three possible class probabilities:

(a) $p_1 = 1.5/k$, $p_j = (1 - p_1)/(k - 1)$, $j = 2, \ldots, k$.

(b) $k_1 = k/2$ if k is even, and $(k + 1)/2$ if k is odd; then $p_1 = 0.95 \times 1.5/k_1$, $p_i = (0.95 - p_1)/(k_1 - 1)$ for $i = 2, \ldots, k_1$, and $p_i = 0.05/(k - k_1)$ for $i = k_1 + 1, \ldots, k$.

(c) $p_1 = 0.95 \times 1.5/2$, $p_2 = 0.95 - p_1$, and $p_i = 0.05/(k - 2), i = 3, \ldots, k$.

All classes are competitive in case (a), but only two dominate in (c). For given I_i, $i = 1, \ldots, m$, we generate r_i by adding some noise to q_i^+/q_i and then check if the proposed method obtains good probability estimates. Since q_i^+/q_i of these three cases are different, it is difficult to have a fair way of adding noise. Furthermore, various ECOC settings (described later) will also result in different q_i^+/q_i. Though far from perfect, here we try two ways:

1. An "absolute" amount of noise:

$$r_i = \min(\max(\epsilon, \frac{q_i^+}{q_i} + 0.1N(0,1)), 1-\epsilon). \tag{23}$$

   Then $r_i' = 1 - r_i$. Here $\epsilon = 10^{-7}$ is used so that all r_i, r_i' are positive.

   This is the setting considered in (Hastie and Tibshirani, 1998).

2. A "relative" amount of noise:

$$r_i = \min(\max(\epsilon, \frac{q_i^+}{q_i}(1 + 0.1N(0,1))), 1-\epsilon). \tag{24}$$

   r_i' and ϵ are set in the same way.

5.2 Results of Various ECOC Settings

We consider the four encodings used in (Allwein et al., 2001) to generate I_i:

1. "1vs1": the pairwise approach (Eq. (5)).

2. "1vsrest": the "One-against-the rest" approach in Section 4.1.

3. "dense": $I_i = \{1, \ldots, k\}$ for all i. I_i is randomly split to two equally-sized sets I_i^+ and I_i^-. $[10 \log_2 k]$ [3] such splits are generated. That is, $m = [10 \log_2 k]$.

   Intuitively, more combinations of subjects as teams give more information and may lead to a better approximation of individual skill. Thus, we would like to select a diversified $I_i^+, I_i^-, i = 1, \ldots, m$. Following Allwein et al. (2001), we repeat the selection 100 times. For each collection of $I_i^+, I_i^-,\ i = 1, \ldots, m$, we calculate the smallest distance between any pair of (I_i^+, I_i^-) and (I_j^+, I_j^-). A larger value indicates better quality of the coding, so we pick the one with the largest value. For the distance between any pair of (I_i^+, I_i^-) and (I_j^+, I_j^-), Allwein et al. (2001) consider a generalized Hamming distance defined as follows:

$$\sum_{s=1}^{k} \begin{cases} 0 & \text{if } s \in I_i^+ \cap I_j^+ \text{ or } s \in I_i^- \cap I_j^-, \\ 1 & \text{if } s \in I_i^+ \cap I_j^- \text{ or } s \in I_i^- \cap I_j^+, \\ 1/2 & \text{if } s \notin I_i \text{ or } s \notin I_j. \end{cases}$$

4. "sparse": I_i^+, I_i^- are randomly drawn from $\{1, \ldots, k\}$ with $E(|I_i^+|) = E(|I_i^-|) = k/4$. Then $[15 \log_2 k]$ such splits are generated. Similar to "dense," we repeat the procedure 100 times to find a good coding.

The way of adding noise may favor some ECOC settings. Since in general

$$\frac{q_i^+}{q_i} \text{ for "1vs1"} \gg \frac{q_i^+}{q_i} \text{ for "1vsrest,"}$$

adding $0.1N(0,1)$ to q_i^+/q_i result in very inaccurate r_i for "1vsrest." On the other hand, if using a relative way, noise added to r_i and r_i' for "1vsrest" is smaller than that for

3. We use $[x]$ to denote the nearest integer value of x.

"1vs1." This analysis indicates that using the two different noise makes the experiment more complete.

Figures 1 and 2 show results of adding an "absolute" amount of noise. Two criteria are used to evaluate the obtained probability estimates: Figures 1 presents averaged accuracy rates over 500 replicates for each of the four encodings when $k = 2^2, 2^3, \ldots, 2^6$. Figure 2 gives the (relative) mean squared error (MSE):

$$\text{MSE} = \frac{1}{500} \sum_{j=1}^{500} \left(\sum_{i=1}^{k} (\hat{p}_i^j - p_i)^2 / \sum_{i=1}^{k} p_i^2 \right), \tag{25}$$

where $\hat{\mathbf{p}}^j$ is the probability estimate obtained in the jth of the 500 replicates. Using the same two criteria, Figures 3 and 4 present results of adding a "relative" amount of noise. Clearly, following our earlier analysis on adding noise, results of "1vsrest" in Figures 3 and 4 are much better than those in Figures 1 and 2. In all figures, "dense and "sparse" are less competitive in cases (a) and (b) when k is large. Due to the large $|I_i^+|$ and $|I_i^-|$, the model is unable to single out a clear winner when probabilities are more balanced. For "1vs1," it is good for (a) and (b), but suffers some losses in (c), where the class probabilities are highly unbalanced. Wu et al. (2004) have observed this shortcoming and proposed a quadratic model for the "1vs1" setting.

Results here indicate that the four encodings perform very differently under various conditions. Later in experiments for real data, we will see that in general the situation is closer to case (c), and all four encodings are practically viable.[4]

6. Experiments: Real Data

In this section we present experimental results on some real-world multi-class problems. There are two goals of experiments here:

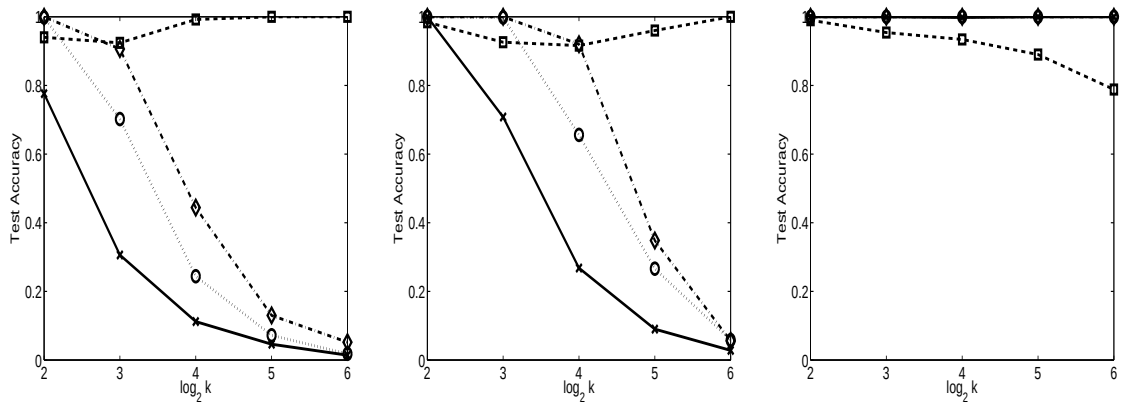
1. Check the viability of the proposed multi-class probability estimates. We hope that under reasonable ECOC settings, equally good probabilities are obtained.

2. Compare with the standard ECOC approach without extracting probabilities. This is less important than the first goal as the paper focuses on probability estimates. However, as the classification accuracy is one of the evaluation criteria used here, we can easily conduct a comparison.

6.1 Data and Experimental Settings

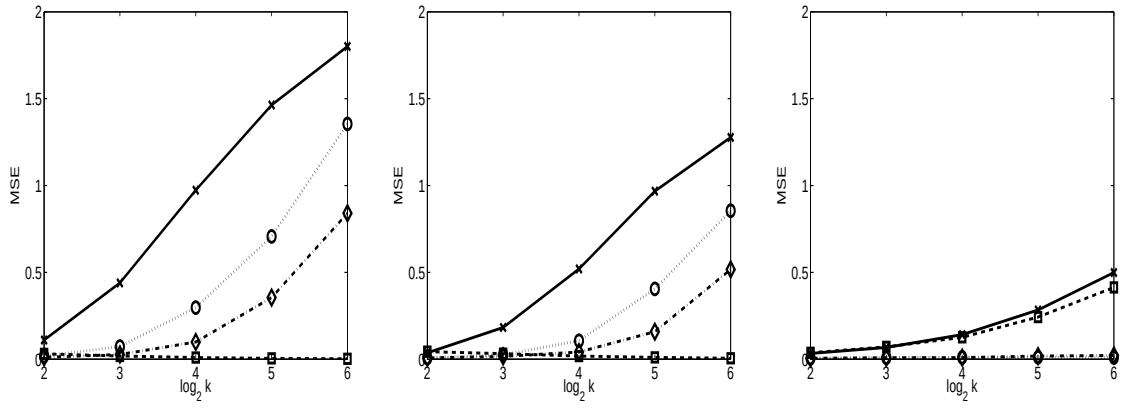
We consider data sets used in (Wu et al., 2004): dna, satimage, segment, and letter from the Statlog collection (Michie et al., 1994), waveform from UCI Machine Learning Repository (Blake and Merz, 1998), USPS (Hull, 1994), and MNIST (LeCun et al., 1998). Except dna, which takes two possible values 0 and 1, each attribute of all other data is linearly scaled to $[-1, 1]$. The data set statistics are in Table 6.1.

4. Experiments here are done using MATLAB (`http://www.mathworks.com`), and the programs are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/libsvm-errorcode/generalBT.zip`.



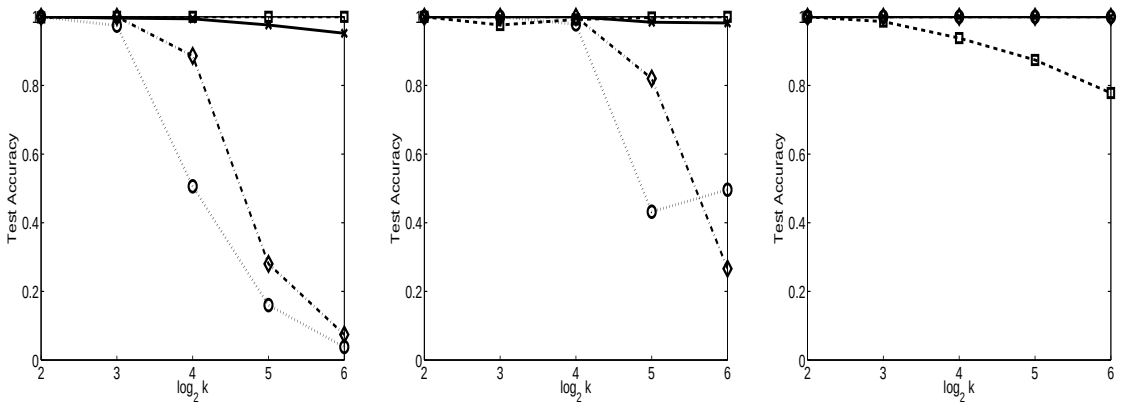
(a) (b) (c)

Figure 1: Accuracy of predicting the true class by four encodings and (23) for generating noise: "1vs1" (dashed line, square marked), "1vsrest" (solid line, cross marked), "dense" (dotted line, circle marked), "sparse" (dashdot line, diamond marked). Sub-figures 1(a), 1(b) and 1(c) correspond to the three settings of class probabilities in Section 5.1.



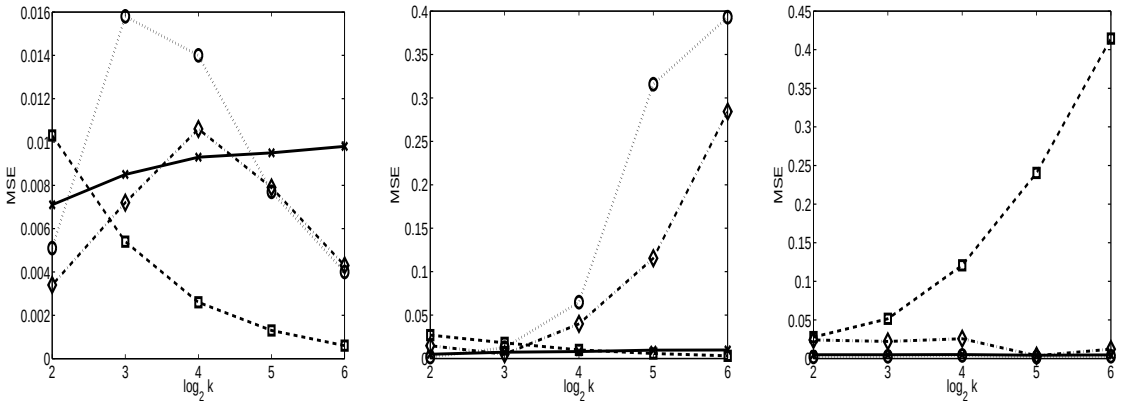
(a) (b) (c)

Figure 2: MSE by four encodings and (23) for generating noise. The legend is the same as that of Figure 1.



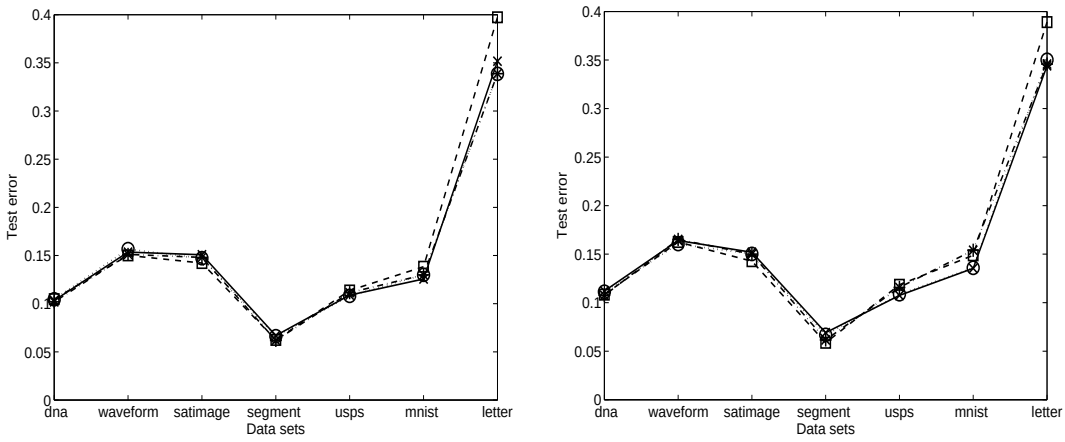
(a) (b) (c)

Figure 3: Accuracy of predicting the true class by four encodings and (24) for generating noise. The legend is the same as that of Figure 1.



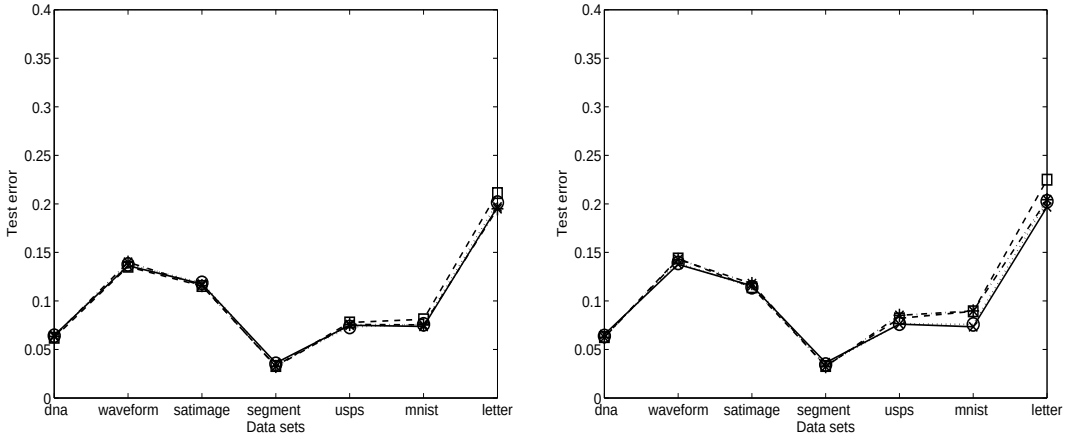
(a) (b) (c)

Figure 4: MSE by four encodings and (24) for generating noise. The legend is the same as that of Figure 1.



(a) via proposed probability estimates (b) via exponential-loss decoding

Figure 5: Testing error on smaller (300 training, 500 testing) data sets by four encodings: "1vs1" (dashed line, square marked), "1vsrest" (solid line, cross marked), "dense" (dotted line, circle marked), "sparse" (dashdot line, asterisk marked).



(a) via proposed probability estimates (b) via exponential-loss decoding

Figure 6: Testing error on larger (800 training, 1000 testing) data sets by four encodings. The legend is the same as that of Figure 5.

dataset	dna	waveform	satimage	segment	USPS	MNIST	letter
#classes	3	3	6	7	10	10	26
#attributes	180	21	36	19	256	784	16

Table 1: Data Set Statistics

After data scaling, we randomly select smaller (300/500) and larger (800/1,000) training/testing sets from thousands of points for experiments. 20 such selections are generated and results are averaged.[5]

We use the same four ways in Section 5 to generate I_i. All of them have $|I_1| \approx \cdots \approx |I_m|$. With the property that these multi-class problems are reasonably balanced, we set $n_i = 1$ in (16).

We consider support vector machines (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) with the RBF (Radial Basis Function) kernel $e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}$ as the binary classifier. An improved version (Lin et al., 2003) of (Platt, 2000) obtains r_i using SVM decision values. It is known that SVM may not give good probability estimates (e.g., Zhang (2004)), but Platt (2000) and Wu et al. (2004) empirically show that using decision values from cross validation yields acceptable results in practice. In addition, SVM is sometimes sensitive to parameters, so we conduct a selection procedure before testing. Details can be found in Figure 4 of (Wu et al., 2004). The code is modified from LIBSVM (Chang and Lin, 2001), a library for support vector machines.

6.2 Evaluation Criteria and Results

For these real data sets, there are no true probability values available. We consider the same three evaluation criteria used in (Wu et al., 2004):

1. Test errors. Averages of 20 errors for smaller and larger sets are in Figures 5(a) and 6(a), respectively.

2. MSE (Brier Score).
$$\frac{1}{l}\sum_{j=1}^{l}\left(\sum_{i=1}^{k}(I_{y_j=i} - \hat{p}_i^j)^2\right),$$
where l is the number of test data, $\hat{\mathbf{p}}^j$ is the probability estimate of the jth data, y_j is the true class label, and $I_{y_j=i}$ is an indicator function (1 if $y_j = i$ and 0 otherwise). This measurement (Brier, 1950), popular in meteorology, satisfies the following property:
$$\arg\min_{\hat{\mathbf{p}}} E_Y[\sum_{i=1}^{k}(I_{Y=i} - \hat{p}_i)^2] \equiv \arg\min_{\hat{\mathbf{p}}} \sum_{i=1}^{k}(\hat{p}_i - p_i)^2,$$
where Y, a random variable for the class label, has the probability distribution $\mathbf{p}$. Brier score is thus useful when the true probabilities are unknown. We present the average of 20 Brier scores in Figure 7.

5. All training/testing sets used are at `http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob/data`.

3. Log loss:

$$\frac{-1}{l}\sum_{j=1}^{l}\log \hat{p}^j_{y_j},$$

where $\hat{\mathbf{p}}^j$ is the probability estimate of the jth data and y_j is its actual class label. It is another useful criterion when true probabilities are unknown:

$$\min_{\hat{\mathbf{p}}} E_Y[-\sum_{i=1}^{k}\log \hat{p}_i \cdot I_{Y=i}] \equiv \min_{\hat{\mathbf{p}}} -\sum_{i=1}^{k} p_i \log \hat{p}_i$$

has the minimum at $\hat{p}_i = p_i, i = 1, \ldots, k$. Average of 20 splits are presented in Figure 8.

Results of using the three criteria all indicate that the four encodings are quite competitive. Such an observation suggests that in practical problems class probabilities may resemble those specified in case (c) in Section 5; that is, only few classes dominate. Wu et al. (2004) is the first one pointing out this resemblance. In addition, all figures show that "1vs1" is slightly worse than others in the case of larger k (e.g., letter). Earlier Wu et al. (2004) proposed a quadratic model, which gives better probability estimates than the Bradley-Terry model for "1vs1."

In terms of the computational time, because the number of binary problems for "dense" and "sparse" ($[10\log_2 k]$ and $[15\log_2 k]$, respectively) are larger than k, and each binary problem involves many classes of data (all and one half), their training time is longer than that of "1vs1" and "1vsrest." "Dense" is particularly time consuming. Note that though "1vs1" solves $k(k-1)/2$ SVMs, each is small via using only two classes of data.

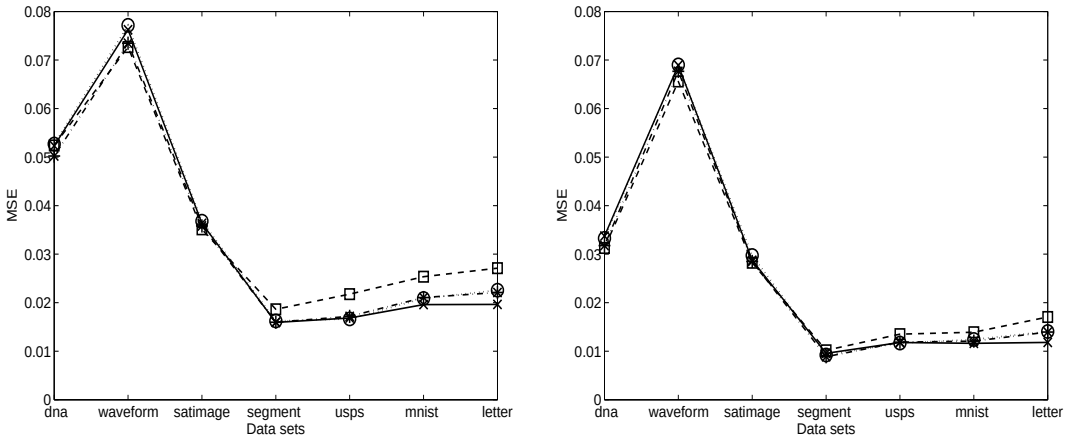
To check the effectiveness of the proposed model in multi-class classification, we compare it with a standard ECOC-based strategy which does not produce probabilities: exponential loss-based decoding by Allwein et al. (2001). Let $\hat{f}_i$ be the decision function of the ith binary classifier, and $\hat{f}_i(\mathbf{x}) > 0$ (< 0) specifies that data $\mathbf{x}$ to be in classes in I_i^+ (I_i^-). This approach determines the predicted label by the following rule:

$$\text{predicted label } = \arg\min_s \left(\sum_{i:s\in I_i^+} e^{-\hat{f}_i} + \sum_{i:s\in I_i^-} e^{\hat{f}_i} \right).$$

Testing errors for smaller and larger sets are in Figures 5(b) and 6(b), respectively. Comparing them with results by the proposed model in Figures 5(a) and 6(a), we observe that both approaches have very similar errors. Therefore, in terms of predicting class labels only, our new method is competitive.

7. Extensions of the Generalized Bradley-Terry Model

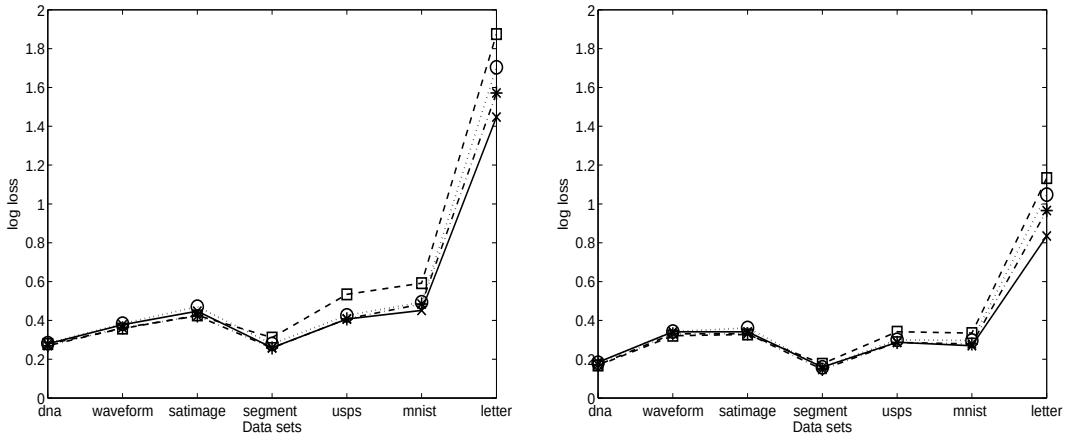
In addition to multi-class probability estimates, the proposed generalized Bradley-Terry model, as mentioned in Section 1, has some potential applications in sports. We consider in this section several extensions based on common sport scenarios and show that, with a slight modification of Algorithm 2, they can be easily solved as well.



(a) 300 training, 500 testing

(b) 800 training, 1000 testing

Figure 7: MSE by four encodings. The legend is the same as that of Figure 5.



(a) 300 training, 500 testing

(b) 800 training, 1000 testing

Figure 8: Log loss by four encodings. The legend is the same as that of Figure 5.

7.1 Weighted Individual Skill

In some sports, team performance is highly affected by certain positions. For example, many people think guards are relatively more important than centers and forwards in basketball games. We can extend the generalized Bradley-Terry model to this case: Define

$$\bar{q}_i \equiv \sum_{j:j\in I_i} w_{ij}p_j, \quad \bar{q}_i^+ \equiv \sum_{j:j\in I_i^+} w_{ij}p_j, \quad \bar{q}_i^- \equiv \sum_{j:j\in I_i^-} w_{ij}p_j,$$

where $w_{ij} > 0$ is a given weight parameter reflecting individual j's position in the game between I_i^+ and I_i^-. By minimizing the same negative log-likelihood function (6), estimated

individual skill can be obtained. Here Algorithm 2 can still be applied but with the updating rule replaced by

$$p_s^{t+1} = \frac{\sum_{i:s\in I_i^+} \frac{r_i w_{is}}{\bar{q}_i^{t,+}} + \sum_{i:s\in I_i^-} \frac{r'_i w_{is}}{\bar{q}_i^{t,-}}}{\sum_{i:s\in I_i} \frac{(r_i+r'_i)w_{is}}{\bar{q}_i^t}} p_s^t, \tag{26}$$

which is derived similarly to (7) so that the multiplicative factor is equal to one when $\partial l(\mathbf{p})/\partial p_s = 0$. The convergence can be proved similarly. However, it may be harder to obtain the global optimality: Case 1 in Theorem 4 still holds, but Case 2 may not since $\bar{q}_i$ needs not be equal to one (the proof of Case 2 requires $q_i = 1$, which is guaranteed by $|I_i| = k$).

7.2 Home-field Advantage

The original home-field advantage model (Agresti, 1990) is based on paired individual comparisons. We can incorporate its idea into our proposed model by taking

$$P(I_i^+ \text{ beats } I_i^-) = \begin{cases} \frac{\theta q_i^+}{\theta q_i^+ + q_i^-} & \text{if } I_i^+ \text{ is home,} \\ \frac{q_i^+}{q_i^+ + \theta q_i^-} & \text{if } I_i^- \text{ is home,} \end{cases}$$

where $\theta > 0$ measures the strength of the home-field advantage or disadvantage. Note that θ is an unknown parameter to be estimated, while the weights w_{ij} in Section 7.1 are given.

Let $\bar{r}_i \geq 0$ and $\bar{r}'_i \geq 0$ be the number of times that I_i^+ wins and loses at home, respectively. For I_i^+'s away games, we let $\tilde{r}_i \geq 0$ and $\tilde{r}'_i \geq 0$ be the number of times that I_i^+ wins and loses. The minimization of the negative log-likelihood function thus becomes:

$$\min_{\mathbf{p},\theta} \; l(\mathbf{p},\theta) = \\ -\sum_{i=1}^m \left(\bar{r}_i \log \frac{\theta q_i^+}{\theta q_i^+ + q_i^-} + \tilde{r}_i \log \frac{q_i^+}{q_i^+ + \theta q_i^-} + \bar{r}'_i \log \frac{q_i^-}{\theta q_i^+ + q_i^-} + \tilde{r}'_i \log \frac{\theta q_i^-}{q_i^+ + \theta q_i^-} \right)$$

under the constraints in (6) and the condition $\theta \geq 0$.

To apply Algorithm 2 on the new optimization problem, we must modify the updating rule. For each s, $\partial l(\mathbf{p},\theta)/\partial p_s = 0$ leads to the following rule:[6]

$$p_s^{t+1} = \frac{\sum_{i:s\in I_i^+} \frac{\bar{r}_i+\tilde{r}_i}{q_i^+} + \sum_{i:s\in I_i^-} \frac{\bar{r}'_i+\tilde{r}'_i}{q_i^-}}{\sum_{i:s\in I_i^+} \left(\frac{\theta(\bar{r}_i+\bar{r}'_i)}{\theta q_i^+ + q_i^-} + \frac{\tilde{r}_i+\tilde{r}'_i}{q_i^+ + \theta q_i^-} \right) + \sum_{i:s\in I_i^-} \left(\frac{\bar{r}_i+\bar{r}'_i}{\theta q_i^+ + q_i^-} + \frac{\theta(\tilde{r}_i+\tilde{r}'_i)}{q_i^+ + \theta q_i^-} \right)} p_s^t. \tag{27}$$

For θ, from $\partial l(\mathbf{p},\theta)/\partial\theta = 0$, we have

$$\theta^{t+1} = \frac{\sum_{i=1}^m (\bar{r}_i + \tilde{r}'_i)}{\sum_{i=1}^m \left(\frac{q_i^+(\bar{r}_i+\bar{r}'_i)}{\theta^t q_i^+ + q_i^-} + \frac{q_i^-(\tilde{r}_i+\tilde{r}'_i)}{q_i^+ + \theta^t q_i^-} \right)}. \tag{28}$$

6. For convenience, $q_i^{t,+}(q_i^{t,-})$ is abbreviated as $q_i^+(q_i^-)$. The same abbreviation is used in the updating rule in Sections 7.3 and 7.4.

Unlike the case of updating p_s^t, there is no need to normalize θ^{t+1}. The algorithm then cyclically updates $p_1, \ldots, p_k$, and θ. If p_s is updated, we can slightly modify the proof of Theorem 1 and obtain the strict decrease of $l(\mathbf{p}, \theta)$. Moreover, Appendix F gives a simple derivation of $l(\mathbf{p}^t, \theta^{t+1}) < l(\mathbf{p}^t, \theta^t)$. Thus, if we can ensure that θ^t is bounded above, then under a modified version of Assumption 1 where $\max(\bar{r}_{s_i}, \tilde{r}_{s_i}) > 0$ replaces $r_{s_i} > 0$, the convergence of Algorithm 2 (i.e., Theorem 3) still holds by a similar proof.

7.3 Ties

Suppose ties are possible between teams. Extending the model proposed in (Rao and Kupper, 1967), we consider:

$$\begin{aligned} P(I_i^+ \text{ beats } I_i^-) &= \frac{q_i^+}{q_i^+ + \theta q_i^-}, \\ P(I_i^- \text{ beats } I_i^+) &= \frac{q_i^-}{\theta q_i^+ + q_i^-}, \text{ and} \\ P(I_i^+ \text{ ties } I_i^-) &= \frac{(\theta^2 - 1) q_i^+ q_i^-}{(q_i^+ + \theta q_i^-)(\theta q_i^+ + q_i^-)}, \end{aligned}$$

where $\theta > 1$ is a threshold parameter to be estimated.

Let t_i be the number of times that I_i^+ ties I_i^- and r_i, r_i' defined as before. We then minimize the following negative log-likelihood function:

$$\begin{aligned} &\min_{\mathbf{p},\theta} \; l(\mathbf{p}, \theta) \\ &= -\sum_{i=1}^m \left(r_i \log \frac{q_i^+}{q_i^+ + \theta q_i^-} + r_i' \log \frac{q_i^-}{\theta q_i^+ + q_i^-} + t_i \log \frac{(\theta^2 - 1) q_i^+ q_i^-}{(q_i^+ + \theta q_i^-)(\theta q_i^+ + q_i^-)} \right) \\ &= -\sum_{i=1}^m \left(r_i \log \frac{q_i^+}{q_i^+ + \theta q_i^-} + r_i' \log \frac{q_i^-}{\theta q_i^+ + q_i^-} + t_i \log \frac{\theta q_i^+}{\theta q_i^+ + q_i^-} + t_i \log \frac{\theta q_i^-}{q_i^+ + \theta q_i^-} \right) \quad (29) \\ &\quad -\sum_{i=1}^m t_i \log \frac{\theta^2 - 1}{\theta^2} \quad (30) \end{aligned}$$

under the constraints in (6) and the condition $\theta > 1$.

For updating p_s^t, θ is considered as a constant and (29) is in a form of the Home-field model, so the rule is similar to (27). The strict decrease of $l(\mathbf{p}, \theta)$ can be established as well. For updating θ, we have

$$\theta^{t+1} = \frac{1}{2C_t} + \sqrt{1 + \frac{1}{4C_t^2}}, \quad (31)$$

where

$$C_t = \frac{1}{2\sum_{i=1}^m t_i} \left(\sum_{i=1}^m \frac{(r_i + t_i) q_i^-}{q_i^+ + \theta^t q_i^-} + \sum_{i=1}^m \frac{(r_i' + t_i) q_i^+}{\theta^t q_i^+ + q_i^-} \right).$$

The derivation and the strict decrease of $l(\mathbf{p}, \theta)$ are in Appendix F. If we can ensure that $1 < \theta^t < \infty$ and modify Assumption 1 as in Section 7.2, the convergence of Algorithm 2 also holds.

7.4 Multiple Team Comparisons

In this type of comparison, a game may include more than two participants, and the result is a ranking of the participants. For a game of three participants. Pendergrass and Bradley (1960) proposed using

$$\begin{aligned} & P(i \text{ best, } j \text{ in the middle, and } k \text{ worst}) \\ = \; & P(i \text{ beats } j \text{ and } k) \cdot P(j \text{ beats } k) \\ = \; & \frac{p_i}{p_i + (p_j + p_k)} \cdot \frac{p_j}{p_j + p_k}. \end{aligned}$$

A general model introduced in (Placket, 1975) is:

$$P(a(1) \to a(2) \to \cdots \to a(k)) = \prod_{i=1}^{k} \frac{p_{a(i)}}{p_{a(i)} + p_{a(i+1)} + \cdots + p_{a(k)}}, \tag{32}$$

where $a(i), 1 \le i \le k$ is the ith ranked individual and $\to$ denotes the relation "is ranked higher than." A detailed discussion of this model is in (Hunter, 2004, Section 5).

With similar ideas, we consider a more general setting: Each game may include more than two participating teams. Assume that there are k individuals and N games resulting in N rankings; the mth game involves g_m disjoint teams. Let $I_m^i \subset \{1, \ldots, k\}$ be the ith ranked team in the mth game, $1 \le i \le g_m$, $1 \le m \le N$. We consider the model:

$$P(I_m^1 \to I_m^2 \to \cdots \to I_m^{g_m}) = \prod_{i=1}^{g_m} \frac{\sum_{s:s \in I_m^i} p_s}{\sum_{j=i}^{g_m} \sum_{s:s \in I_m^j} p_s}. \tag{33}$$

Defining

$$q_m^i = \sum_{s:s \in I_m^i} p_s,$$

we minimize the negative log-likelihood function:

$$\min_{\mathbf{p}} l(\mathbf{p}) = -\sum_{m=1}^{N} \sum_{i=1}^{g_m} \log \frac{q_m^i}{\sum_{j=i}^{g_m} q_m^j} \tag{34}$$

under the constraints in (6).

In fact, (34) is a special case of (6). Each ranking can be viewed as the result of a series of paired team comparisons: the first ranked team beats the others, the second ranked team beats the others except the first, and so on; for each paired comparison, $r_i = 1$ and $r_i' = 0$. Therefore, Algorithm 2 can be applied and the updating rule is:

$$p_s^{t+1} = \frac{\sum_{j:s \in I_j} (q_j^{\phi_j(s)})^{-1}}{\sum_{j:s \in I_j} \sum_{i=1}^{\phi_j(s)} (\sum_{v=i}^{g_j} q_j^v)^{-1}} p_s^t, \tag{35}$$

where $\phi_j(s)$ is the rank of the team that individual s belongs to in the jth game and $I_j = \cup_{i=1}^{g_j} I_j^i$.

We explain in detail how (35) is derived. Since teams are disjoint in one game and (33) implies that ties are not allowed, $\phi_j(i)$ is unique under a given i. In the jth game, individual s appears in $\phi_j(s)$ paired comparisons:

$$\begin{array}{lll} I_j^1 & \text{vs.} & I_j^2 \cup \cdots \cup I_j^{g_j}, \\ I_j^2 & \text{vs.} & I_j^3 \cup \cdots \cup I_j^{g_j}, \\ & \vdots & \\ I_j^{\phi_j(s)} & \text{vs.} & I_j^{\phi_j(s)+1} \cup \cdots \cup I_j^{g_j}. \end{array}$$

From (7), the numerator of the multiplicative factor involves winning teams that individual s is in, so there is only one (i.e., $\phi_j(s)$) in each game that s joins; the denominator involves teams of both sides, so it is in the form of $\sum_{i=1}^{\phi_j(s)}(\sum_{v=i}^{g_j} q_j^v)^{-1}$.

8. Discussion and Conclusions

We propose a generalized Bradley-Terry model which gives individual skill from group competition results. We develop a simple iterative method to maximize the log-likelihood and prove the convergence. The new model has many potential applications. In particular, minimizing the negative log likelihood of the proposed model coincides with minimizing the KL distance for multi-class probability estimates under error correcting output codes. Hence the iterative scheme is useful for finding class probabilities. Similar to the original Bradley-Terry model, we can extend the proposed generalized model to other settings such as home-field advantages, ties, and multiple team comparisons.

Investigating more practical applications using the proposed model is certainly an important future direction. The lack of convexity of $l(\mathbf{p})$ also requires more studies. In Section 5, the "sparse" coding has $E(|I_i^+|) = E(|I_i^-|) = k/4$, and hence is not covered by Theorem 4 which proves the global optimality. However, this coding is competitive with others in Section 6. If possible, we hope to show in the future that in general the global optimality holds.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grants NSC 92-2213-E-002-062 and NSC 92-2118-M-004-003.

Appendix A. Proof of Theorem 1

Define

$$q_{i\backslash s}^{t,+} \equiv \sum_{j \in I_i^+, j \neq s} p_j^t, \text{ and } q_{i\backslash s}^t \equiv \sum_{j \in I_i^-, j \neq s} p_j^t.$$

Using

$$-\log x \geq 1 - \log y - x/y \text{ with equality if and only if } x = y,$$

we have

$$Q^1(p_s) \geq l([p_1^t, \ldots, p_{s-1}^t, p_s, p_{s+1}^t, \ldots, p_k^t]^T) \text{ with equality if } p_s = p^t,$$

where

$$\begin{aligned} Q^1(p_s) &\equiv -\sum_{i:s\in I_i^+} r_i \left(\log(q_{i\backslash s}^{t,+} + p_s) - \frac{q_{i\backslash s}^t + p_s}{q_i^t} - \log q_i^t + 1 \right) - \\ &\quad \sum_{i:s\in I_i^-} r_i' \left(\log(q_{i\backslash s}^{t,-} + p_s) - \frac{q_{i\backslash s}^t + p_s}{q_i^t} - \log q_i^t + 1 \right) \\ &= -\sum_{i:s\in I_i^+} r_i \log(q_{i\backslash s}^{t,+} + p_s) - \sum_{i:s\in I_i^-} r_i' \log(q_{i\backslash s}^{t,-} + p_s) + \\ &\quad \sum_{i:s\in I_i} (r_i + r_i') \left(\frac{q_{i\backslash s}^t + p_s}{q_i^t} + \log q_i^t - 1 \right). \end{aligned}$$

For $0 < \lambda < 1$, we have

$$\log(\lambda x + (1-\lambda)y) \geq \lambda \log x + (1-\lambda)\log y \text{ with equality if and only if } x = y.$$

With $p_s^t > 0$,

$$\begin{aligned} & \log(q_{i\backslash s}^{t,+} + p_s) \\ = \; & \log\left(\frac{q_{i\backslash s}^{t,+}}{q_i^{t,+}} \cdot 1 + \frac{p_s^t}{q_i^{t,+}} \cdot \frac{p_s}{p_s^t} \right) + \log(q_i^{t,+}) \\ \geq \; & \frac{p_s^t}{q_i^{t,+}} (\log p_s - \log p_s^t) + \log q_i^{t,+} \text{ with equality if } p_s = p_s^t. \end{aligned}$$

Then

$$Q^2(p_s) \geq l([p_1^t, \ldots, p_{s-1}^t, p_s, p_{s+1}^t, \ldots, p_k^t]^T) \text{ with equality if } p_s = p_s^t,$$

where

$$\begin{aligned} & Q^2(p_s) \\ \equiv \; & -\sum_{i:s\in I_i^+} r_i \left(\frac{p_s^t}{q_i^{t,+}} (\log p_s - \log p_s^t) + \log q_i^{t,+} \right) - \\ & \sum_{i:s\in I_i^-} r_i' \left(\frac{p_s^t}{q_i^{t,-}} (\log p_s - \log p_s^t) + \log q_i^{t,-} \right) + \sum_{i:s\in I_i} (r_i + r_i') \left(\frac{q_{i\backslash s}^t + p_s}{q_i^t} + \log q_i^t - 1 \right). \end{aligned}$$

As we assume $p_s^t > 0$ and $\sum_{i:s\in I_i}(r_i + r_i') > 0$, $Q^2(p_s)$ is a strictly convex function of p_s. By $dQ^2(p_s)/dp_s = 0$,

$$\left(\sum_{i:s\in I_i^+} \frac{r_i}{q_i^{t,+}} \sum_{i:s\in I_i^-} \frac{r_i'}{q_i^{t,-}} \right) \frac{p_s^t}{p_s} = \sum_{i:s\in I_i} \frac{r_i + r_i'}{q_i^t}$$

leads to the updating rule. Thus, if $p_s^{t+1} \neq p_s^t$, then

$$l(\mathbf{p}^{t+1}) \leq Q^2(p_s^{t+1}) < Q^2(p_s^t) = l(\mathbf{p}^t).$$

Appendix B. Proof of Lemma 2

If the result does not hold, there is an index $\bar{s}$ and an infinite index set T such that

$$\lim_{t\in T, t\to\infty} p_{\bar{s}}^t = p_{\bar{s}}^* = 0.$$

Since $\sum_{s=1}^k p_s^t = 1, \forall t$ and k is finite,

$$\lim_{t\in T, t\to\infty} \sum_{s=1}^k p_s^t = \sum_{s=1}^k p_s^* = 1.$$

Thus, there is an index j such that

$$\lim_{t\in T, t\to\infty} p_j^t = p_j^* > 0. \tag{36}$$

Under Assumption 1, one of the two conditions linking individual $\bar{s}$ and j must hold. As both cases are similar, we consider only the first here. With $p_{\bar{s}}^* = 0$ and $I_{s_0}^+ = \{\bar{s}\}$, we claim that $p_u^* = 0, \forall u \in I_{s_0}^-$. If this claim is wrong, then

$$\begin{aligned} l(\mathbf{p}^t) &= -\sum_{i=1}^m \left(r_i \log \frac{q_i^{t,+}}{q_i^t} + r_i' \log \frac{q_i^{t,-}}{q_i^t} \right) \\ &\geq -r_{s_0} \log \frac{q_{s_0}^{t,+}}{q_{s_0}^t} \\ &= -r_{s_0} \log \frac{p_{\bar{s}}^t}{p_{\bar{s}}^t + \sum_{u\in I_{s_0}^-} p_u^t} \\ &\to \infty \text{ when } t \in T, t \to \infty. \end{aligned}$$

This result contradicts Theorem 1, which implies $l(\mathbf{p}^t)$ is bounded above by $l(\mathbf{p}^0)$. Thus, $p_u^* = 0, \forall u \in I_{s_0}$. With $I_{s_1}^+ \subset I_{s_0}$, we can use the same way to prove $p_u^* = 0, \forall u \in I_{s_1}$. Continuing the same derivation, in the end $p_u^* = 0, \forall u \in I_{s_t}$. Since $j \in I_{s_t}^-$, $p_j^* = 0$ contradicts (36) and the proof is complete.

Appendix C. Proof of Theorem 3

Recall that we assume $\lim_{t\in K, t\to\infty} \mathbf{p}^t = \mathbf{p}^*$. For each $\mathbf{p}^t, t \in K$, there is a corresponding index s for the updating rule (7). Thus, one of $\{1, \ldots, k\}$ must be considered infinitely many times. Without loss of generality, we assume that all $\mathbf{p}^t, t \in K$ have the same corresponding s. If $\mathbf{p}^*$ does not satisfy

$$\frac{\partial l(\mathbf{p}^*)}{\partial p_j} = 0, \ j = 1, \ldots, k,$$

starting from $s, s{+}1, \ldots, k, 1, \ldots, s{-}1$, there is a first component $\bar{s}$ such that $\partial l(\mathbf{p}^*)/\partial p_{\bar{s}} \neq 0$. As $p_{\bar{s}}^* > 0$, by applying one iteration of Algorithm 2 on $p_{\bar{s}}^*$, and using Theorem 1, we obtain $\mathbf{p}^{*+1} \neq \mathbf{p}^*$ and

$$l(\mathbf{p}^{*+1}) < l(\mathbf{p}^*). \tag{37}$$

Since $\bar{s}$ is the first index so that the partial derivative is not zero,

$$\frac{\partial l(\mathbf{p}^*)}{\partial p_s} = 0 = \cdots = \frac{\partial l(\mathbf{p}^*)}{\partial p_{\bar{s}-1}}.$$

Thus, at the tth iteration,

$$\lim_{t\in K, t\to\infty} p_s^{t+1} = \lim_{t\in K, t\to\infty} \frac{\sum_{i:s\in I_i^+} \frac{r_i}{q_i^{t,+}} + \sum_{i:s\in I_i^-} \frac{r_i'}{q_i^{t,-}}}{\sum_{i:s\in I_i} \frac{r_i+r_i'}{q_i^t}} p_s^t = \frac{\sum_{i:s\in I_i^+} \frac{r_i}{q_i^{*,+}} + \sum_{i:s\in I_i^-} \frac{r_i'}{q_i^{*,-}}}{\sum_{i:s\in I_i} \frac{r_i+r_i'}{q_i^*}} p_s^* = p_s^*$$

and hence

$$\lim_{t\in K, t\to\infty} \mathbf{p}^{t+1} = \lim_{t\in K, t\to\infty} \mathbf{p}^t = \mathbf{p}^*.$$

Assume $\bar{s}$ corresponds to the $\bar{t}$th iteration, by a similar derivation,

$$\lim_{t\in K, t\to\infty} \mathbf{p}^{t+1} = \cdots = \lim_{t\in K, t\to\infty} \mathbf{p}^{\bar{t}} = \mathbf{p}^*$$

and

$$\lim_{t\in K, t\to\infty} \mathbf{p}^{\bar{t}+1} = \mathbf{p}^{*+1}.$$

Thus, with (37),

$$\lim_{t\in K, t\to\infty} l(\mathbf{p}^{\bar{t}+1}) = l(\mathbf{p}^{*+1}) < l(\mathbf{p}^*)$$

contradicts the fact that

$$l(\mathbf{p}^*) \le \cdots \le l(\mathbf{p}^t), \forall t.$$

Appendix D. Proof of Theorem 4

The first case reduces to the original Bradley-Terry model, so we can directly use existing results. As explained in Section 3, Assumption 1 goes back to (Hunter, 2004, Assumption 1). Then the results in Section 4 of (Hunter, 2004) imply that (6) has a unique global minimum and Algorithm 2 globally converges to it.

For the second case, as $q_i = \sum_{j=1}^k p_j = 1$, $l(\mathbf{p})$ can be reformulated as

$$\bar{l}(\mathbf{p}) = -\sum_{i=1}^m (r_i \log q_i^+ + r_i' \log q_i^-), \tag{38}$$

which is a convex function of $\mathbf{p}$. Then solving (6) is equivalent to minimizing (38). One can also easily show that they have the same set of stationary points.

From Assumption 1, for each p_j, there is i such that either $I_i^+ = \{s\}$ with $r_i > 0$, or $I_i^- = \{s\}$ with $r_i' > 0$. Therefore, either $-r_i \log p_s$ or $-r_i' \log p_s$ appears in (38). Since they are strictly convex functions of p_s, the summation on all $s = 1, \ldots, k$ makes (38) a strictly convex function of $\mathbf{p}$. Hence (6) has a unique global minimum, which is also (6)'s unique stationary point. From Theorem 3, Algorithm 2 globally converges to this unique minimum.

Appendix E. Solving Nonlinear Equations for the "One-against-the-rest" Approach

We show that if $\delta \neq 0$, p_s defined in (19) satisfies $0 \leq p_s \leq 1$. If $\delta > 0$, then

$$(1+\delta) \geq \sqrt{(1+\delta)^2 - 4r_s\delta},$$

so $p_s \geq 0$. The situation for $\delta < 0$ is similar. To prove $p_s \leq 1$, we consider three cases:

1. $\delta \geq 1$.
   Clearly,
   $$p_s = \frac{(1+\delta) - \sqrt{(1+\delta)^2 - 4r_s\delta}}{2\delta}. \leq \frac{1+\delta}{2\delta} \leq 1.$$

2. $0 < \delta < 1$.
   With $0 \leq r_s \leq 1$, we have $4\delta - 4r_s\delta \geq 0$ and
   $$(1+\delta)^2 - 4r_s\delta \geq 1 - 2\delta + \delta^2.$$
   Using $0 < \delta < 1$,
   $$p_s = \frac{(1+\delta) - \sqrt{(1+\delta)^2 - 4r_s\delta}}{2\delta}. \leq \frac{1+\delta-(1-\delta)}{2\delta} = 1.$$

3. $\delta < 0$.
   Now $4\delta - 4r_s\delta \leq 0$, so
   $$0 \leq (1+\delta)^2 - 4r_s\delta \leq 1 - 2\delta + \delta^2.$$
   Then
   $$-\sqrt{(1+\delta)^2 - 4r_s\delta} \geq \delta - 1.$$
   Adding $1+\delta$ on both sides and dividing them by 2δ leads to $p_s \leq 1$.

To find δ by solving $\sum_{s=1}^{k} p_s - 1 = 0$, the discontinuity at $\delta = 0$ is a concern. A simple calculation shows

$$\lim_{\delta \to 0} \sum_{s=1}^{k} \frac{(1+\delta) - \sqrt{(1+\delta)^2 - 4r_s\delta}}{2\delta} - 1 = \sum_{s=1}^{k} r_s - 1.$$

One can thus define the following continuous function:

$$f(\delta) = \begin{cases} \sum_{s=1}^{k} r_s - 1 & \text{if } \delta = 0, \\ \sum_{s=1}^{k} \frac{(1+\delta) - \sqrt{(1+\delta)^2 - 4r_s\delta}}{2\delta} - 1 & \text{otherwise.} \end{cases}$$

Since

$$\lim_{\delta \to -\infty} f(\delta) = k - 1 > 0 \text{ and } \lim_{\delta \to \infty} f(\delta) = -1 < 0,$$

$f(\delta) = 0$ has at least one root. Next we show that $f(\delta)$ is strictly decreasing: Consider $\delta \neq 0$, then

$$f'(\delta) = \sum_{s=1}^{k} \frac{-1 + \frac{1+\delta-2r_s\delta}{\sqrt{(1+\delta)^2-4r_s\delta}}}{2\delta^2}.$$

If $1+\delta-2r_s\delta \leq 0$, then of course $f'(\delta) < 0$. For the other case, first we use $-4r_s\delta^2+4r_s^2\delta^2 < 0$ to obtain

$$(1+\delta)^2 - 4r_s\delta(1+\delta) + 4r_s^2\delta^2 < (1+\delta)^2 - 4r_s\delta.$$

Since $1+\delta-2r_s\delta > 0$, taking the square root on both sides leads to $f'(\delta) < 0$.

Therefore,

$$f(\delta) = 0 \text{ has a unique solution at } \delta > 0 \ (< 0) \text{ if } \sum_{s=1}^{k} r_s - 1 > 0 \ (< 0).$$

Appendix F. Update θ for Models with Home-field Advantages or Ties

For the Home-field model, we use the "minorizing"-function approach in (Hunter, 2004):

$$\begin{aligned}
& \text{Terms of } l(\mathbf{p}^t, \theta) \text{ related to } \theta \\
= \ & -\sum_{i=1}^{m} \left((\bar{r}_i + \tilde{r}'_i) \log\theta - (\bar{r}_i + \bar{r}'_i) \log(\theta q_i^+ + q_i^-) - (\tilde{r}_i + \tilde{r}'_i) \log(q_i^+ + \theta q_i^-) \right) \\
\leq \ & -\sum_{i=1}^{m} \Bigg((\bar{r}_i + \tilde{r}'_i) \log\theta - (\bar{r}_i + \bar{r}'_i) \big(-1 + \log(\theta^t q_i^+ + q_i^-) + \frac{\theta q_i^+ + q_i^-}{\theta^t q_i^+ + q_i^-}\big) \\
& \qquad -(\tilde{r}_i + \tilde{r}'_i) \big(-1 + \log(q_i^+ + \theta^t q_i^-) + \frac{q_i^+ + \theta q_i^-}{q_i^+ + \theta^t q_i^-}\big) \Bigg) \\
\equiv \ & Q(\theta).
\end{aligned}$$

The inequality becomes equality if $\theta = \theta^t$. Thus, $Q'(\theta) = 0$ leads to (28) and $l(\mathbf{p}^t, \theta^{t+1}) < l(\mathbf{p}^t, \theta^t)$.

For the model which allows ties, we again define a minorizing function of θ.

$$\begin{aligned}
& \text{Terms of } l(\mathbf{p}^t, \theta) \text{ related to } \theta \\
= \ & -\sum_{i=1}^{m} \left(t_i \log(\theta^2 - 1) - (r_i + t_i) \log(q_i^+ + \theta q_i^-) - (r'_i + t_i) \log(\theta q_i^+ + q_i^-) \right) \\
\leq \ & -\sum_{i=1}^{m} \Bigg(t_i \log(\theta^2 - 1) - (r_i + t_i) \left(1 + \log(q_i^+ + \theta^t q_i^-) + \frac{q_i^+ + \theta q_i^-}{q_i^+ + \theta^t q_i^-} \right) \\
& \qquad -(r'_i + t_i) \left(1 + \log(\theta^t q_i^+ + q_i^-) + \frac{\theta q_i^+ + q_i^-}{\theta^t q_i^+ + q_i^-} \right) \Bigg) \\
\equiv \ & Q(\theta).
\end{aligned}$$

Then $Q'(\theta) = 0$ implies

$$\sum_{i=1}^{m} \left(\frac{2\theta t_i}{\theta^2 - 1} - \frac{(r_i + t_i) q_i^-}{q_i^+ + \theta^t q_i^-} - \frac{(r'_i + t_i) q_i^+}{\theta^t q_i^+ + q_i^-} \right) = 0$$

and hence θ^{t+1} is defined as in (31).

References

A. Agresti. *Categorical Data Analysis*. Wiley, New York, 1990.

E. L. Allwein, R. E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001. ISSN 1533-7928.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

R. A. Bradley and M. Terry. The rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324–345, 1952.

G. W. Brier. Verification of forecasts expressed in probabilities. *Monthly Weather Review*, 78:1–3, 1950.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

H. A. David. *The method of paired comparisons*. Oxford University Press, New York, second edition, 1988.

R. R. Davidson and P. H. Farquhar. A bibliography on the method of paired comparisons. *Biometrics*, 32:241–252, 1976.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. URL `citeseer.ist.psu.edu/dietterich95solving.html`.

L. R. Jr. Ford. Solution of a ranking problem from binary comparisons. *American Mathematical Monthly*, 64(8):28–33, 1957.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

T.-K. Huang, R. C. Weng, and C.-J. Lin. A generalized Bradley-Terry model: From group competition to individual skill. In *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

D. R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32:386–408, 2004.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. MNIST database available at `http://yann.lecun.com/exdb/mnist/`.

H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. Technical report, Department of Computer Science, National Taiwan University, 2003. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/plattprob.ps`.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. CRC Press, 2nd edition, 1990.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at `http://www.ncc.up.pt/liacc/ML/statlog/datasets.html`.

R. N. Pendergrass and R. A. Bradley. Ranking in triple comparisons. In Ingram Olkin, editor, *Contributions to Probability and Statistics*. Stanford University Press, Stanford, CA, 1960.

R. L. Placket. The analysis of permutations. *Applied Statistics*, 24:193–202, 1975.

J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press. URL `citeseer.nj.nec.com/platt99probabilistic.html`.

P. V. Rao and L. L. Kupper. Ties in paired-comparison experiments: A generalization of the Bradley-Terry model. *Journal of the American Statistical Association*, 62:194–204, 1967. [Corrigendum J. Amer. Statist. Assoc. 63 1550-1551].

G. Simons and Y.-C. Yao. Asymptotics when the number of parameters tends to infinity in the Bradley-Terry model for paired comparisons. *The Annals of Statistics*, 27(3): 1041–1060, 1999.

T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob/svmprob.pdf`.

B. Zadrozny. Reducing multiclass to binary by coupling probability estimates. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1041–1048. MIT Press, Cambridge, MA, 2002.

E. Zermelo. Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29:436–460, 1929.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–134, 2004.

Journal of Machine Learning Research 7 (2006) 117–139 Submitted 9/05; Revised 12/05; Published 1/06

Bounds for Linear Multi-Task Learning

Andreas Maurer ANDREASMAURER@COMPUSERVE.COM
Adalbertstrasse 55
D-80799 München, Germany

Editor: Nello Cristianini

Abstract

We give dimension-free and data-dependent bounds for linear multi-task learning where a common linear operator is chosen to preprocess data for a vector of task specific linear-thresholding classifiers. The complexity penalty of multi-task learning is bounded by a simple expression involving the margins of the task-specific classifiers, the Hilbert-Schmidt norm of the selected preprocessor and the Hilbert-Schmidt norm of the covariance operator for the total mixture of all task distributions, or, alternatively, the Frobenius norm of the total Gramian matrix for the data-dependent version. The results can be compared to state-of-the-art results on linear single-task learning.

Keywords: learning to learn, transfer learning, multi-task learning

1. Introduction

Simultaneous learning of different tasks under some common constraint, often called *multi-task learning*, has been tested in practice with good results under a variety of different circumstances (see Baxter 1998, Caruana 1998, Thrun 1998, Ando and Zhang 2005). The technique has been analyzed theoretically and in some generality by Baxter (2000) and Ando and Zhang (2005). The latter reference appears to be the first to use Rademacher averages in this context. The purpose of this paper is to improve some of these theoretical results in a special case of practical importance, when input data are represented in a linear, potentially infinite dimensional space, and the common constraint is a linear preprocessor.

Finite systems provide simple examples illustrating the potential advantages of multi-task learning. Consider agnostic learning with an input space $\mathcal{X}$ and a finite set $\mathcal{F}$ of hypotheses $f : \mathcal{X} \to \{0,1\}$. For a hypothesis $f \in \mathcal{F}$ let $\mathrm{er}(f)$ be the expected error and $\hat{\mathrm{er}}(f)$ the empirical error on a training sample S of size n (drawn iid from the underlying task distribution) respectively. Combining Hoeffding's inequality with a union bound one shows (see e.g. Anthony and Bartlett 1999), that with probability greater than $1-\delta$ we have for every $f \in \mathcal{F}$ the error bound

$$\mathrm{er}(f) \leq \hat{\mathrm{er}}(f) + \frac{1}{\sqrt{2n}}\sqrt{\ln|\mathcal{F}| + \ln(1/\delta)}. \tag{1}$$

Suppose now that there are a set $\mathcal{Y}$, a finite but large set $\mathcal{G}$ of preprocessors $g : \mathcal{X} \to \mathcal{Y}$, and another set $\mathcal{H}$ of classifiers $h : \mathcal{Y} \to \{0,1\}$ with $|\mathcal{H}| \ll |\mathcal{F}|$. For a cleverly chosen preprocessor $g \in \mathcal{G}$ it will likely be the case that we find some $h \in \mathcal{H}$ such that $h \circ g$ has the same or even a smaller empirical error than the best $f \in \mathcal{F}$. But this will lead to an improvement of the bound above (replacing $|\mathcal{F}|$ by $|\mathcal{H}|$) only if we choose g before seeing the data, otherwise we incur a large estimation penalty for the selection of g (replacing $|\mathcal{F}|$ by $|\mathcal{H} \circ \mathcal{G}|$).

The situation is improved if we have a set of m different learning tasks with corresponding task distributions and samples $S_1,...,S_m$, each of size n and drawn iid from the corresponding distributions. We now consider solutions $h_1 \circ g,... \; h_m \circ g$ for each of the m tasks where the preprocessing map $g \in \mathcal{G}$ is *constrained to be the same for all tasks* and only the $h_l \in \mathcal{H}$ specialize to each task l at hand. Again Hoeffding's inequality and a union bound imply that with probability greater $1-\delta$ we have for all $(h_1,...,h_m) \in \mathcal{H}^m$ and every $g \in \mathcal{G}$

$$\frac{1}{m}\sum_{l=1}^{m} \mathrm{er}^l (h_l \circ g) \leq \frac{1}{m}\sum_{l=1}^{m} \hat{\mathrm{er}}^l (h_l \circ g) + \frac{1}{\sqrt{2n}}\sqrt{\ln \left|\mathcal{H}\right| + \frac{\ln |\mathcal{G}| + \ln (1/\delta)}{m}}. \tag{2}$$

Here $\mathrm{er}^l (f)$ and $\hat{\mathrm{er}}^l (f)$ denote the expected error in task l and the empirical error on training sample S_l respectively. The left hand side above is an average of the expected errors, so that the guarantee implied by the bound is a little weaker than the usual PAC guarantees (but see Ben-David, 2003, for bounds on the individual errors). The first term on the right is the average empirical error, which a multi-task learning algorithm seeks to minimize. We can take it as an operational definition of task-relatedness relative to $(\mathcal{H}, \mathcal{G})$ that we are able to obtain a very small value for this term. The remaining term, which bounds the estimation error, now exhibits the advantage of multi-task learning: Sharing the preprocessor implies sharing its cost of estimation, and the entropy contribution arising from the selection of $g \in \mathcal{G}$ decreases with the number of learning tasks. Since by assumption $\left|\mathcal{H}\right| \ll |\mathcal{F}|$, the estimation error in the multi-task bound (2) can become much smaller than in the single task case (1) if the number m of tasks becomes large.

The choice of the preprocessor $g \in \mathcal{G}$ can also be viewed as the selection of the hypothesis space $\mathcal{H} \circ g$. This leads to an alternative formulation of multi-task learning, where the common object is a hypothesis space chosen from a class of hypothesis spaces (in this case $\left\{\mathcal{H} \circ g : g \in \mathcal{G}\right\}$), and the classifiers for the individual tasks are all chosen from the selected hypothesis space. Here we prefer the functional formulation of selecting a preprocessor instead of a hypothesis space, because it is more intuitive and sufficient in the situations which we consider.

The arguments leading to (2) can be refined and extended to certain infinite classes to give general bounds for multi-task learning (Baxter 2000, Ando and Zhang 2005). In this paper we concentrate on the case where the input space $\mathcal{X}$ is a subset of the unit ball in a Hilbert space H, the class $\mathcal{G}$ of preprocessors is a set $\mathcal{A}$ of bounded symmetric linear operators on H, and the class $\mathcal{H}$ is the set of classifiers h_v obtained by 0-thresholding linear functionals v in H with $\|v\| \leq B$, that is

$$h_v (x) = \mathrm{sign}\left(\langle x, v\rangle\right) \text{ and } h_v \circ T\,(x) = \mathrm{sign}\left(\langle Tx, v\rangle\right), x \in H, T \in \mathcal{A}, \; \|v\| \leq B.$$

The learner now searches for a multi-classifier $\mathbf{h_v} \circ T = (h_{v^1} \circ T, ..., h_{v^m} \circ T)$ where the preprocessing operator $T \in \mathcal{A}$ is the same for all tasks and only the vectors v^l specialize to each task l at hand. The desired multi-classifier $\mathbf{h_v} \circ T$ should have a small value of the average error

$$\mathrm{er}\left(\mathbf{h_v} \circ T\right) = \frac{1}{m}\sum_{l=1}^{m} \mathrm{er}^l (h_{v_l} \circ T) = \frac{1}{m}\sum_{l=1}^{m} \Pr\left\{\mathrm{sign}\left(\left\langle TX^l, v^l\right\rangle\right) \neq Y^l\right\},$$

where X^l and Y^l are the random variables modeling input-values and labels for the l-th task. To guide this search we look for bounds on $\mathrm{er}(\mathbf{h_v} \circ T)$ in terms of the total observed data for all tasks, valid uniformly for all $\mathbf{v} = \left(v^1, ..., v^m\right)$ with $\left\|v^l\right\| \leq B$ and all $T \in \mathcal{A}$. We will prove the following :

Theorem 1 *Let $\delta \in (0,1)$. With probability greater than $1-\delta$ it holds for all $\mathbf{v} = \left(v^1,...,v^m\right) \in H$ with $\left\|v^l\right\| \leq 1$ and all bounded symmetric operators T on H with $\|T\|_{HS} \geq 1$, and for all $\gamma \in (0,1)$ that*

$$er\left(\mathbf{h}_{\mathbf{v}} \circ T\right) \leq \hat{er}_{\gamma}(\mathbf{v} \circ T) + \frac{8\|T\|_{HS}}{\gamma\sqrt{n}}\sqrt{\|C\|_{HS} + \frac{1}{m}} + \sqrt{\frac{\ln\frac{4\|T\|_{HS}}{\delta\gamma}}{2nm}}.$$

Here $\hat{\mathrm{er}}_{\gamma}(\mathbf{v} \circ T)$ is a margin-based empirical error estimate, bounded by the relative number of examples $\left(X_i^l, Y_i^l\right)$ in the total training sample for all tasks l, where $Y_i^l \left\langle TX_i^l, v^l \right\rangle < \gamma$ (see section 4).

The quantity $\|T\|_{HS}$ is the *Hilbert-Schmidt norm* of T, defined for symmetric T by

$$\|T\|_{HS} = \left(\sum \lambda_i^2\right)^{1/2},$$

where λ_i is the sequence of eigenvalues of T (counting multiplicities, see section 2).

C is *the total covariance operator* corresponding to the mixture of all the task-input-distributions in H. Since data are constrained to the unit ball in H we always have $\|C\|_{HS} \leq 1$ (see section 3).

The above theorem is the simplest, but not the tightest or most general form of our results. For example the factor 8 on the right hand side can be decreased to be arbitrarily close to 2, thereby incurring only a logarithmic penalty in the last term.

A special case results from restricting the set of candidate preprocessors to $\mathcal{P}_d$, the set of orthogonal projections in H with d-dimensional range. In this case learning amounts to the selection of a d-dimensional subspace M of H and of an m-tuple of vectors v^l in M (components of v^l orthogonal to M are irrelevant to the projected data). All operators $T \in \mathcal{P}_d$ satisfy $\|T\|_{HS} = \sqrt{d}$, which can then be substituted in the above bound. Identifying such a projection with the structural parameter Θ, this corresponds to the case considered by Ando and Zhang (2005), where a practical algorithm for this type of multi-task learning is presented. The identity $\|\Theta\|_{HS} = \sqrt{d}$ then expresses the regularization condition mentioned in (Ando, Zhang 2005).

The bound in the above theorem is dimension free, it does not require the data distribution in H to be confined to a finite dimensional subspace. Almost to the contrary: Suppose that the input data are distributed uniformly on $M \cap S_1$ where M is a k-dimensional subspace in H and S_1 is the sphere consisting of vectors with unit norm in H. Then C has the k-fold eigenvalue $1/k$, the remaining eigenvalues being zero. Therefore $\|C\|_{HS} = 1/\sqrt{k}$, so part of the bound above decreases to zero as the dimensionality of the data-distribution increases, in contrast to the bound in (Ando, Zhang, 2005), which increases linearly in k. The fact that our bounds are dimension free allows their general use for multi-task learning in kernel-induced Hilbert spaces (see Cristianini and Shawe-Taylor 2000).

If we compare the second term on the right hand side to the estimation error bound in (2), we can recognize a certain similarity: Loosely speaking we can identify $\|T\|_{HS}^2/m$ with the cost of estimating the operator T, and $\|T\|_{HS}^2\|C\|_{HS}$ with the cost of finding the linear classifiers $v_1,...,v_m$. The order of dependence on the number of tasks m is the same in Theorem 1 as in (2).

In the limit $m \to \infty$ it is preferable to use a different bound (see Theorems 13 and 15), at the expense of slower convergence in m. The main inequality of the theorem then becomes

$$\mathrm{er}\left(\mathbf{h}_{\mathbf{v}} \circ T\right) \leq \hat{\mathrm{er}}_{\gamma}(\mathbf{v} \circ T) + \frac{2\left\|T^2\right\|_{HS}^{1/2}}{(1-\varepsilon)^2\gamma\sqrt{n}}\left(\|C\|_{HS}^2 + \frac{3}{m}\right)^{1/4} + \sqrt{\frac{\ln\frac{\left\|T^2\right\|_{HS}^{1/2}}{\delta\gamma\varepsilon^2}}{2nm}}. \tag{3}$$

for some very small $\varepsilon > 0$ to be fixed in advance. If T is an orthogonal projection with d-dimensional range then $\left\|T^2\right\|_{HS}^{1/2} = d^{1/4}$, so for a large number of tasks m the bound on the estimation error becomes approximately

$$\frac{2d^{1/4}\left\|C\right\|_{HS}^{1/2}}{\gamma\sqrt{n}}.$$

One of the best dimension-free bounds for linear single-task learning (see e.g. Bartlett and Mendelson 2002, or Lemma 11 below) would give $2/\left(\gamma\sqrt{n}\right)$ for this term, if all data are constrained to unit vectors. We therefore expect superior estimation for multi-task learning of d-dimensional projections with large m, whenever $d^{1/4}\left\|C\right\|_{HS}^{1/2} \ll 1$. If we again assume the data-distribution to be uniform on $M \cap S_1$ with M a k-dimensional subspace, this is the case whenever $d \ll k$, that is, qualitatively speaking, whenever the dimension of the utilizable part of the data is considerably smaller than the dimension of the total data distribution.

The results stated above give some insights, but they have the practical disadvantage of being unobservable, because they depend on the properties of the covariance operator C, which in turn depends on an unknown data distribution. One way to solve this problem is using the fact that the finite-sample approximations to the covariance operator have good concentration properties (see Theorem 8 below). The corresponding result is:

Theorem 2 *With probability greater than $1-\delta$ in the sample $\mathbf{X}$ it holds for all $v_1,...,v_m \in H$ with $\|v_l\| \leq 1$ and all bounded symmetric operators T on H with $\|T\|_{HS} \geq 1$, and for all $\gamma \in (0,1)$ that*

$$er\left(\mathbf{h}_{\mathbf{v}} \circ T\right) \leq \hat{er}_{\gamma}\left(\mathbf{v} \circ T\right) + \frac{8\left\|T\right\|_{HS}}{\gamma\sqrt{n}}\sqrt{\frac{1}{mn}\left\|\hat{C}\left(\mathbf{X}\right)\right\|_{Fr} + \frac{1}{m}} + \sqrt{\frac{9\ln\frac{8\|T\|_{HS}}{\delta\gamma}}{2nm}}.$$

where the $\left\|\hat{C}\left(\mathbf{X}\right)\right\|_{Fr}$ is the Frobenius norm of the gramian.

By definition

$$\left\|\hat{C}\left(\mathbf{X}\right)\right\|_{Fr} = \left(\sum_{l,r,i,j}\left\langle X_i^l, X_j^r\right\rangle^2\right)^{1/2}.$$

Here X_i^l is the random variable describing the i-th data point in the sample corresponding to the l-th task. The corresponding label Y_i^l enters only in the empirical margin error. The quantity $(mn)^{-1}\left\|\hat{C}\left(\mathbf{X}\right)\right\|_{Fr}$ can be regarded as an approximation to $\left\|C\right\|_{HS}$, valid with high probability, so that Theorem 2 is a sample based version of Theorem 1.

In section 2 we introduce the necessary terminology and results on Hilbert-Schmidt operators and in section 3 the covariance operator of random elements in a Hilbert space. Section 4 gives a formal definition of multi-task systems and a general PAC bound in terms of Rademacher complexities. For the readers benefit a proof of this bound is given in an appendix, where we follow the path prepared by Kolchinskii and Panchenko (2002) and Bartlett and Mendelson (2002). In section 5 we study the Rademacher complexities of linear multi-task systems. In section 6 we give bounds for non-interacting systems, which are essentially equivalent to single-task learning, and derive bounds for proper, interacting multi-task learning, including the above mentioned results. We conclude with

the construction of an example to demonstrate the advantages of multi-task learning. The appendix contains missing proofs and a convenient reference-table to the notation and definitions introduced in the paper.

2. Hilbert-Schmidt Operators

For a fixed real, separable Hilbert space H, with inner product $\langle .,. \rangle$ and norm $\|.\|$, we define a second real, separable Hilbert space consisting of *Hilbert-Schmidt operators*. With HS we denote the real vector space of operators T on H satisfying $\sum_{i=1}^{\infty} \|Te_i\|^2 \leq \infty$ for every orthonormal basis $(e_i)_{i=1}^{\infty}$ of H. Every $T \in HS$ is bounded. For $S,T \in HS$ and an orthonormal basis (e_i) the series $\sum_i \langle Se_i, Te_i \rangle$ is absolutely summable and independent of the chosen basis. The number $\langle S,T \rangle_{HS} = \sum \langle Se_i, Te_i \rangle$ defines an inner product on HS, making it into a Hilbert space. We denote the corresponding norm with $\|.\|_{HS}$ in contrast to the usual operator norm $\|.\|_{\infty}$. See Reed and Simon (1980) for background on functional analysis). We use HS^* to denote the set of symmetric Hilbert-Schmidt operators. For every member of HS^* there is a complete orthonormal basis of eigenvectors, and for $T \in HS^*$ the norm $\|T\|_{HS}$ is just the ℓ_2-norm of its sequence of eigenvalues. With HS^+ we denote the members of HS^* with only nonnegative eigenvalues.

We use two simple maps from H or H^2 to HS to relate the geometries of objects in H to the geometry in HS.

Definition 3 *Let $x,y \in H$. We define two operators Q_x and $G_{x,y}$ on H by*

$$\begin{aligned} Q_x z &= \langle z,x \rangle x, \forall z \in H \\ G_{x,y} z &= \langle x,z \rangle y, \forall z \in H. \end{aligned}$$

We will frequently use parts of the following lemma, the proof of which is very easy.

Lemma 4 *Let $x,y,x',y' \in H$ and $T \in HS$. Then*
(i) $Q_x \in HS^+$ and $\|Q_x\|_{HS} = \|x\|^2$.
(ii) $\langle Q_x, Q_y \rangle_{HS} = \langle x,y \rangle^2$.
(iii) $\langle T, Q_x \rangle_{HS} = \langle Tx,x \rangle$.
(iv) $\langle T^*T, Q_v \rangle_{HS} = \|Tv\|^2$.
(v) $Q_y Q_x = \langle x,y \rangle G_{x,y}$.
(vi) $G_{x,y} \in HS$ and $\|G_{x,y}\|_{HS} = \|x\| \|y\|$.
(vii) $\langle G_{x,y}, G_{x',y'} \rangle_{HS} = \langle x,x' \rangle \langle y,y' \rangle$
(viii) $\langle T, G_{x,y} \rangle_{HS} = \langle Tx,y \rangle$.
(ix) For $\alpha \in \mathbb{R}$, $Q_{\alpha x} = \alpha^2 Q_x$.

Proof For $x = 0$ (iii) is obvious. For $x \neq 0$ choose an orthonormal basis $(e_i)_1^{\infty}$, so that $e_1 = x/\|x\|$. Then e_1 is the only nonzero eigenvector of Q_x with eigenvalue $\|x\|^2 > 0$. Also

$$\langle T, Q_x \rangle_{HS} = \sum_i \langle Te_i, Q_x e_i \rangle = \langle Tx, Q_x x \rangle / \|x\|^2 = \langle Tx,x \rangle,$$

which gives (iii). (ii), (i) and (iv) follow from substitution of Q_y, Q_x and T^*T respectively for T. (v) follows directly from the definition when applied to any $z \in H$. Let $(e_k)_{k=1}^{\infty}$ be any orthonormal

basis. Then $x = \sum_k \langle x, e_k\rangle e_k$, so by boundedness of T

$$\begin{aligned}\langle Tx, y\rangle &= \left\langle T\sum_k \langle x, e_k\rangle e_k, y\right\rangle = \sum_k \langle Te_k, \langle x, e_k\rangle y\rangle = \sum_k \langle Te_k, G_{x,y}e_k\rangle \\ &= \langle T, G_{x,y}\rangle_{HS},\end{aligned}$$

which is (viii). Similarly

$$\begin{aligned}\left\langle G_{x,y}, G_{x',y'}\right\rangle_{HS} &= \sum_k \left\langle \langle x, e_k\rangle y, \left\langle x', e_k\right\rangle y'\right\rangle = \left\langle y, y'\right\rangle \sum_k \langle x, e_k\rangle \left\langle x', e_k\right\rangle \\ &= \left\langle x, x'\right\rangle \left\langle y, y'\right\rangle,\end{aligned}$$

which gives (vii) and (vi). (ix) is obvious. ■

The following application of Lemma 4 is the key to our main results.

Lemma 5 *Let $T \in HS$ and $w_1, \ldots, w_m$ and $v_1, \ldots, v_m$ vectors in H with $\|v_i\| \le B$. Then*

$$\sum_{l=1}^m \langle Tw_l, v_l\rangle \le B\|T\|_{HS}\left(\sum_{l,r}|\langle w_l, w_r\rangle|\right)^{1/2}$$

and

$$\sum_{l=1}^m \langle Tw_l, v_l\rangle \le Bm^{1/2}\|T^*T\|_{HS}^{1/2}\left(\sum_{l,r}\langle w_l, w_r\rangle^2\right)^{1/4}$$

Proof Without loss of generality assume $B = 1$. Using Lemma 4 (viii), Schwarz' inequality in HS and Lemma 4 (vii) we have

$$\begin{aligned}\sum_{l=1}^m \langle Tw_l, v_l\rangle &= \left\langle T, \sum_{l=1}^m G_{w_l,v_l}\right\rangle_{HS} \le \|T\|_{HS}\left\|\sum_{l=1}^m G_{w_l,v_l}\right\|_{HS} \\ &= \|T\|_{HS}\left(\sum_{l,r}^m \langle w_l, w_r\rangle\langle v_l, v_r\rangle\right)^{1/2} \\ &\le \|T\|_{HS}\left(\sum_{l,r}^m |\langle w_l, w_r\rangle|\right)^{1/2}.\end{aligned}$$

This proves the first inequality. Also, using Schwarz' inequality in H and $\mathbb{R}^m$, Lemma 4 (iv) and Schwarz' inequality in HS

$$\begin{aligned}\sum_{l=1}^m \langle Tw_l, v_l\rangle &\le \left(\sum_{l=1}^m \|v_l\|^2\right)^{1/2}\left(\sum_{l=1}^m \|Tw_l\|^2\right)^{1/2} \le \sqrt{m}\left\langle T^*T, \sum_{l=1}^m Q_{w_l}\right\rangle_{HS}^{1/2} \\ &\le \sqrt{m}\|T^*T\|_{HS}^{1/2}\left\|\sum_{l=1}^m Q_{w_l}\right\|_{HS}^{1/2} = \sqrt{m}\|T^*T\|_{HS}^{1/2}\left(\sum_{l,r}\langle w_l, w_r\rangle^2\right)^{1/4}\end{aligned}$$

■

The set of d-dimensional, orthogonal projections in H is denoted with $\mathcal{P}_d$. We have $\mathcal{P}_d \subset HS^*$ and if $P \in \mathcal{P}_d$ then $\|P\|_{HS} = \sqrt{d}$ and $P^2 = P$.

An operator T is called *trace-class* if $\sum_{i=1}^{\infty} \langle Te_i, e_i \rangle$ is an absolutely convergent series for every orthonormal basis $(e_i)_{i=1}^{\infty}$ of H. In this case the number $tr(T) = \sum_{i=1}^{\infty} \langle Te_i, e_i \rangle$ is called the *trace* of T and it is independent of the chosen basis.

If $\mathcal{A} \subset HS^*$ is a set of symmetric and bounded operators in H we use the notation

$$\|\mathcal{A}\|_{HS} = \sup \{\|T\|_{HS} : T \in \mathcal{A}\} \text{ and } \mathcal{A}^2 = \left\{T^2 : T \in \mathcal{A}\right\}.$$

3. Vector- and Operator-Valued Random Variables

Let (Ω, Σ, μ) be a probability space with expectation $E[F] = \int_\Omega F d\mu$ for a random variable $F : \Omega \to \dot{\mathbb{R}}$. Let X be a random variable with values in H, such that $E[\|X\|] < \infty$. The linear functional $v \in H \mapsto E[\langle X, v \rangle]$ is bounded by $E[\|X\|]$ and thus defines (by the Riesz Lemma) a unique vector $E[X] \in H$ such that $E[\langle X, v \rangle] = \langle E[X], v \rangle, \forall v \in H$, with $\|E[X]\| \leq E[\|X\|]$.

We now look at the random variable Q_X, with values in HS. Suppose that $E\left[\|X\|^2\right] < \infty$. Passing to the space HS of Hilbert-Schmidt operators the above construction can be carried out again: By Lemma 4 (i) $E[\|Q_X\|_{HS}] = E\left[\|X\|^2\right] < \infty$, so there is a unique operator $E[Q_X] \in HS$ such that $E[\langle Q_X, T \rangle_{HS}] = \langle E[Q_X], T \rangle_{HS}, \forall T \in HS$.

Definition 6 *The operator $E[Q_X]$ is called the covariance operator of X.*

Lemma 7 *The covariance operator $E[Q_X] \in HS^+$ has the following properties.*
(i) $\|E[Q_X]\|_{HS} \leq E[\|Q_X\|_{HS}]$.
(ii) $\langle E[Q_X]y, z \rangle = E[\langle y, X \rangle \langle z, X \rangle], \forall y, z \in H$.
(iii) $tr(E[Q_X]) = E\left[\|X\|^2\right]$.
(iv) For H-valued independent X_1 and X_2 with $E\left[\|X_i\|^2\right] \leq \infty$ we have

$$\langle E[Q_{X_1}], E[Q_{X_2}] \rangle_{HS} = E\left[\langle X_1, X_2 \rangle^2\right].$$

(v) Under the same hypotheses, if $E[X_2] = 0$ then

$$E[Q_{X_1+X_2}] = E[Q_{X_1}] + E[Q_{X_2}]$$

Proof (i) follows directly from the construction, (iv) from the identity $\langle E[Q_{X_1}], E[Q_{X_2}] \rangle_{HS} = E\left[\langle Q_{X_1}, Q_{X_2} \rangle_{HS}\right]$. Let $y, z \in H$. Then using 4 (viii) we get

$$\begin{aligned} \langle E[Q_X]y, z \rangle &= \langle E[Q_X], G_{y,z} \rangle_{HS} = E\left[\langle Q_X, G_{y,z} \rangle_{HS}\right] = E[\langle Q_X y, z \rangle] \\ &= E[\langle y, X \rangle \langle z, X \rangle] \end{aligned}$$

and hence (ii). We have with orthonormal basis $(e_k)_{k=1}^{\infty}$ and using (ii)

$$tr(E[Q_X]) = \sum_k \langle E[Q_X]e_k, e_k \rangle = \sum_k E[\langle e_k, X \rangle \langle e_k, X \rangle] = E\left[\|X\|^2\right],$$

which gives (iii). Substitution of an eigenvector v for both y and z in (ii) shows that the corresponding eigenvalue must be nonnegative, so $E[Q_X] \in HS^+$.

Finally (v) holds because for any $y,z \in H$ we have, using independence and the mean-zero condition for X_2, that

$$\begin{aligned}
&\langle E[Q_{X_1+X_2}]y,z\rangle \\
&= E[\langle y,X_1+X_2\rangle\langle X_1+X_2,z\rangle] \\
&= E[\langle y,X_1\rangle\langle X_1,z\rangle] + E[\langle y,X_2\rangle\langle X_2,z\rangle] + E[\langle y,X_1\rangle\langle X_2,z\rangle] + E[\langle y,X_2\rangle\langle X_1,z\rangle] \\
&= \langle (E[Q_{X_1}]+E[Q_{X_2}])y,z\rangle + \langle y,E[X_1]\rangle\langle E[X_2],z\rangle + \langle y,E[X_2]\rangle\langle E[X_1],z\rangle \\
&= \langle (E[Q_{X_1}]+E[Q_{X_2}])y,z\rangle
\end{aligned}$$

■

Property (ii) above is sometimes taken as the defining property of the covariance operator (see Baxendale 1976).

If X is distributed uniformly on $M \cap S_1$, where M is a k-dimensional subspace and S_1 the unit sphere in H, then $E\left[\langle X,y\rangle^2\right] = \langle E[Q_X]y,y\rangle$ is zero if and only if $y \in M^\perp$, so the range of $E[Q_X]$ is M, so there are exactly k-eigenvectors corresponding to non-zero eigenvalues of $E[Q_X]$. By symmetry these eigenvalues must all be equal, and by (iii) above they sum up to 1, so $E[Q_X]$ has the k-fold eigenvalue $1/k$, with zero as the only other eigenvalue. It follows that $\|E[Q_X]\|_{HS} = 1/\sqrt{k}$. We have given this derivation to illustrate the tendency of the Hilbert-Schmidt norm of the covariance operator of a distribution concentrated on unit vectors to decrease with the effective dimensionality of the distribution. This idea is relevant to the interpretation of our results.

The fact that HS is a separable Hilbertspace just as H allows to define an operator $E[T]$ whenever T is a random variable with values in HS and $E[\|T\|_{HS}] < \infty$. Also any result valid in H has a corresponding analogue valid in HS. We quote a corresponding operator-version of a Theorem of Cristianini and Shawe-Taylor (2004) on the concentration of independent vector-valued random variables.

Theorem 8 *Suppose that $T_1,\ldots,T_m$ are independent random variables in H with $\|T_i\| \leq 1$. Then for all $\delta > 0$ with probability greater than δ we have*

$$\left\| \frac{1}{m}\sum_{i=1}^m E[T_i] - \frac{1}{m}\sum_{i=1}^m T_i \right\|_{HS} \leq \frac{2}{\sqrt{m}}\left(1+\sqrt{\frac{\ln(1/\delta)}{2}}\right).$$

Apply this with $T_i = Q_{X_i}$ where the X_i are iid H-valued with $\|X_i\| \leq 1$. The theorem then shows that the covariance operator $E[Q_X]$ can be approximated in HS-norm with high probability by the empirical estimates $(1/m)\sum_i Q_{X_i}$. The quantity

$$\left\| \sum_i Q_{X_i} \right\|_{HS} = \left(\sum_{i,j} \langle X_i,X_j\rangle^2 \right)^{1/2}$$

is the Frobenius norm of the Gramian (or kernel-) matrix $\hat{C}(\mathbf{X})_{ij} = \langle X_i,X_j\rangle$, denoted $\left\|\hat{C}(\mathbf{X})\right\|_{Fr}$. An immediate corollary to the above is, that $(1/m)\left\|\hat{C}(\mathbf{X})\right\|_{Fr}$ is with high probability a good approximation of $\|E[Q_X]\|_{HS}$. In the proof of Theorem 2 we will not need this fact however.

4. Multi-Task Problems and General Bounds

For our discussion of multi-task learning we concentrate on binary labeled data. Let (Ω, Σ, μ) be a probability space. We assume that there is a *multi-task problem* modeled by m independent random variables $Z^l = \left(X^l, Y^l\right) : \Omega \to \mathcal{X} \times \{-1,1\}$, where

- $l \in \{1,...,m\}$ identifies one of the m learning tasks,
- X^l models the input data of the l-th task, distributed in a set $\mathcal{X}$, called the *input space*.
- $Y^l \in \{-1,1\}$ models the output-, or label-data of the l-th task.
- For each $l \in \{1,...,m\}$ there is an n-tuple of independent random variables $\left(Z_i^l\right)_{i=1}^n = \left(X_i^l, Y_i^l\right)_{i=1}^n$, where each Z_i^l is identically distributed to Z^l.

The random variable $\mathbf{Z} = \left(Z_i^l\right)_{(i,l)=(1,1)}^{(n,m)}$ is called the *training sample* or training data. We also write $\mathbf{X} = \left(X_i^l\right)_{(i,l)=(1,1)}^{(n,m)}$. We use the superscript ^{l} to identify learning tasks running from 1 to m, the subscript $_i$ to identify data points in the sample, running from 1 to n. We will use the notations $\mathbf{x} = \left(x_i^l\right)_{(i,l)=(1,1)}^{(n,m)}$ for generic members of $(\mathcal{X}^n)^m$ and $\mathbf{z} = \left(z_i^l\right)_{(i,l)=(1,1)}^{(n,m)} = (\mathbf{x}, \mathbf{y}) = \left(x_i^l, y_i^l\right)_{(i,l)=(1,1)}^{(n,m)}$ for generic members of $((\mathcal{X} \times \{-1,1\})^n)^m$.

A *multiclassifier* is a map $\mathbf{h} : \mathcal{X} \to \{-1,1\}^m$. We write $\mathbf{h} = \left(h^1, ..., h^m\right)$ and interpret $h^l(x)$ as the label assigned to the vector x when the task is known to be l. The average error of a multiclassifier $\mathbf{h}$ is the quantity

$$\mathrm{er}(\mathbf{h}) = \frac{1}{m} \sum_{l=1}^{m} \Pr\left\{h^l\left(X^l\right) \neq Y^l\right\},$$

which is just the misclassification probability averaged over all tasks. Typically a classifier is chosen from some candidate set minimizing some error estimate based on the training data $\mathbf{Z}$. Here we consider zero-threshold classifiers $\mathbf{h_f}$ which arise as follows:

Suppose that $\mathcal{F}$ is a class of vector valued functions $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ with $\mathbf{f} = \left(f^1, ..., f^m\right)$. A function $\mathbf{f} \in \mathcal{F}$ defines a multi-classifier $\mathbf{h_f} = \left(h_\mathbf{f}^1, ..., h_\mathbf{f}^m\right)$ through $h_\mathbf{f}^l(x) = \mathrm{sign}\left(f^l(x)\right)$. To give uniform error bounds for such classifiers in terms of empirical estimates, we define for $\gamma > 0$ the margin functions

$$\phi_\gamma(t) = \begin{cases} 1 & \text{if} \quad t \leq 0 \\ 1 - t/\gamma & \text{if} \quad 0 < t < \gamma \\ 0 & \text{if} \quad \gamma \leq t \end{cases},$$

and for $\mathbf{f} \in \mathcal{F}$ the random variable

$$\hat{\mathrm{er}}_\gamma(\mathbf{f}) = \frac{1}{mn} \sum_{l=1}^{m} \sum_{i=1}^{n} \phi_\gamma\left(Y_i^l f^l\left(X_i^l\right)\right),$$

called the *empirical γ-margin error* of $\mathbf{f}$. The following Theorem gives a bound on $\mathrm{er}(\mathbf{h_f})$ in terms of $\hat{\mathrm{er}}_\gamma(\mathbf{f})$, valid with high probability uniformly in $\mathbf{f} \in \mathcal{F}$ and γ.

Theorem 9 *Let* $\varepsilon, \delta \in (0,1)$

(i) With probability greater than $1-\delta$ *it holds for all* $\mathbf{f} \in \mathcal{F}$ *and all* $\gamma \in (0,1)$ *that*

$$er(\mathbf{h_f}) \leq \hat{er}_\gamma(\mathbf{f}) + \frac{1}{\gamma(1-\varepsilon)} E\left[\hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X})\right] + \sqrt{\frac{\ln(1/(\delta\gamma\varepsilon))}{2nm}}.$$

(ii) With probability greater than $1-\delta$ *it holds for all* $\mathbf{f} \in \mathcal{F}$ *and all* $\gamma \in (0,1)$ *that*

$$er(\mathbf{h_f}) \leq \hat{er}_\gamma(\mathbf{f}) + \frac{1}{\gamma(1-\varepsilon)} \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X}) + \sqrt{\frac{9\ln(2/(\delta\gamma\varepsilon))}{2nm}}.$$

Here $\hat{\mathcal{R}}_n^m(\mathcal{F})$ is the *empirical Rademacher complexity* in the sense of the following

Definition 10 *Let* $\left\{\sigma_i^l : l \in \{1,...,m\}, i \in \{1,...,n\}\right\}$ *be a collection of independent random variables, distributed uniformly in* $\{-1,1\}$*. The empirical Rademacher complexity of a class* $\mathcal{F}$ *of functions* $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ *is the function* $\hat{\mathcal{R}}_n^m(\mathcal{F})$ *defined on* $(\mathcal{X}^n)^m$ *by*

$$\hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{x}) = E_\sigma\left[\sup_{\mathbf{f}\in\mathcal{F}} \frac{2}{mn} \sum_{l=1}^{m}\sum_{i=1}^{n} \sigma_i^l f^l\left(x_i^l\right)\right].$$

For the readers convenience we give a proof of Theorem 9 in the appendix.

The bounds in the Theorem each involve three terms. The last one expresses the dependence of the estimation error on the confidence parameter δ and a model-selection penalty $\ln(1/(\gamma\varepsilon))$ for the choice of the margin γ. Note that it generally decreases as $1/\sqrt{nm}$. This is not an a priori advantage of multi-task learning, but a trivial consequence of the fact that we estimate an average of m probabilities (in contrast to Ben David, 2003, where bounds are valid for each individual task - of course under more restrictive assumptions). The $1/\sqrt{nm}$ decay however implies that even for moderate values of m and n the parameter ε in Theorem 9 can be chosen very small, so that the factor $1/(1-\varepsilon)$ in the second term on the right of the two bounds is very close to unity.

The second term involves the complexity of the function class $\mathcal{F}$, either as measured in terms of the distribution of the random variable $\mathbf{X}$ or in terms of the observed sample. Since the distribution of $\mathbf{X}$ is unobservable in practice, the bound (i) is primarily of theoretical importance, while (ii) can be used to drive an algorithm which selects the multi-classifier $\mathbf{h}_{\mathbf{f}^*}$, where $(\mathbf{f}^*, \gamma) \in \mathcal{F} \times (0,1)$ are chosen to minimize the right side of the bound with given δ, ε. It is questionable if minimizing upper bounds is a good strategy, but it can serve as a motivating guideline.

Of key importance in the analysis of these algorithms is the empirical Rademacher complexity $\hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X})$, as observed on the sample $\mathbf{X}$, and its expectation, measuring respectively the sample- and distribution-dependent complexities of the function class $\mathcal{F}$. Bounds on these quantities can be substituted in Theorem 9 to give corresponding error bounds.

5. The Rademacher Complexity of Linear Multi-Task Learning

We will now concentrate on multi-task learning in the linear case, when the data live in a real, separable Hilbert space H, by means of some kernel-induced embedding (see Cristianini and Shawe-Taylor 2000), the details of which will not concern us at this point. We therefore take H as input space $\mathcal{X}$, so that the random variables X^l take values in H for all $l \in \{1,...,m\}$, and we generally

require $\left\|X^l\right\| \leq 1$. The case $\left\|X^l\right\| = 1$ where the data are constrained to the unit sphere in H is of particular interest, corresponding to a class of radial basis function kernels.

We write C^l for the covariance operator $E\left[Q_{X^l}\right]$ and C for the total covariance operator $C = (1/m)\sum_l C^l$, corresponding to a uniform mixture of distributions. By Lemma 7 we have $\left\|C^l\right\|_{HS} \leq tr\left(C^l\right) = E\left[\left\|X^l\right\|^2\right] \leq 1$.

Let $B > 0$, let T be a fixed symmetric, bounded linear operator on H with $\|T\|_\infty \leq 1$, and let $\mathcal{A}$ be a set of symmetric, bounded linear operators T on H, all satisfying $\|T\|_\infty \leq 1$. We will consider the vector-valued function classes

$$\begin{aligned}
\mathcal{F}_B &= \{x \in H \mapsto (v_1,...,v_m)(x) := (\langle x, v_1\rangle,...,\langle x, v_m\rangle) : \|v_i\| \leq B\} \\
\mathcal{F}_B \circ T &= \{x \in H \mapsto (v_1,...,v_m) \circ T(x) := (\langle Tx, v_1\rangle,...,\langle Tx, v_m\rangle) : \|v_i\| \leq B\} \\
\mathcal{F}_B \circ \mathcal{A} &= \{x \in H \mapsto (v_1,...,v_m) \circ T(x) : \|v_i\| \leq B, T \in \mathcal{A}\}.
\end{aligned}$$

The algorithms which choose from $\mathcal{F}_B$ and $\mathcal{F}_B \circ T$ are essentially trivial extensions of linear single-task learning, where the tasks do not interact in the selection of the individual classifiers v_i, which are chosen independently. In the case of $\mathcal{F}_B \circ T$ the preprocessing operator T is chosen before seeing the training data. Since $\|T\|_\infty \leq 1$ we have $\mathcal{F}_B \circ T \subseteq \mathcal{F}_B$, so that we can expect a reduced complexity for $\mathcal{F}_B \circ T$ and the key question becomes if the choice of T (possibly based on experience with other data) was lucky enough to allow for a sufficiently low empirical error.

The non-interacting classes $\mathcal{F}_B$ and $\mathcal{F}_B \circ T$ are important for comparison to $\mathcal{F}_B \circ \mathcal{A}$ which represents proper multi-task learning. Here the preprocessing operator T is selected from $\mathcal{A}$ in response to the data. The constraint that T be the same for all tasks forces an interaction of tasks in the choice of T and $(v_1,...,v_m)$, deliberately aiming for a low empirical error. At the same time we also have $\mathcal{F}_B \circ \mathcal{A} \subseteq \mathcal{F}_B$, so that again a reduced complexity is to be expected, giving a smaller contribution to the estimation error. The promise of multi-task learning is based on the combination of these two ideas: Aiming for a low empirical error, using a function class of reduced complexity.

We first look at the complexity of the function class $\mathcal{F}_B$. The proof of the following lemma is essentially the same as the proof of Lemma 22 in Bartlett and Mendelson (2002).

Lemma 11 *We have*

$$\begin{aligned}
\hat{\mathcal{R}}_n^m(\mathcal{F}_B)(\mathbf{x}) &\leq \frac{2B}{nm}\sum_{l=1}^m\left(\sum_{i=1}^n\left\|x_i^l\right\|^2\right)^{1/2} \\
E\left[\hat{\mathcal{R}}_n^m(\mathcal{F}_B)(\mathbf{X})\right] &\leq \frac{2B}{\sqrt{n}}\frac{1}{m}\sum_{l=1}^m\left(E\left[\left\|X^l\right\|^2\right]\right)^{1/2} = \frac{2B}{\sqrt{n}}\frac{1}{m}\sum_{l=1}^m tr\left(C^l\right)^{1/2}
\end{aligned}$$

Proof Using Schwarz' and Jensen's inequality and the independence of the σ_i^l we get

$$\begin{aligned}
\hat{\mathcal{R}}_n^m(\mathcal{F}_B)(\mathbf{x}) &= E_\sigma\left[\sup_{v_1,\ldots,v_m,\|v_l\|\le B}\frac{2}{nm}\sum_{l=1}^m\left\langle\sum_{i=1}^n\sigma_i^l x_i^l, v_l\right\rangle\right] \\
&\le BE_\sigma\left[\frac{2}{nm}\sum_{l=1}^m\left\|\sum_{i=1}^n\sigma_i^l x_i^l\right\|\right] \\
&\le \frac{2B}{nm}\sum_{l=1}^m\left(E_\sigma\left[\left\|\sum_{i=1}^n\sigma_i^l x_i^l\right\|^2\right]\right)^{1/2} \\
&= \frac{2B}{nm}\sum_{l=1}^m\left(\sum_{i=1}^n\left\|x_i^l\right\|^2\right)^{1/2}.
\end{aligned}$$

Jensen's inequality then gives the second conclusion ■

The first bound in the lemma is just the average of the bounds given by Bartlett and Mendelson in on the empirical complexities for the various task-components of the sample. For inputs constrained to the unit sphere in H, when $\left\|X^l\right\| = 1$, both bounds become $2B/\sqrt{n}$, which sets the mark for comparison with the interacting case $\mathcal{F}_B \circ \mathcal{A}$. For motivation we next look at the case $\mathcal{F}_B \circ T$, working with a fixed linear preprocessor T of operator norm bounded by 1. Using the above bound we obtain

$$\hat{\mathcal{R}}_n^m(\mathcal{F}_B \circ T)(\mathbf{x}) = \hat{\mathcal{R}}_n^m(\mathcal{F}_B)(T\mathbf{x}) \le \frac{2B}{nm}\sum_{l=1}^m\left(\sum_{i=1}^n\left\|Tx_i^l\right\|^2\right)^{1/2}, \tag{4}$$

which is always bounded by $B/\sqrt{n}$, because $\|Tx\| \le \|x\|, \forall x$. Using Lemma 4 (iv) we can rewrite the right side above as

$$\frac{2B}{\sqrt{n}}\frac{1}{m}\sum_{l=1}^m\left\langle T^2, \frac{1}{n}\sum_{i=1}^n Q_{x_i^l}\right\rangle_{HS}^{1/2}.$$

Taking the expectation and using the concavity of the root function gives, with two applications of Jensen's inequality and an application of Schwarz' inequality (in HS),

$$E\left[\hat{\mathcal{R}}_n^m(\mathcal{F}_B \circ T)(\mathbf{X})\right] \le \frac{2B}{\sqrt{n}}\left\|T^2\right\|_{HS}^{1/2}\|C\|_{HS}^{1/2},$$

which can be significantly smaller than $B/\sqrt{n}$, for example if T is a d-dimensional projection, and the data-distribution is spread well over a much more than d-dimensional submanifold of the unit ball in H, as explained in the introduction and section 3. If we substitute the bound above in Theorem 9 we obtain an inequality which looks like (3) in the limit $m \to \infty$.

We now consider the case where T is chosen from some set $\mathcal{A}$ of (symmetric, bounded) candidate operators on the basis of the same sample $\mathbf{X}$, simultaneous to the determination of the classification vectors $v_1, \ldots, v_l$. We give two bounds each for the Rademacher complexity and its expectation. One is somewhat similar to other bounds for multi-task learning (e.g. (2)) and another one is tighter in the limit when the number of tasks m goes to infinity.

Theorem 12 *The following inequalities hold*

$$\hat{\mathcal{R}}_{\mathbf{x}}^{m}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{x}) \leq \frac{2B\left\|\mathcal{A}\right\|_{HS}}{\sqrt{n}}\sqrt{\frac{1}{mn}\left\|\hat{C}(\mathbf{x})\right\|_{Fr}+\frac{1}{m}} \tag{5}$$

$$\hat{\mathcal{R}}_{\mathbf{x}}^{m}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{x}) \leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{\sqrt{n}}\left(\left(\frac{1}{mn}\left\|\hat{C}(\mathbf{x})\right\|_{Fr}\right)^2+\frac{2}{m}\right)^{1/4} \tag{6}$$

$$E\left[\hat{\mathcal{R}}_{\mathbf{x}}^{m}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{X})\right] \leq \frac{2B\left\|\mathcal{A}\right\|_{HS}}{\sqrt{n}}\sqrt{\left\|C\right\|_{HS}+\frac{1}{m}} \tag{7}$$

$$E\left[\hat{\mathcal{R}}_{\mathbf{x}}^{m}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{X})\right] \leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{\sqrt{n}}\left(\left\|C\right\|_{HS}^2+\frac{3}{m}\right)^{1/4}. \tag{8}$$

Proof Fix $\mathbf{x}$ and define vectors $w_l = w_l(\sigma) = \sum_{i=1}^{n}\sigma_i^l x_i^l$ depending on the Rademacher variables σ_i^l. Then by Lemma 5 and Jensen's inequality

$$\begin{aligned}\hat{\mathcal{R}}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{x}) &= E_\sigma\left[\sup_{T\in\mathcal{A}}\sup_{v_1,\ldots,v_m,\left\|v_i\right\|\leq B}\frac{2}{nm}\sum_{l=1}^{m}\left\langle Tw_l, v_l\right\rangle\right] \\ &\leq \frac{2B}{nm}\left\|\mathcal{A}\right\|_{HS}E_\sigma\left[\left(\sum_{l,r}\left|\left\langle w_l, w_r\right\rangle\right|\right)^{1/2}\right] \\ &\leq \frac{2B}{nm}\left\|\mathcal{A}\right\|_{HS}\left(\sum_{l,r}E_\sigma\left[\left|\left\langle w_l, w_r\right\rangle\right|\right]\right)^{1/2}.\end{aligned} \tag{9}$$

Now we have

$$E_\sigma\left[\left\|w_l\right\|^2\right] = \sum_{i=1}^{n}\sum_{j=1}^{n}E_\sigma\left[\sigma_i^l\sigma_j^l\right]\left\langle x_i^l, x_j^l\right\rangle = \sum_{i=1}^{n}\left\|x_i^l\right\|^2. \tag{10}$$

Also, for $l \neq r$, we get, using Jensen's inequality and independence of the Rademacher variables,

$$\begin{aligned}\left(E_\sigma\left[\left|\left\langle w_l, w_r\right\rangle\right|\right]\right)^2 &\leq E_\sigma\left[\left\langle w_l, w_r\right\rangle^2\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{i'=1}^{n}\sum_{j'=1}^{n}E_\sigma\left[\sigma_i^l\sigma_j^r\sigma_{i'}^l\sigma_{j'}^r\right]\left\langle x_i^l, x_j^r\right\rangle\left\langle x_{i'}^l, x_{j'}^r\right\rangle \\ &= \sum_{i.j=1}^{n}\left\langle x_i^l, x_j^r\right\rangle^2.\end{aligned} \tag{11}$$

Taking the square-root and inserting it together with (10) in (9) we obtain the following intermediate bound

$$\hat{\mathcal{R}}_{\mathbf{x}}^{m}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{x}) \leq \frac{2B\left\|\mathcal{A}\right\|_{HS}}{nm}\left(\sum_{l=1}^{m}\sum_{i=1}^{n}\left\|x_i^l\right\|^2+\sum_{l\neq r}\left(\sum_{i,j=1}^{n}\left\langle x_i^l, x_j^r\right\rangle^2\right)^{1/2}\right)^{1/2} \tag{12}$$

By Jensen's inequality we have

$$\frac{1}{m^2}\sum_{l\neq r}\left(\sum_{i,j=1}^{n}\left\langle x_i^l, x_j^r\right\rangle^2\right)^{1/2} \leq \left(\frac{1}{m^2}\sum_{l,r=1}^{m}\sum_{i,j=1}^{n}\left\langle x_i^l, x_j^r\right\rangle^2\right)^{1/2} = \frac{1}{m}\left\|\hat{C}(\mathbf{x})\right\|_{Fr},$$

which together with (12) and $\left\|x_i^l\right\| \leq 1$ implies (5).

To prove (6) first use the second part of Lemma 5 and Jensen's inequality to get

$$\hat{\mathcal{R}}\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{x}) \leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{n\sqrt{m}} \left(\sum_{l,r} E_\sigma\left[\left\langle w_l, w_r\right\rangle^2\right]\right)^{1/4}. \tag{13}$$

Now we have $E_\sigma\left[\sigma_i^l \sigma_j^l \sigma_{i'}^l \sigma_{j'}^l\right] \leq \delta_{ij}\delta_{i'j'} + \delta_{ii'}\delta_{jj'} + \delta_{ij'}\delta_{ji'}$ so

$$\begin{aligned} E_\sigma\left[\left\langle w_l, w_l\right\rangle^2\right] &\leq \sum_{i,j=1}^n \left(\left\|x_i^l\right\|^2 \left\|x_j^l\right\|^2 + 2\left\langle x_i^l, x_j^l\right\rangle^2\right) \\ &\leq 2\left(\sum_{i=1}^n \left\|x_i^l\right\|^2\right)^2 + \sum_{i,j=1}^n \left\langle x_i^l, x_j^l\right\rangle^2 \leq 2n^2 + \sum_{i,j=1}^n \left\langle x_i^l, x_j^l\right\rangle^2, \end{aligned}$$

where we used $\left\|x_i^l\right\| \leq 1$. Inserting this together with (11) in (13) gives

$$\begin{aligned} \hat{\mathcal{R}}_{\mathbf{u}}^m\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{x}) &\leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{n\sqrt{m}} \left(\sum_{l,r:l\neq r} E_\sigma\left[\left\langle w_l, w_r\right\rangle^2\right] + \sum_{l=1}^m E_\sigma\left[\left\langle w_l, w_l\right\rangle^2\right]\right)^{1/4} \\ &\leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{n\sqrt{m}} \left(\sum_{l,r=1}^m \sum_{i,j=1}^n \left\langle x_i^l, x_j^r\right\rangle^2 + 2mn^2\right)^{1/4}, \end{aligned} \tag{14}$$

which is (6).

Taking the expectation of (12), using Jensen's inequality, $\left\|X^l\right\| \leq 1$ and independence of X^l and X^r for $l \neq r$, and Jensen's inequality again, we get

$$\begin{aligned} &E\left[\hat{\mathcal{R}}_{\mathbf{u}}^m\left(\mathcal{F}_B \circ \mathcal{A}\right)(\mathbf{X})\right] \\ &\leq \frac{2B\left\|\mathcal{A}\right\|_{HS}}{nm}\left(nm + \sum_{l\neq r}\left(E\left[\sum_{i,j=1}^n \left\langle X_i^l, X_j^r\right\rangle^2\right]\right)^{1/2}\right)^{1/2} \\ &= \frac{2B\left\|\mathcal{A}\right\|_{HS}}{nm}\left(\sum_{l\neq r}\left(E\left[\left\langle \sum_{i=1}^n Q_{X_i^l}, \sum_{j=1}^n Q_{X_j^r}\right\rangle_{HS}\right]\right)^{1/2} + nm\right)^{1/2} \\ &= \frac{2B\left\|\mathcal{A}\right\|_{HS}}{\sqrt{n}}\left(\frac{1}{m^2}\sum_{l\neq r}\left\langle E\left[Q_{X^l}\right], E\left[Q_{X^r}\right]\right\rangle_{HS}^{1/2} + \frac{1}{m}\right)^{1/2} \\ &\leq \frac{2B\left\|\mathcal{A}\right\|_{HS}}{\sqrt{n}}\left(\left\langle \frac{1}{m}\sum_{l=1}^m E\left[Q_{X^l}\right], \frac{1}{m}\sum_{r=1}^m E\left[Q_{X^r}\right]\right\rangle_{HS}^{1/2} + \frac{1}{m}\right)^{1/2}, \end{aligned}$$

which gives (7). In a similar way we obtain from (14)

$$\begin{aligned}
&E\left[\hat{\mathcal{R}}_{\mathbf{a}}^{m}\left(\mathcal{F}_B\circ\mathcal{A}\right)(\mathbf{X})\right]\\
&\leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{n\sqrt{m}}\left(mn^2+\sum_{l\neq r}^{m}\sum_{i,j=1}^{n}\left\langle E\left[Q_{X_i^l}\right],E\left[Q_{X_j^r}\right]\right\rangle_{HS}+2mn^2\right)^{1/4}\\
&\leq \frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{n\sqrt{m}}\left(m^2n^2\left\|E\left[\frac{1}{mn}\sum_{l=1}^{m}\sum_{i=1}^{n}Q_{X_i^l}\right]\right\|_{HS}^{2}+3mn^2\right)^{1/4},
\end{aligned}$$

which gives (8) ■

6. Bounds for Linear Multi-Task Learning

Inserting the bounds of Theorem 12 in Theorem 9 immediately gives

Theorem 13 *Let $\mathcal{A}$ be a be set of bounded, symmetric operators in H and $\varepsilon,\delta\in(0,1)$*

(i) With probability greater than $1-\delta$ it holds for all $\mathbf{f}=(v_1,...,v_m)\circ T\in\mathcal{F}_B\circ\mathcal{A}$ and all $\gamma\in(0,1)$ that

$$er\left(\mathbf{h}_{\mathbf{f}}\right)\leq e\hat{r}_{\gamma}\left(\mathbf{f}\right)+\frac{1}{\gamma(1-\varepsilon)}A+\sqrt{\frac{\ln\left(1/\left(\delta\gamma\varepsilon\right)\right)}{2nm}},$$

where A is either

$$A=\frac{2B\left\|\mathcal{A}\right\|_{HS}}{\sqrt{n}}\sqrt{\left\|C\right\|_{HS}+\frac{1}{m}} \tag{15}$$

or

$$A=\frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{\sqrt{n}}\left(\left\|C\right\|_{HS}^{2}+\frac{3}{m}\right)^{1/4}. \tag{16}$$

(ii) With probability greater than $1-\delta$ it holds for all $\mathbf{f}=(v_1,...,v_m)\circ T\in\mathcal{F}_B\circ\mathcal{A}$ and for all $\gamma\in(0,1)$ that

$$er\left(\mathbf{h}_{\mathbf{f}}\right)\leq e\hat{r}_{\gamma}\left(\mathbf{f}\right)+\frac{1}{\gamma(1-\varepsilon)}A\left(\mathbf{X}\right)+\sqrt{\frac{9\ln\left(2/\left(\delta\gamma\varepsilon\right)\right)}{2nm}},$$

where the random variable $A(\mathbf{X})$ is either

$$A\left(\mathbf{X}\right)=\frac{2B\left\|\mathcal{A}\right\|_{HS}}{\sqrt{n}}\sqrt{\frac{1}{mn}\left\|\hat{C}\left(\mathbf{x}\right)\right\|_{Fr}+\frac{1}{m}}$$

or

$$A\left(\mathbf{X}\right)=\frac{2B\left\|\mathcal{A}^2\right\|_{HS}^{1/2}}{\sqrt{n}}\left(\left(\frac{1}{mn}\left\|\hat{C}\left(\mathbf{x}\right)\right\|_{Fr}\right)^{2}+\frac{2}{m}\right)^{1/4}.$$

We finally extend this result from uniformly bounded sets $\mathcal{A}$ of operators to the set HS^* of all symmetric Hilbert-Schmidt operators. This is done following the techniques described in (Anthony, Bartlett, 1999), using the following lemma (a copy of Lemma 15.5 from this reference):

Lemma 14 *Suppose*

$$\{F(\alpha_1,\alpha_2,\delta) : 0 < \alpha_1,\alpha_2,\delta \leq 1\}$$

is a set of events such that:

(i) For all $0 < \alpha \leq 1$ *and* $0 < \delta \leq 1$,

$$\Pr\{F(\alpha,\alpha,\delta)\} \leq \delta.$$

(ii) For all $0 < \alpha_1 \leq \alpha \leq \alpha_2 \leq 1$ *and* $0 < \delta_1 \leq \delta \leq 1$,

$$F(\alpha_1,\alpha_2,\delta_1) \subseteq F(\alpha,\alpha,\delta).$$

Then for $0 < a,\delta < 1$,

$$\Pr\left(\bigcup_{\alpha\in(0,1]} F(\alpha a,\alpha,\delta\alpha(1-a))\right) \leq \delta.$$

Applications of this lemma follow a standard pattern, as explained in detail in (Anthony, Bartlett, 1999). Let ε,δ,B be as in the previous theorem. For $\alpha \in (0,1]$ set

$$\mathcal{A}(\alpha) = \{T \in HS^* : \|T\|_{HS} \leq 1/\alpha\}$$

and consider the events

$$F(\alpha_1,\alpha_2,\delta) = \{\exists \mathbf{f} \in \mathcal{F}_B \circ \mathcal{A}(\alpha_2) \text{ such that}$$
$$\operatorname{er}(\mathbf{h_f}) > \hat{\operatorname{er}}_\gamma(\mathbf{f}) + \frac{2B}{\alpha_1\gamma(1-\varepsilon)\sqrt{n}}\sqrt{\|C\|_{HS}+\frac{1}{m}} + \sqrt{\frac{\ln(1/(\delta\gamma\varepsilon))}{2nm}}\Bigg\}.$$

By the first conclusion of Theorem 13 the events $F(\alpha_1,\alpha_2,\delta)$ satisfy hypothesis (i) of Lemma 14, and it is easy to see that (ii) also holds. If we set $a = 1-\varepsilon$ and replace α by $1/\|T\|_{HS}$, then the conclusion of Lemma 14 reads as follows:

With probability greater $1-\delta$ it holds for every $\mathbf{f} = (v_1,...,v_m)\circ T$ with $(v_1,...,v_m) \in \mathcal{F}_B$ and $T \in HS^*$ with $\|T\|_{HS} \geq 1$ and all $\gamma \in (0,1)$ that

$$\operatorname{er}(\mathbf{h_f}) \leq \hat{\operatorname{er}}_\gamma(\mathbf{f}) + \frac{2B\|T\|_{HS}}{\gamma(1-\varepsilon)^2\sqrt{n}}\sqrt{\|C\|_{HS}+\frac{1}{m}} + \sqrt{\frac{\ln\left(\frac{\|T\|_{HS}}{\delta\gamma\varepsilon^2}\right)}{2nm}}.$$

Applying the same technique to the other conclusions of Theorem 13 gives the following result, which we state in abbreviated fashion:

Theorem 15 *Theorem 13 holds with the following modifications:*

- *The class* $\mathcal{F}_B \circ \mathcal{A}$ *is replaced by all* $\mathbf{f} = (v_1,...,v_m)\circ T \in \mathcal{F}_B \circ HS^*$ *with* $\|T\|_{HS} \geq 1$ *(or* $\left\|T^2\right\|_{HS} \geq 1$*).*
- $\|\mathcal{A}\|_{HS}$ *(or* $\left\|\mathcal{A}^2\right\|_{HS}$*) is replaced by* $\|T\|_{HS}$ *(or* $\left\|T^2\right\|_{HS}$*).*
- $(1-\varepsilon)$ *and* $1/(\delta\gamma\varepsilon)$ *are replaced by* $(1-\varepsilon)^2$ *and* $\|T\|_{HS}/\left(\delta\gamma\varepsilon^2\right)$ *(or* $\left\|T^2\right\|_{HS}^{1/2}/\left(\delta\gamma\varepsilon^2\right)$*) respectively.*

The requirement $\|T\|_{HS} \geq 1$ (or $\left\|T^2\right\|_{HS} \geq 1$) is an artifact introduced by the stratification lemma 14. Setting $\varepsilon = 1/2$ and $B = 1$ gives Theorem 1 and Theorem 2.

7. An Example

We conclude with an example illustrating the behavior of our bounds when learning from noisy data. We start with a fairly generic situation and subsequently introduce several idealized assumptions.

Suppose that $\left(X^l, Y^l\right)_{l=1}^m$ are random variables modelling a multi-task problem as above. In the following let M be the smallest closed subspace $M \subset H$ such that $X^l \in M$ a.s..

We now mix the data variables X^l with noise, modeled by a random variable X^N with values in H, such that $\left\|X^N\right\| \leq 1$ and $E\left[X^N\right] = 0$. The mixture is controlled by a parameter $s \in (0,1]$ and replaces the original data-variables X^l with the contaminated random variable $\hat{X}^l = sX^l + (1-s)X^N$. We now make the assumption that

- X^N is independent of X^l and Y^l for all l.

Let us call s the signal amplitude and $1-s$ the noise amplitude. The case $s = 1$ corresponds to the original multi-task problem. Decreasing s, and adding more noise, clearly makes learning more difficult up to the case $s = 0$ (which we exclude), where the data variables become independent of the labels and learning becomes impossible. We will look at the behavior of both of our bounds for non-interacting (single-task) and interacting (multi-task) learners as we decrease the signal amplitude s. The bounds which we use are implied by Lemma 11 and Theorem 9 for the non-interacting case and Theorem 13 for the interacting case, and state that each of the following two statements holds with probability at least $1-\delta$:

1. **Non-interacting bound**. $\forall \mathbf{v} \in \mathcal{F}_1$, $\forall \gamma$,
   $\mathrm{er}(\mathbf{h}_\mathbf{v}) \leq \hat{\mathrm{er}}_\gamma(\mathbf{v}) + \frac{2}{\gamma(1-\varepsilon)\sqrt{n}} + \sqrt{\frac{\ln(1/(\delta\gamma\varepsilon))}{2nm}}$
2. **Interacting bound**. $\forall \mathbf{v} \circ T \in \mathcal{F}_1 \circ \mathcal{A}$, $\forall \gamma$,
   $\mathrm{er}(\mathbf{h}_\mathbf{v} \circ T) \leq \hat{\mathrm{er}}_\gamma(\mathbf{v} \circ T) + \frac{2\|\mathcal{A}\|_{HS}}{\gamma(1-\varepsilon)\sqrt{n}} \sqrt{\|C\|_{HS} + \frac{1}{m}} + \sqrt{\frac{\ln(1/(\delta\gamma\varepsilon))}{2nm}}$

The first damage done by decreasing s is that the margin γ must be also decreased to $s\gamma$ to obtain a comparable empirical margin error for the mixed problem as for the original problem. Replacing γ by γs is very crude and normally insufficient, because of interfering noise, but the replacement can be justified if one is willing to accept the orthogonality assumption:

- $X^N \perp M$ a.s.

The assumption that the noise to be mixed in is orthogonal to the signal is somewhat artificial. We will later assume that the dimension d of the signal space M is small and that X^N is distributed homogeneously on a high-dimensional sphere. This implies weak orthogonality in the sense that $\left\langle X^l, X^N \right\rangle^2$ is small with high probability, a statement which could also be used, but at the expense of considerable complications. To immediately free us from consideration of the empirical term (and only for this purpose), we make the orthogonality assumption. By projecting to M we can then find for any sample $\left(X_i^l, Y_i^l\right)$ and any preprocessor T and any multi-classifying vector $\mathbf{v}$ some T' and $\mathbf{v}'$ such that $\hat{\mathrm{er}}_{\gamma s}(\mathbf{v}')$ and $\hat{\mathrm{er}}_{\gamma s}(\mathbf{v}' \circ T')$ for the mixed sample $\left(\hat{X}_i^l, Y_i^l\right)$ are the same as $\hat{\mathrm{er}}_\gamma(\mathbf{v})$ and $\hat{\mathrm{er}}_\gamma(\mathbf{v} \circ T)$ for the original sample. We can therefore regard the empirical terms as equal in both bounds and for all values of s as long as $\mathcal{A}$ is stable under projection to M and γ is replaced by γs.

This will cause a logarithmic penalty in the last term depending on the confidence parameter δ, but we will neglect this entire term on the grounds of rapid decay with the product nm. The remaining term, which depends on s and is different for both bounds, is then

$$\frac{2}{\gamma s(1-\varepsilon)\sqrt{n}} \tag{17}$$

for the non-interacting and

$$\frac{2\|\mathcal{A}\|_{HS}}{\gamma s(1-\varepsilon)\sqrt{n}}\sqrt{\|C_s\|_{HS}+\frac{1}{m}} \tag{18}$$

for the interacting case. Here $C_s = (1/m)\sum_l E\left[Q_{sX^l+(1-s)X^N}\right]$ is the total covariance operator for the noisy mixture problem. By independence of X^N and X^l and the mean-zero assumption for X^N we obtain from Lemma 4 and 7 that

$$C_s = (1/m)\sum_l \left(s^2 E\left[Q_{X^l}\right] + (1-s)^2 E\left[Q_{X^N}\right]\right) = s^2 C + (1-s)^2 E\left[Q_{X^N}\right],$$

where C would be the total covariance operator for the original problem. To bound the HS-norm of this operator we now introduce a simplifying assumption of homogeneity for the noise distribution:

- X^N is distributed uniformly on a k-dimensional unit-sphere centered at the origin.

This implies that $\|E\left[Q_{X^N}\right]\|_{HS} = 1/\sqrt{k}$ so that

$$\|C_s\|_{HS}^2 \leq s^2\|C\|_{HS} + \|E\left[Q_{X^N}\right]\|_{HS} \leq s^2\|C\|_{HS} + 1/\sqrt{k},$$

and substitution in (18) gives the new term

$$\frac{2\|\mathcal{A}\|_{HS}}{\gamma(1-\varepsilon)\sqrt{n}}\sqrt{\|C\|_{HS}+\frac{1}{s^2}\left(\frac{1}{\sqrt{k}}+\frac{1}{m}\right)} \tag{19}$$

for the interacting bound. The dependence on the inverse signal amplitude in the first factor has disappeared, and as k and m increase, the bound for the noisy problem tends to the same limiting value

$$\frac{2\|\mathcal{A}\|_{HS}\|C\|_{HS}^{1/2}}{\gamma(1-\varepsilon)\sqrt{n}}$$

as the bound for the original 'noise free' problem, for any fixed positive value of s. This contrasts the behavior of all bounds which depend linearly on the dimension of the input space (such as in) and diverge as $k \to \infty$.

The quotient of (19) to the non-interacting (17) is

$$\|\mathcal{A}\|_{HS}\sqrt{s^2\|C\|_{HS}+\frac{1}{\sqrt{k}}+\frac{1}{m}},$$

and the interacting bound will be better than the non-interacting bound whenever this expression is less than unity. This is more likely to happen when the signal amplitude s is small, and the dimension k of the noise distribution and the number of tasks m are large.

An intuitive explanation of the fact, that for multi-task learning a large dimension k of the noise-distribution has a positive effect on the bound, is that for large k a sample of homogeneously distributed random unit vectors is less likely to lie in a common low-dimensional subspace, a circumstance which could mislead the multi-task learner.

Of course there are many situations, when multi-task learning doesn't give any advantage over single-task learning. To make a quantitative comparison we make more three more simplifying assumptions on the data distribution of the X^l:

- $\dim(M) = d < \infty$

The signal space M is of course unknown to the learner, but we assume that

- we know its dimension d.

Multi-task learning can then select from an economically chosen set $\mathcal{A} \subset HS^*$ of preprocessors such that $\mathcal{A}$ contains the set of d-dimensional projections $\mathcal{P}_d$ and $\|\mathcal{A}\|_{HS} = \sqrt{d}$. We assume knowledge of d mainly for simplicity, without it we could invoke Theorem 15 instead of Theorem 13 above, causing some complications, which we seek to avoid.

- The mixture of the distributions of the X^l is homogeneous on $S_1 \cap M$.

This implies $\|C\|_{HS} = 1/\sqrt{d}$, and, with $\|\mathcal{A}\|_{HS} = \sqrt{d}$, the multi-task bound will improve over the non-interacting bound if

$$\sqrt{d}s^2 + \frac{d}{\sqrt{k}} + \frac{d}{m} < 1.$$

From this condition we conclude with four cook-book-rules to decide when it is worthwhile to go through the computational trouble of multi-task learning instead of the simpler single-task learning.

1. The problem is very noisy (s is expected to be small)
2. The noise is high-dimensional (k is expected to be large)
3. There are many learning tasks (m is large)
4. We suspect that the relevant information for all m tasks lies in a low-dimensional (d is small)

If one believes these criteria to be met, then one can use an algorithm as the one developed in (Ando, Zhang, 2005) to minimize the interacting bound above, with $\mathcal{A} = \mathcal{P}_d$.

Appendix

We give a proof of Theorem 9 for the readers convenience. Most of this material is combined from Anthony and Bartlett (1999), Bartlett and Mendelson (2002), Bartlett et al (2005) and Ando and Zhang (2005), and we make no claim to originality for any of it. A preliminary result is

Theorem 16 *Let $\mathcal{F}$ be a $[0,1]^m$-valued function class on a space $\mathcal{X}$, and $\mathbf{X} = \left(X_i^l\right)_{(l,i)=(1,1)}^{(m,n)}$ a vector of $\mathcal{X}$-valued independent random variables where for fixed l and varying i all the X_i^l are identically distributed. Fix $\delta > 0$. Then with probability greater than $1-\delta$ we have for all $\mathbf{f} = \left(f^1, ..., f^m\right) \in \mathcal{F}$*

$$\frac{1}{m}\sum_{l=1}^{m} E\left[f^l\left(X_1^l\right)\right] \leq \frac{1}{mn}\sum_{l=1}^{m}\sum_{i=1}^{n} f^l\left(X_i^l\right) + \mathcal{R}_n^m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2mn}}.$$

We also have with probability greater than $1-\delta$ for all $\mathbf{f} = \left(f^1, ..., f^m\right) \in \mathcal{F}$, that

$$\frac{1}{m}\sum_{l=1}^{m} E\left[f^l\left(X_1^l\right)\right] \leq \frac{1}{mn}\sum_{l=1}^{m}\sum_{i=1}^{n} f^l\left(X_i^l\right) + \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X}) + \sqrt{\frac{9\ln(2/\delta)}{2mn}}.$$

Proof Let Ψ be the function on $\mathcal{X}^{mn}$ given by

$$\Psi(\mathbf{x}) = \sup_{\mathbf{f}\in\mathcal{F}} \frac{1}{m}\sum_{l=1}^{m}\left(E\left[f^l\left(X_1^l\right)\right] - \frac{1}{n}\sum_{i=1}^{n} f^l\left(X_i^l\right)\right)$$

and let $\mathbf{X}'$ be an iid copy of the $\mathcal{X}^{mn}$-valued random variable $\mathbf{X}$. Then

$$\begin{aligned} E[\Psi(\mathbf{X})] &= E_{\mathbf{X}}\left[\sup_{\mathbf{f}\in\mathcal{F}} \frac{1}{mn} E_{\mathbf{X}'}\left[\sum_{l=1}^{m}\sum_{i=1}^{n}\left(f^l\left(\left(X_i^l\right)'\right) - f^l\left(X_i^l\right)\right)\right]\right] \\ &\leq E_{\mathbf{X}\mathbf{X}'}\left[\sup_{\mathbf{f}\in\mathcal{F}} \frac{1}{mn}\sum_{l=1}^{m}\sum_{i=1}^{n}\left(f^l\left(\left(X_i^l\right)'\right) - f^l\left(X_i^l\right)\right)\right] \\ &= E_{\mathbf{X}\mathbf{X}'}\left[\sup_{\mathbf{f}\in\mathcal{F}} \frac{1}{mn}\sum_{l=1}^{m}\sum_{i=1}^{n}\sigma_i^l\left(f^l\left(\left(X_i^l\right)'\right) - f^l\left(X_i^l\right)\right)\right], \end{aligned}$$

for any realization $\sigma = \left(\sigma_i^l\right)$ of the Rademacher variables, because the expectation $E_{\mathbf{X}\mathbf{X}'}$ is symmetric under the exchange $\left(X_i^l\right)' \longleftrightarrow X_i^l$. Hence

$$E[\Psi(\mathbf{X})] \leq E_{\mathbf{X}} E_{\sigma}\left[\sup_{\mathbf{f}\in\mathcal{F}} \frac{2}{mn}\sum_{l=1}^{m}\sum_{i=1}^{n}\sigma_i^l f^l\left(X_i^l\right)\right] = \mathcal{R}_n^m(\mathcal{F}).$$

Now fix $\mathbf{x} \in \mathcal{X}^{mn}$ and let $\mathbf{x}' \in \mathcal{X}^{mn}$ be as $\mathbf{x}$, except for one modified coordinate $\left(x_i^l\right)'$. Since each f^l has values in $[0,1]$ we have $|\Psi(\mathbf{x}) - \Psi(\mathbf{x}')| \leq 1/(mn)$. So by the one-sided version of the bounded difference inequality (see McDiarmid, 1998)

$$\Pr\left\{\Psi(\mathbf{X}) > E_{\mathbf{X}'}\left[\Psi\left(\mathbf{X}'\right)\right] + \sqrt{\frac{\ln(1/\delta)}{2mn}}\right\} \leq \delta.$$

Together with the above bound on $E[\Psi(\mathbf{X})]$ and the definition of Ψ this gives the first conclusion.

With $\mathbf{x}$ and $\mathbf{x}'$ as above we have $\left|\hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{x}) - \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{x}')\right| \leq 2/(mn)$, so by the other tail of the bounded difference inequality

$$\Pr\left\{\mathcal{R}_n^m(\mathcal{F}) < \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X}) + \sqrt{\frac{4\ln(1/\delta)}{2mn}}\right\} \leq \delta,$$

which, combined with the first conclusion in a union bound, gives the second conclusion. ■

We quote the following folklore theorem (see for example Bartlett et al, 2005) bounding the Rademacher complexity of a function class composed with a fixed Lipschitz function.

Theorem 17 *Let $\mathcal{F}$ be an $\mathbb{R}^m$-valued function class on a space $\mathcal{X}$ and suppose that $\phi : \mathbb{R} \to \mathbb{R}$ has Lipschitz constant L. Let*

$$\phi \circ \mathcal{F} = \left\{ \left(\phi \circ f^1, ..., \phi \circ f^m\right) : \left(f^1, ..., f^m\right) \in \mathcal{F} \right\}.$$

Then

$$\hat{\mathcal{R}}_n^m(\phi \circ \mathcal{F}) \leq L\, \hat{\mathcal{R}}_n^m(\mathcal{F}).$$

Suppose now that $\mathcal{F}$ is an $\mathbb{R}^m$-valued function class on $\mathcal{X}$. For $\mathbf{f} = \left(f^1, ..., f^m\right)$ define functions $\mathbf{f}' = \left(f'^1, ..., f'^m\right)$ and $\mathbf{f}'' = \left(f''^1, ..., f''^m\right)$, from $\mathcal{X} \times \{-1,1\}$ to $\mathbb{R}^m$ or $[0,1]^m$ respectively, by

$$f'^l(x,y) = y f^l(x) \text{ and } f''^l(x,y) = \phi_\gamma \circ f'^l(x,y) = \phi_\gamma(y f(x))$$

and function classes $\mathcal{F}' = \{\mathbf{f}' : \mathbf{f} \in \mathcal{F}\}$ and $\mathcal{F}'' = \{\mathbf{f}'' : \mathbf{f} \in \mathcal{F}\}$. It follows from the definition of $\hat{\mathcal{R}}$ that $\hat{\mathcal{R}}_n^m(\mathcal{F}')(\mathbf{x},\mathbf{y}) = \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{x})$ for all $(\mathbf{x},\mathbf{y}) \in (\mathcal{X} \times \{-1,1\})^{nm}$. Since ϕ_γ is Lipschitz with constant γ^{-1}, the previous theorem implies that

$$\hat{\mathcal{R}}_n^m\left(\mathcal{F}''\right)(\mathbf{X},\mathbf{Y}) \leq \gamma^{-1} \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X}) \text{ and } \mathcal{R}_n^m\left(\mathcal{F}''\right) \leq \gamma^{-1} \mathcal{R}_n^m(\mathcal{F}). \tag{20}$$

On the other hand, for every $\mathbf{f} = \left(f^1, ..., f^m\right) \in \mathcal{F}$ we have

$$\begin{aligned} \mathrm{er}(\mathbf{h_f}) &= \frac{1}{m} \sum E\left[1_{(-\infty,0]}\left(Y_1^l f^l\left(X_1^l\right)\right)\right] \\ &\leq \frac{1}{m} \sum E\left[\phi_\gamma \circ \left(f'\right)^l \left(X_1^l, Y_1^l\right)\right] \\ &= \frac{1}{m} \sum E\left[\left(f''\right)^l \left(X_1^l, Y_1^l\right)\right] \end{aligned} \tag{21}$$

and

$$\frac{1}{mn} \sum_{l=1}^{m} \sum_{i=1}^{n} f''^l\left(X_i^l, Y_i^l\right) = \frac{1}{mn} \sum_{l=1}^{m} \sum_{i=1}^{n} \phi_\gamma\left(Y_i^l f^l\left(X_i^l\right)\right) = \hat{\mathrm{er}}_\gamma(\mathbf{f}). \tag{22}$$

Applying Theorem 16 to the class $\mathcal{F}''$ and substitution of (21), (22) and (20) yield

Theorem 18 *Let $\mathcal{F}$ be a $\mathbb{R}^m$-valued function class on a space $\mathcal{X}$, $\gamma \in (0,1)$ and*

$$(\mathbf{X},\mathbf{Y}) = \left(X_i^l, Y_i^l\right)_{(l,i)=(1,1)}^{(m,n)}$$

a vector of $\mathcal{X} \times \{-1,1\}$-valued independent random variables where for fixed l and varying i all the $\left(X_i^l, Y_i^l\right)$ are identically distributed. Fix $\delta > 0$. Then with probability greater than $1-\delta$ we have for all $\mathbf{f} \in \mathcal{F}$

$$er(\mathbf{h_f}) \leq \hat{er}_\gamma(\mathbf{f}) + \gamma^{-1} \mathcal{R}_n^m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2mn}}.$$

We also have with probability greater than $1-\delta$ for all $\mathbf{f} \in \mathcal{F}$, that

$$er(\mathbf{h_f}) \leq \hat{er}_\gamma(\mathbf{f}) + \gamma^{-1} \hat{\mathcal{R}}_n^m(\mathcal{F})(\mathbf{X}) + \sqrt{\frac{9\ln(2/\delta)}{2mn}}.$$

To arrive at Theorem 9 we still need to convert this into a statement valid with high probability for all margins $\gamma \in (0,1)$. This is done with Lemma 14, which we now apply to the event

$$F(\alpha_1,\alpha_2,\delta) = \left\{ \exists \mathbf{f} \in \mathcal{F} \text{ s.t. } \mathrm{er}(\mathbf{h_f}) > \hat{\mathrm{er}}_{\alpha_2}(\mathbf{f}) + \alpha_1^{-1}\mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2mn}} \right\}.$$

Hypothesis (i) of Lemma 14 follows from the previous theorem, hypothesis (ii) from the fact that the right side in the inequality increases if we decrease δ and α_1 and increase α_2. If we replace a by $1-\varepsilon$ and α by γ, then the conclusion of the lemma becomes the first conclusion of Theorem 9. The second conclusion of Theorem 9 is handled similarly.

The following table is intended as an index and a quick reference to the notation and definitions introduced in the paper.

Notation	Short Description	Section
H	real, separable Hilbert space	2
$\langle .,. \rangle$ and $\|.\|$	inner product and norm on H	2
S_1	unit-sphere in H	3
HS	Hilbert-Schmidt operators on H	2
$\langle .,. \rangle_{HS}$ and $\|.\|_{HS}$	inner product and norm on HS	2
HS^*	symmetric operators in HS	2
$\mathcal{P}_d$	d-dimensional orthogonal projections in H	2
$\mathcal{A}$	a subset of HS^*	2
$\|\mathcal{A}\|_{HS}$	$\sup_{T\in\mathcal{A}} \|T\|_{HS}$	2
$\left\|\mathcal{A}^2\right\|_{HS}$	$\sup_{T\in\mathcal{A}} \left\|T^2\right\|_{HS}$	2
Q_x, for $x \in H$	operator $Q_x z = \langle z,x\rangle x$, $\forall z \in H$	2
$G_{x,y}$, for $x,y \in H$	operator $G_{x,y} z = \langle x,z\rangle y$, $\forall z \in H$	2
$tr(T)$	trace of the operator T	2
$E[Q_X]$	covariance operator of H-valued r.v. X	3
$\mathcal{X}$	generic input space	4
(X^l, Y^l)	random variables for multi-task problem	4
(X_i^l, Y_i^l)	random variables for multi-task sample	4
$\mathrm{er}(\mathbf{h})$	average error of multiclassifier $\mathbf{h}$	4
$\mathbf{h_f}$	multiclassifier obtained by thresholding $\mathbf{f}$	4
ϕ_γ	margin function	4
$\hat{\mathrm{er}}_\gamma(\mathbf{f})$	empirical margin error of vector function $\mathbf{f}$	4
$\hat{\mathcal{R}}_m(\mathcal{F})$	empirical Rademacher complexity	4
C^l	covariance operator for l-th task	5
C	total covariance operator	5
$\hat{C}(\mathbf{X})$	Gramian of data-sample $\mathbf{X}$	3
$\mathcal{F}_B$, $\mathcal{F}_B \circ \mathcal{A}$	fctn. classes for linear multi-task learning	5

References

[1] R. K. Ando, T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6: 1817-1853, 2005.

[2] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge, UK, 1999.

[3] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research*, 3: 463-482, 2002.

[4] P. Bartlett, O. Bousquet and S. Mendelson. Local Rademacher complexities. Available online: http://www.stat.berkeley.edu/˜bartlett/papers/bbm-lrc-02b.pdf.

[5] P. Baxendale. Gaussian measures on function spaces. *Amer. J. Math.*, 98:891-952, 1976.

[6] J. Baxter. Theoretical Models of Learning to Learn, in *Learning to Learn*, S.Thrun, L.Pratt Eds. Springer 1998.

[7] J. Baxter. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research* 12: 149-198, 2000.

[8] S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *COLT 03*, 2003.

[9] R. Caruana. Multitask Learning, in *Learning to Learn*, S.Thrun, L.Pratt Eds. Springer 1998.

[10] Nello Cristianini and John Shawe-Taylor. Support Vector Machines. *Cambridge University Press*, 2000.

[11] T. Evgeniou and M. Pontil. Regularized multi-task learning. *Proc. Conference on Knowledge Discovery and Data Mining*, 2004.

[12] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, Vol. 30, No 1, 1-50.

[13] Colin McDiarmid. Concentration, in *Probabilistic Methods of Algorithmic Discrete Mathematics*, p. 195-248. Springer, Berlin, 1998.

[14] C. A. Micchelli and M. Pontil. Kernels for multi-task learning. Available online, 2005.

[15] S.Mika, B.Schölkopf, A.Smola, K.-R.Müller, M.Scholz and G.Rätsch. Kernel PCA and Denoising in Feature Spaces. *Advances in Neural Information Processing Systems* 11, 1998.

[16] J. Shawe-Taylor, N. Cristianini. Estimating the moments of a random vector. *Proceedings of GRETSI 2003 Conference*, I: 47–52, 2003.

[17] Michael Reed and Barry Simon. *Functional Analysis*, part I of *Methods of Mathematical Physics, Academic Press*, 1980.

[18] S. Thrun. Lifelong Learning Algorithms, in *Learning to Learn*, S.Thrun, L.Pratt Eds. Springer 1998

Journal of Machine Learning Research 7 (2006) 141–166 Submitted 6/05; Revised 9/05 and 12/05; Published 1/06

Active Learning in Approximately Linear Regression Based on Conditional Expectation of Generalization Error

Masashi Sugiyama SUGI@CS.TITECH.AC.JP
Department of Computer Science
Tokyo Institute of Technology
2-12-1, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan

Editor: Greg Ridgeway

Abstract

The goal of active learning is to determine the locations of training input points so that the generalization error is minimized. We discuss the problem of active learning in linear regression scenarios. Traditional active learning methods using least-squares learning often assume that the model used for learning is correctly specified. In many practical situations, however, this assumption may not be fulfilled. Recently, active learning methods using "importance"-weighted least-squares learning have been proposed, which are shown to be robust against misspecification of models. In this paper, we propose a new active learning method also using the weighted least-squares learning, which we call *ALICE* (Active Learning using the Importance-weighted least-squares learning based on Conditional Expectation of the generalization error). An important difference from existing methods is that we predict the *conditional* expectation of the generalization error given training input points, while existing methods predict the *full* expectation of the generalization error. Due to this difference, the training input design can be fine-tuned depending on the realization of training input points. Theoretically, we prove that the proposed active learning criterion is a more accurate predictor of the *single-trial* generalization error than the existing criterion. Numerical studies with toy and benchmark data sets show that the proposed method compares favorably to existing methods.

Keywords: Active Learning, Conditional Expectation of Generalization Error, Misspecification of Models, Importance-Weighted Least-Squares Learning, Covariate Shift.

1. Introduction

In a standard setting of supervised learning, the training input points are provided from the environment (Vapnik, 1998). On the other hand, there are cases where the location of the training input points can be designed by users (Fedorov, 1972; Pukelsheim, 1993). In such situations, it is expected that the accuracy of learned results can be improved by appropriately choosing the location of the training input points, e.g., by densely allocating the training input points in the regions with high uncertainty. *Active learning* (MacKay, 1992; Cohn et al., 1996; Fukumizu, 2000)—also referred to as *experimental design* in statistics (Kiefer, 1959; Fedorov, 1972; Pukelsheim, 1993)—is the problem of optimizing location of training input points so that the generalization error is minimized.

The generalization error can be decomposed into the *bias* and *variance* terms. In active learning research, it is often assumed that the model used for learning is correctly specified (Fedorov, 1972; Cohn et al., 1996; Fukumizu, 2000), i.e., the learning target function can be expressed by the model. Then, under a mild condition, the ordinary least-squares (OLS) learning yields that the bias term vanishes and only the variance term remains. Based on this fact, a traditional active learning method

with OLS tries to determine the location of the training input points so that the variance term is minimized (Fedorov, 1972). In practice, however, the correctness of the model may not be fulfilled.

Active learning is a situation under the *covariate shift* (Shimodaira, 2000), where the training input distribution is different from the test input distribution. When the model used for learning is correctly specified, the covariate shift does not matter because OLS is still unbiased under a mild condition. However, OLS is no longer unbiased even asymptotically for misspecified models, and therefore we have to explicitly deal with the bias term if OLS is used.

Under the covariate shift, it is known that a form of weighted least-squares learning (WLS) is shown to be asymptotically unbiased even for misspecified models (Shimodaira, 2000; Wiens, 2000). The key idea of this WLS is the use of the ratio of density functions of test and training input points: the goodness-of-fit of the training input points is adjusted to that of the test input points by the density ratio, which is similar to *importance sampling*.

In this paper, we propose a variance-only active learning method using WLS, which can be regarded as an extension of the traditional variance-only active learning method using OLS. The proposed method can be theoretically justified for the approximately correct models, and thus is *robust* against the misspecification of models.

Conditional Expectation of Generalization Error: A variance-only active learning method using WLS has also been proposed by Wiens (2000), which can also be theoretically justified for approximately correct models. The important difference is how the generalization error is predicted: we predict the *conditional* expectation of the generalization error given training input points, while in Wiens (2000), the *full* expectation of the generalization error is predicted. In order to explain this difference in more detail, we first note that the generalization error of the WLS estimator depends on the training input density since WLS explicitly uses it. Therefore, when WLS is used in active learning, the generalization error is predicted as a function of the training input density, and the training input density is optimized so that the predicted generalization error is minimized.

The parameters in the model are learned using the training examples, which consist of training input points drawn from the user-designed distribution and corresponding noisy output values. This means that the generalization error is a random variable which depends on the location of the training input points and noise contained in the training output values. We ideally want to predict the *single-trial* generalization error, i.e., the generalization error for a single realization of the training examples at hand. From this viewpoint, we do not want to average out the random variables, but we want to plug the realization of the random variables into the generalization error and evaluate the realized value of the generalization error. However, we may not be able to avoid taking the expectation over the training output noise since the training output noise is inaccessible. In contrast, the location of the training input points are accessible by nature. Motivated by this fact, in this paper, we predict the generalization error *without* taking the expectation over the training input points. That is, we predict the *conditional* expectation of the generalization error given training input points. On the other hand, in Wiens (2000), the generalization error is predicted in terms of the expectation over *both* the training input points and the training output noise.

A possible advantage of the conditional-expectation approach is schematically illustrated in Figure 1. For illustration purposes, we consider the case of sampling only one training example. The solid curves in the left graph (Figure 1-(a)) depict $G_{p_a}(\varepsilon|x)$, the generalization error for a training input density p_a as a function of the training output noise ε given a training input point x. The three solid curves correspond to the cases where the realizations of the training input point x are a_1, a_2,

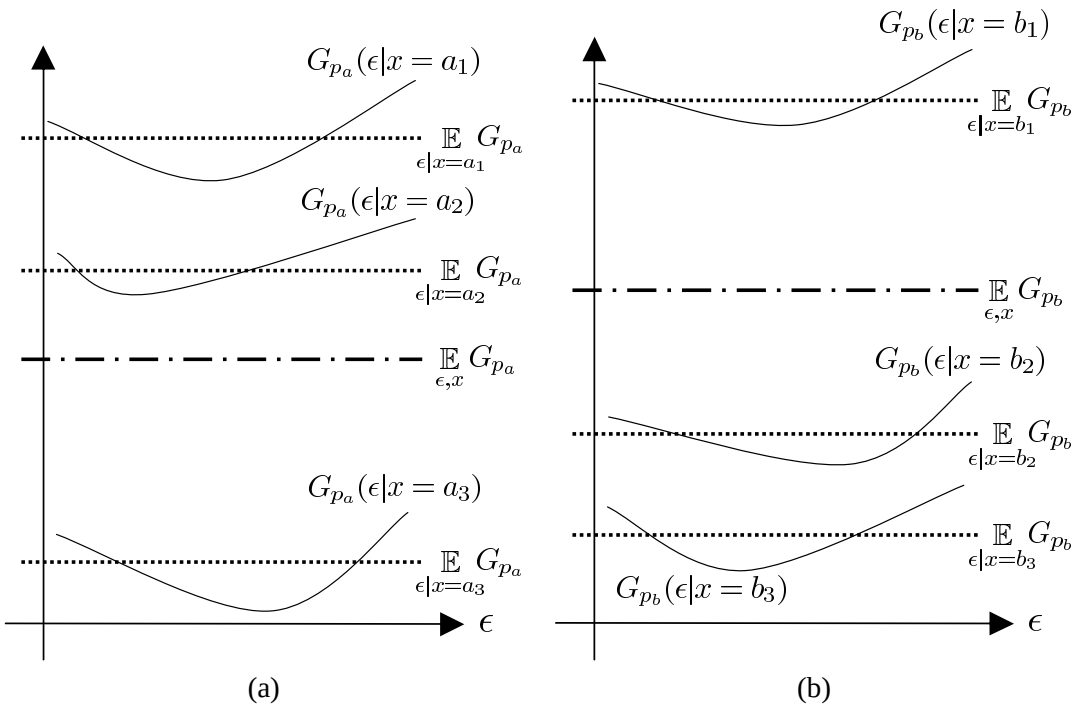


Figure 1: Schematic illustration of conditional expectation and full expectation of the generalization error. (a) and (b) correspond to the generalization error for p_a and p_b, respectively.

and a_3, respectively. The value of the generalization error for the density p_a in the full-expectation approach is depicted by the dash-dotted line, where the generalization error is expected over both the training output noise ε and the training input points x (i.e., the mean of the three solid curves). The values of the generalization error in the conditional-expectation approach are depicted by the dotted lines, where the generalization errors are expected only over the training output noise ε, given $x = a_1, a_2, a_3$, respectively (i.e., the mean of each solid curve). The right graph (Figure 1-(b)) depicts the generalization errors for the training input density p_b in the same manner.

In the full-expectation framework, the density p_a is judged to be better than p_b regardless of the realization of the training input point since the dash-dotted line in the left graph is lower than that in the right graph (see Figure 1 again). However, as the solid curves show, p_a is often worse than p_b in single trials. On the other hand, in the conditional-expectation framework, the goodness of the density is adaptively judged depending on the realizations of the training input point x. For example, p_b is judged to be better than p_a if a_2 and b_3 are realized, or p_a is judged to be better than p_b if a_3 and b_1 are realized. That is, the conditional-expectation framework may yield a better choice of the training input density (and the training input points) than the full-expectation framework.

The above discussion illustrates a conceptual advantage of the conditional-expectation approach. Theoretically, we prove that the proposed active learning criterion derived in the conditional-expectation framework is a better predictor of the single-trial generalization error than the full-expectation active learning criterion proposed by Wiens (2000). This substantiates the advantage of the conditional-expectation approach. Experimental results also support this claim: the

proposed method compares favorably to Wiens's method in the simulations with toy and benchmark data sets.

Bias-and-Variance Approach for Misspecified Models: Kanamori and Shimodaira (2003) also proposed an active learning algorithm using WLS. This method is not variance-only, but it takes both the bias and the variance into account by gathering training input points in two stages. In the first stage, a certain number of training examples are randomly gathered from the environment, and the generalization error (i.e., the sum of the bias and variance) is predicted by using the gathered training examples. Then in the second stage, the training input density for the remaining training examples is optimized based on the generalization error prediction. Theoretically, the two-stage method is shown to asymptotically give the optimal training input density not only for approximately correct models, but also for totally misspecified models. Although this property is solid, it may not be practically valuable since learning with totally misspecified models may not work well because of the model error. A drawback of this method is that it requires some randomly collected training examples in the first stage, so we are not allowed to optimally design all the training input locations by ourselves. Our experiments show that the proposed method works better than the two-stage method of Kanamori and Shimodaira (2003).

Batch Selection of Training Input Points: Active learning in the machine learning community is often thought of as being a *sequential* process: selecting one or a few training input points, observing corresponding training output values, training the model using the gathered training examples, and iterating this process. An alternative approach is the *batch* approach, where all training input points are gathered in the beginning.

If the environment is non-stationary, i.e., the learning target function drifts, taking the sequential approach would be necessary. On the other hand, under the stationary environment, i.e., the learning target function is fixed, the batch approach gives the globally optimal solution and the sequential approach can be regarded as a greedy approximation to it. In this paper, we consider the stationary case, so the batch approach is desirable.

In correctly specified linear regression, the expected generalization error does not depend on the learning target function under a mild condition. Therefore, the globally optimal solution can be obtained in principle. However, in misspecified linear regression which we discuss in this paper, the expected generalization error depends on the unknown learning target function. In this scenario, the sequential approach would be natural: estimating the unknown learning target function and optimizing location of the training input points are carried out alternately. On the other hand, in this paper, we do not estimate the learning target function, but we approximate the generalization error by the quantity which does *not* depend on the learning target function. This makes it possible to take the batch approach of determining all the training input points at once in advance.

A general criticism of the batch approach is that except for some special cases where the global optimal solution can be obtained analytically (Fedorov, 1972; Sugiyama and Ogawa, 2001), the batch approach usually requires the simultaneous optimization of all training input points, which is computationally very demanding. On the other hand, the sequential approach is computationally efficient since only one or a few training input points are optimized in each iteration (Cohn et al., 1996; Fukumizu, 2000; Sugiyama and Ogawa, 2000). In this paper, we avoid the computational difficulty of the batch approach not by resorting to the sequential approach, but by optimizing the training input distribution, rather than directly optimizing the training input points themselves. This

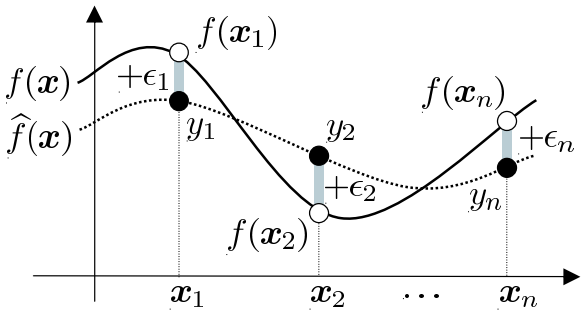


Figure 2: Regression problem.

seems to be a popular approach in batch active learning research (Wiens, 2000; Kanamori and Shimodaira, 2003).

Organization: The rest of this paper is organized as follows. We derive a new active learning method in Section 2, and we discuss relations between the proposed method and the existing methods in Section 3. We report numerical results using toy and benchmark data sets in Section 4. Finally, we state conclusions and future prospects in Section 5.

2. Derivation of New Active Learning Method

In this section, we formulate the active learning problem in regression scenarios, and derive a new active learning method.

2.1 Problem Formulation

Let us discuss the regression problem of learning a real-valued function $f(x)$ defined on $\mathbb{R}^d$ from training examples (see Figure 2). Training examples are given as

$$\{(x_i, y_i) \mid y_i = f(x_i) + \varepsilon_i\}_{i=1}^n,$$

where $\{\varepsilon_i\}_{i=1}^n$ are i.i.d. noise with mean zero and unknown variance σ^2. We suppose that the training input points $\{x_i\}_{i=1}^n$ are independently drawn from a user-defined distribution with density $p(x)$.

Let $\widehat{f}(x)$ be a learned function obtained from the training examples $\{(x_i, y_i)\}_{i=1}^n$. We evaluate the goodness of the learned function $\widehat{f}(x)$ by the expected squared test error over test input points, to which refer as the *generalization error*. When the test input points are drawn independently from a distribution with density $q(x)$, the generalization error G' is expressed as

$$G' = \int \left(\widehat{f}(x) - f(x)\right)^2 q(x) dx. \tag{1}$$

We suppose that $q(x)$ is known (or its reasonable estimate is available). This seems to be a common assumption in active learning literature (e.g., Fukumizu, 2000; Wiens, 2000; Kanamori and Shimodaira, 2003). If a large number of *unlabeled samples*[1] are easily gathered, a reasonably good

1. Unlabeled samples are input points without output values. We assume that unlabeled samples are independently drawn from the distribution with density $q(x)$.

estimate of $q(x)$ may be obtained by some standard density estimation method. Therefore, the assumption that $q(x)$ is known or its reasonable estimate is available may not be so restrictive.

In the following, we discuss the problem of optimizing the training input density $p(x)$ so that the generalization error is minimized.

2.2 Approximately Correct Linear Regression

We learn the target function $f(x)$ by the following linear regression model:

$$\widehat{f}(x) = \sum_{i=1}^{b} \widehat{\alpha}_i \varphi_i(x), \tag{2}$$

where $\{\varphi_i(x)\}_{i=1}^{b}$ are fixed linearly independent functions[2] and $\widehat{\alpha} = (\widehat{\alpha}_1, \widehat{\alpha}_2, \ldots, \widehat{\alpha}_b)^\top$ are parameters to be learned (by a variant of least-squares, see Section 2.4 for detail).

Suppose the regression model (2) does not exactly include the learning target function $f(x)$, but it *approximately* includes it, i.e., for a scalar δ such that $|\delta|$ is small, $f(x)$ is expressed as

$$f(x) = g(x) + \delta r(x), \tag{3}$$

where $g(x)$ is the optimal approximation to $f(x)$ by the model (2):

$$g(x) = \sum_{i=1}^{b} \alpha_i^* \varphi_i(x).$$

$\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_b^*)^\top$ is the unknown optimal parameter defined by

$$\alpha^* = \operatorname*{argmin}_{\alpha} \int \left(\sum_{i=1}^{b} \alpha_i \varphi_i(x) - f(x) \right)^2 q(x) dx.$$

$\delta r(x)$ in Eq.(3) is the residual, which is orthogonal to $\{\varphi_i(x)\}_{i=1}^{b}$ under $q(x)$ (see Figure 3):

$$\int r(x) \varphi_i(x) q(x) dx = 0 \quad \text{for } i = 1, 2, \ldots, b. \tag{4}$$

The function $r(x)$ governs the nature of the model error, and δ is the possible magnitude of this error. In order to separate these two factors, we further impose the following normalization condition on $r(x)$:

$$\int r^2(x) q(x) dx = 1. \tag{5}$$

Note that we are essentially estimating the projection $g(x)$, rather than the true target function $f(x)$.

2. Note that we do not impose any restrictions on the choice of basis functions. Therefore, Eq.(2) includes a variety of models such as polynomial models, trigonometric polynomial models, and Gaussian kernel models with fixed centers.

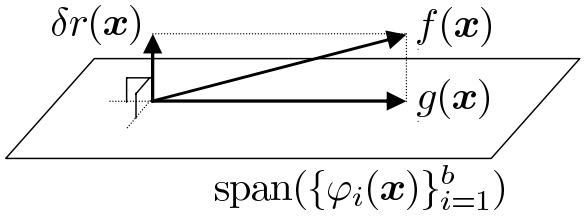


Figure 3: Orthogonal decomposition of $f(x)$.

2.3 Bias/Variance Decomposition of Generalization Error

As described in Section 1, we evaluate the generalization error in terms of the expectation over only the training output noise $\{\varepsilon_i\}_{i=1}^n$, not over the training input points $\{x_i\}_{i=1}^n$.

Let $\mathbb{E}_{\{\varepsilon_i\}}$ denote the expectation over the noise $\{\varepsilon_i\}_{i=1}^n$. Then, the generalization error expected over the training output noise can be decomposed into the (squared) *bias* term B, the *variance* term V, and the model error C:

$$\mathop{\mathbb{E}}_{\{\varepsilon_i\}} G' = B + V + C,$$

where

$$\begin{aligned} B &= \int \left(\mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{f}(x) - g(x) \right)^2 q(x) dx, \\ V &= \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \int \left(\widehat{f}(x) - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{f}(x) \right)^2 q(x) dx, \\ C &= \int (g(x) - f(x))^2 q(x) dx. \end{aligned} \tag{6}$$

Since C is constant which depends neither on $p(x)$ nor $\{x_i\}_{i=1}^n$, we subtract C from G' and define it by G.

$$G = G' - C.$$

2.4 Importance-Weighted Least-Squares Learning

Let X be the *design matrix*, i.e., X is the $n \times b$ matrix with the (i, j)-th element

$$X_{i,j} = \varphi_j(x_i).$$

A standard way to learn the parameters in the regression model (2) is the *ordinary least-squares (OLS) learning*, i.e., parameter vector α is determined as follows.

$$\widehat{\alpha}_O = \mathop{\mathrm{argmin}}_{\alpha} \left[\sum_{i=1}^n \left(\widehat{f}(x_i) - y_i \right)^2 \right], \tag{7}$$

where the subscript 'O' indicates the ordinary LS. $\widehat{\alpha}_O$ is analytically given by

$$\widehat{\alpha}_O = L_O y,$$

where

$$L_O = (X^\top X)^{-1} X^\top,$$
$$y = (y_1, y_2, \ldots, y_n)^\top.$$

When the training input points $\{x_i\}_{i=1}^n$ are drawn from $q(x)$, OLS is asymptotically unbiased even for misspecified models. However, the current situation is under the *covariate shift* (Shimodaira, 2000), where the training input density $p(x)$ is generally different from the test input density $q(x)$. Under the covariate shift, OLS is no longer unbiased even asymptotically for misspecified models. On the other hand, it is known that the following *weighted least-squares (WLS) learning* is asymptotically unbiased (Shimodaira, 2000).

$$\widehat{\alpha}_W = \operatorname*{argmin}_{\alpha} \left[\sum_{i=1}^{n} \frac{q(x_i)}{p(x_i)} \left(\widehat{f}(x_i) - y_i \right)^2 \right], \tag{8}$$

where the subscript 'W' indicates the weighted LS. Asymptotic unbiasedness of $\widehat{\alpha}_W$ would be intuitively understood by the following identity, which resembles the *importance sampling*:

$$\int \left(\widehat{f}(x) - f(x) \right)^2 q(x) dx = \int \left(\widehat{f}(x) - f(x) \right)^2 \frac{q(x)}{p(x)} p(x) dx.$$

In the following, we assume that $p(x)$ and $q(x)$ are strictly positive for all x.

Let D be the diagonal matrix with the i-th diagonal element

$$D_{i,i} = \frac{q(x_i)}{p(x_i)}.$$

Then $\widehat{\alpha}_W$ is analytically given by

$$\widehat{\alpha}_W = L_W y, \tag{9}$$

where

$$L_W = (X^\top D X)^{-1} X^\top D.$$

2.5 Active Learning Based on Importance-Weighted Least-Squares Learning

Let G_W, B_W and V_W be G, B and V for the learned function obtained by WLS, respectively. Let U be the b-dimensional square matrix with the (i, j)-th element

$$U_{i,j} = \int \varphi_i(x) \varphi_j(x) q(x) dx.$$

Then we have the following lemma (Proofs of all lemmas are provided in appendices).

Lemma 1 *For the approximately correct model (3), we have*

$$B_W = O_p(\delta^2 n^{-1}),$$
$$V_W = \sigma^2 \mathrm{tr}(U L_W L_W^\top) = O_p(n^{-1}). \tag{10}$$

Input: A finite set $\widehat{\mathcal{P}}$ of strictly positive probability densities

Calculate U.
For each $p \in \widehat{\mathcal{P}}$
    Create training input points $\{x_i^{(p)}\}_{i=1}^n$ following $p(x)$.
    Calculate L_W.
    Calculate $J(p)$.
End
Choose $\widehat{p}$ that minimizes J.
Put $x_i = x_i^{(\widehat{p})}$ for $i = 1, 2, \ldots, n$.
Observe the training output values $\{y_i\}_{i=1}^n$ at $\{x_i\}_{i=1}^n$.
Calculate $\widehat{\alpha}_W$ by Eq.(9).

Output: $\widehat{\alpha}_W$

Figure 4: Proposed ALICE algorithm.

Note that the asymptotic order in the above lemma is in probability since random variables $\{x_i\}_{i=1}^n$ are included. This lemma implies that if $\delta = o_p(1)$,

$$\mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W = \sigma^2 \mathrm{tr}(UL_W L_W^\top) + o_p(n^{-1}). \tag{11}$$

Motivated by Eq.(11), we propose determining the training input density $p(x)$ as follows: For a set $\mathcal{P}$ of strictly positive probability densities,

$$p^* = \mathop{\mathrm{argmin}}_{p \in \mathcal{P}} J(p),$$

where

$$J = \mathrm{tr}(UL_W L_W^\top). \tag{12}$$

Practically, we may prepare a finite set $\widehat{\mathcal{P}}$ of strictly positive probability densities and choose the one that minimizes J from the set $\widehat{\mathcal{P}}$. A pseudo code of the proposed active learning algorithm is described in Figure 4, which we call *ALICE* (Active Learning using the Importance-weighted least-squares learning based on Conditional Expectation of the generalization error). Note that the value of J depends not only on $p(x)$, but also on the realization of the training input points $\{x_i^{(p)}\}_{i=1}^n$.

3. Relation to Existing Methods

In this section, we qualitatively compare the proposed active learning method with existing methods.

3.1 Active Learning with OLS

Let G_O, B_O and V_O be G, B and V for the learned function obtained by OLS, respectively. If $\delta = 0$ in Eq.(3), i.e., the model is correctly specified, B_O vanishes under a mild condition (Fedorov, 1972) and we have

$$\mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_O = V_O = \sigma^2 \mathrm{tr}(UL_O L_O^\top).$$

Based on the above expression, the training input density $p(x)$ is determined[3] as follows (Fedorov, 1972; Cohn et al., 1996; Fukumizu, 2000).

$$p_O^* = \operatorname*{argmin}_{p \in \mathcal{P}} J_O(p),$$

where

$$J_O = \mathrm{tr}(U L_O L_O^\top). \tag{13}$$

Comparison with J: We investigate the validity of J_O for approximately correct models based on the following lemma.

Lemma 2 *For the approximately correct model (3), we have*

$$B_O = O(\delta^2),$$
$$V_O = O_p(n^{-1}).$$

The above lemma implies that if $\delta = o_p(n^{-\frac{1}{2}})$,

$$\mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_O = \sigma^2 J_O + o_p(n^{-1}).$$

Therefore, if $\delta = o_p(n^{-\frac{1}{2}})$, the use of J_O can be still justified. On the other hand, the proposed J is valid when $\delta = o_p(1)$. This implies that J has a wider range of applications than J_O. As experimentally shown in Section 4, this difference is highly significant in practice.

3.2 Active Learning with WLS: Variance-Only Approach

For the importance-weighted least-squares learning (8), Kanamori and Shimodaira (2003) proved that the generalization error expected over training input points $\{x_i\}_{i=1}^n$ and training output noise $\{\varepsilon_i\}_{i=1}^n$ is asymptotically expressed as

$$\mathop{\mathbb{E}}_{\{x_i\}} \mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W = \frac{1}{n}\mathrm{tr}(U^{-1}H) + O(n^{-\frac{3}{2}}), \tag{14}$$

where $\mathbb{E}_{\{x_i\}}$ is the expectation over training input points $\{x_i\}_{i=1}^n$ and H is the b-dimensional square matrix defined by

$$H = S + \sigma^2 T.$$

S and T are the b-dimensional square matrices with the (i,j)-th elements

$$S_{i,j} = \int \varphi_i(x)\varphi_j(x)(\delta r(x))^2 \frac{q(x)^2}{p(x)} dx, \tag{15}$$

$$T_{i,j} = \int \varphi_i(x)\varphi_j(x) \frac{q(x)^2}{p(x)} dx. \tag{16}$$

3. $p(x)$ is not explicitly used in OLS. Therefore, we do not have to optimize the training input density $p(x)$, but we can directly optimize training input points $\{x_i\}_{i=1}^n$. However, to be consistent with the WLS-based methods, we optimize $p(x)$ in this paper. This also helps to avoid the simultaneous optimization of n input points which is computationally very demanding in general.

Note that $\frac{1}{n}\mathrm{tr}(U^{-1}S)$ corresponds to the squared bias while $\frac{\sigma^2}{n}\mathrm{tr}(U^{-1}T)$ corresponds to the variance. Eq.(14) suggests that $\mathrm{tr}(U^{-1}H)$ may be used as an active learning criterion. However, H includes the inaccessible quantities $\delta r(x)$ and σ^2, so $\mathrm{tr}(U^{-1}H)$ can not be directly calculated.

To cope with this problem, Wiens (2000) proposed[4] ignoring S (the bias term), which yields

$$\mathop{\mathbb{E}}_{\{x_i\}}\mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W \approx \frac{\sigma^2}{n}\mathrm{tr}(U^{-1}T).$$

Note that T is accessible under the current setting. Based on this approximation, the training input density $p(x)$ is determined as follows.

$$p_W^* = \mathop{\mathrm{argmin}}_{p\in\mathcal{P}} J_W(p),$$

where

$$J_W = \frac{1}{n}\mathrm{tr}(U^{-1}T). \tag{17}$$

Comparison with J: A notable feature of J_W is that the optimal training input density $p_W^*(x)$ can be obtained analytically (Wiens, 2000):

$$p_W^*(x) = \frac{\widehat{h}(x)}{\int \widehat{h}(x)dx}, \tag{18}$$

where

$$\widehat{h}(x) = q(x)\left(\sum_{i,j=1}^{b} U_{i,j}^{-1}\varphi_i(x)\varphi_j(x)\right)^{\frac{1}{2}}.$$

This may be confirmed by the fact that J_W can be expressed as

$$J_W(p) = \frac{1}{n}\left(\int \widehat{h}(x)dx\right)^2\left(1+\int\frac{(p_W^*(x)-p(x))^2}{p(x)}dx\right).$$

On the other hand, we do not yet have an analytic form of a minimizer for the criterion J.

It seems that in Wiens (2000), ignoring S has not been well justified. Here, we investigate the validity based on the following corollary immediately obtained from Eqs.(14) and (15).

Corollary 1 *For the approximately correct model (3), we have*

$$\mathop{\mathbb{E}}_{\{x_i\}}\mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W = \sigma^2 J_W + O(\delta^2 n^{-1} + n^{-\frac{3}{2}}),$$

where $\sigma^2 J_W = O(n^{-1})$.

4. In the original paper, discussion is restricted to the cases where the input domain is bounded and $q(x)$ is uniform over the domain. However, it may be easily extended to an arbitrary strictly-positive $q(x)$. For this reason, we deal with the extended version here.

This corollary implies that if $\delta = o(1)$,

$$\mathop{\mathbb{E}}_{\{x_i\}} \mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W = \sigma^2 J_W + o(n^{-1}),$$

by which the use of J_W can be justified asymptotically. Since the order is the same as that of the proposed criterion, J and J_W may be comparable in the robustness against the misspecification of models.

Now the following lemma reveals a more direct relation between J and J_W.

Lemma 3 *J and J_W satisfy*

$$J = J_W + O_p(n^{-\frac{3}{2}}). \tag{19}$$

This lemma implies that J is asymptotically equivalent to J_W. However, they are still different in the order of n^{-1}. In the following, we show that this difference is important.

In the active learning context, we are interested in accurately predicting the *single-trial* generalization error G_W, which depends on the realization of the training examples. Let us measure the goodness of a generalization error predictor $\widehat{G}$ by

$$\mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\widehat{G} - G_W)^2. \tag{20}$$

Then we have the following lemma.

Lemma 4 *Suppose $\delta = o_p(n^{-\frac{1}{4}})$. If terms of $o_p(n^{-3})$ are ignored, we have*

$$\mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J_W - G_W)^2 \geq \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J - G_W)^2.$$

This lemma states that under $\delta = o_p(n^{-\frac{1}{4}})$, $\sigma^2 J$ is asymptotically a more accurate estimator of the single-trial generalization error G_W than $\sigma^2 J_W$ in the sense of Eq.(20).

In Section 4, we experimentally evaluate the difference between J and J_W.

3.3 Active Learning with WLS: Bias-and-Variance Approach

Another idea of approximating H in Eq.(14) is a two-stage sampling scheme proposed[5] by Kanamori and Shimodaira (2003): the training examples sampled in the first stage are used for estimating H and in the second stage, the distribution of the remaining training input points is optimized based on the estimated H. We explain the details of the algorithm below.

First, ℓ ($\leq n$) training input points $\{\widetilde{x}_i\}_{i=1}^{\ell}$ are created independently following the test input distribution with density $q(x)$, and corresponding training output values $\{\widetilde{y}_i\}_{i=1}^{\ell}$ are observed. Let $\widetilde{D}$ and $\widetilde{Q}$ be the ℓ-dimensional diagonal matrices with the i-th diagonal elements

$$\widetilde{D}_{i,i} = \frac{q(\widetilde{x}_i)}{p(\widetilde{x}_i)},$$
$$\widetilde{Q}_{i,i} = [\widetilde{y} - \widetilde{X}(\widetilde{X}^\top \widetilde{X})^{-1} \widetilde{X}^\top \widetilde{y}]_i,$$

5. In the original paper, the method is derived within a slightly different setting of estimating the conditional probability of the output value y given an input point x for regular statistical models. Here, we focus on the cases where the conditional distirbution is Gaussian and the statistical model is linear, by which the setting becomes comparable to that of the current paper.

where $[\cdot]_i$ denotes the i-th element of a vector. $\widetilde{X}$ is the design matrix for $\{\widetilde{x}_i\}_{i=1}^{\ell}$, i.e., the $\ell \times b$ matrix with the (i,j)-th element

$$\widetilde{X}_{i,j} = \varphi_j(\widetilde{x}_i),$$

and

$$\widetilde{y} = (\widetilde{y}_1, \widetilde{y}_2, \ldots, \widetilde{y}_\ell)^\top.$$

Then an approximation $\widetilde{H}$ of the unknown matrix H in Eq.(14) is given by

$$\widetilde{H} = \frac{1}{\ell}\widetilde{X}^\top \widetilde{D}\widetilde{Q}^2\widetilde{X}.$$

Although U^{-1} is accessible in the current setting, Kanamori and Shimodaira (2003) also replaced it by a consistent estimate $\widetilde{U}^{-1}$, where

$$\widetilde{U} = \frac{1}{\ell}\widetilde{X}^\top \widetilde{X}.$$

Based on the above approximations, the training input density $p(x)$ is determined as follows:

$$p^*_{OW} = \operatorname*{argmin}_{p \in \mathcal{P}} J_{OW}(p),$$

where

$$J_{OW} = \frac{1}{n}\mathrm{tr}(\widetilde{U}^{-1}\widetilde{H}). \tag{21}$$

Note that the subscript '*OW*' indicates the combination of the ordinary LS and weighted LS (see below for details).

After determining the optimal density p^*_{OW}, the remaining $n-\ell$ training input points $\{x_i\}_{i=1}^{n-\ell}$ are created independently following $p^*_{OW}(x)$, and corresponding training output values $\{y_i\}_{i=1}^{n-\ell}$ are observed. Then the learned parameter $\widehat{\alpha}_{OW}$ is obtained using $\{(\widetilde{x}_i, \widetilde{y}_i)\}_{i=1}^{\ell}$ and $\{(x_i, y_i)\}_{i=1}^{n-\ell}$ as

$$\widehat{\alpha}_{OW} = \operatorname*{argmin}_{\alpha} \left[\sum_{i=1}^{\ell} \left(\widehat{f}(\widetilde{x}_i) - \widetilde{y}_i \right)^2 + \sum_{i=1}^{n-\ell} \frac{q(x_i)}{p(x_i)} \left(\widehat{f}(x_i) - y_i \right)^2 \right]. \tag{22}$$

Note that J_{OW} depends on the realization of $\{\widetilde{x}_i\}_{i=1}^{\ell}$, but is independent of the realization of $\{x_i\}_{i=1}^{n-\ell}$.

Comparison with J: Kanamori and Shimodaira (2003) proved that for $\ell = o(n)$, $\lim_{n\to\infty} \ell = \infty$, and $\delta = O(1)$,

$$\mathop{\mathbb{E}}_{\{x_i\}} \mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W = \frac{1}{n} J_{OW} + o(n^{-1}),$$

by which the use of J_{OW} can be justified. The order of δ required above is weaker than that required in J. Therefore, J_{OW} may have a wider range of applications than J. However, this property may not be practically valuable since learning with totally misspecified models (i.e., $\delta = O(1)$) may not work well because of the model error.

Due to the two-stage sampling scheme, the above method has several weaknesses. First, ℓ training input points should be gathered following $q(x)$ in the first stage, which implies that users are only allowed to optimize the location of $n-\ell$ remaining training input points. This may be critical when the total number n is not so large. Second, the performance depends on the choice of ℓ, so it has to be appropriately determined. Using $\ell = O(n^{1/2})$ is recommended in Kanamori and

Shimodaira (2003), but the exact choice of ℓ seems still open. Third, J_{OW} is an estimator of G_W, but the finally obtained parameter by this algorithm is not $\widehat{\alpha}_W$ but $\widehat{\alpha}_{OW}$. Therefore, this difference can degrade the performance.[6]

In Section 4, we experimentally compare J and J_{OW}.

4. Numerical Examples

In this section, we quantitatively compare the proposed and existing active learning methods through numerical experiments.

4.1 Toy Data Set

We first illustrate how the proposed and existing methods behave under a controlled setting.

Setting: Let the input dimension be $d = 1$ and the learning target function be

$$f(x) = 1 - x + x^2 + \delta r(x),$$

where

$$r(x) = \frac{z^3 - 3z}{\sqrt{6}} \quad \text{with} \quad z = \frac{x - 0.2}{0.4}. \tag{23}$$

Let the number of training examples to gather be $n = 100$ and $\{\varepsilon_i\}_{i=1}^n$ be i.i.d. Gaussian noise with mean zero and standard deviation 0.3. Let the test input density $q(x)$ be the Gaussian density with mean 0.2 and standard deviation 0.4, which is assumed to be known in this illustrative simulation. See the bottom graph of Figure 5 for the profile of $q(x)$. Let the number of basis functions be $b = 3$ and the basis functions be

$$\varphi_i(x) = x^{i-1} \quad \text{for } i = 1, 2, \ldots, b.$$

Note that for these basis functions, the residual function $r(x)$ in Eq.(23) fulfills Eqs.(4) and (5). Let us consider the following three cases.

$$\delta = 0, 0.005, 0.05, \tag{24}$$

which correspond to "*correctly specified*", "*approximately correct*", and "*misspecified*" cases, respectively. See the top graph of Figure 5 for the profiles of $f(x)$ with different δ.

As a set of training input densities, $\widehat{\mathcal{P}}$, we use the Gaussian densities with mean 0.2 and standard deviation $0.4c$, where

$$c = 0.8, 0.9, 1.0, \ldots, 2.5.$$

See the bottom graph of Figure 5 again for the profiles of $p(x)$ with different c.

In this experiment, we compare the performance of the following methods:

(ALICE): c is determined so that J given by Eq.(12) is minimized. WLS given by Eq.(8) is used for estimating the parameters.

6. It is possible to resolve this problem by not using $\{(\widetilde{x}_i, \widetilde{y}_i)\}_{i=1}^{\ell}$ gathered in the first stage for estimating the parameter (cf. Eq.(22)). However, this may yield further degradation of the performance because only $n - \ell$ training examples are used for learning.

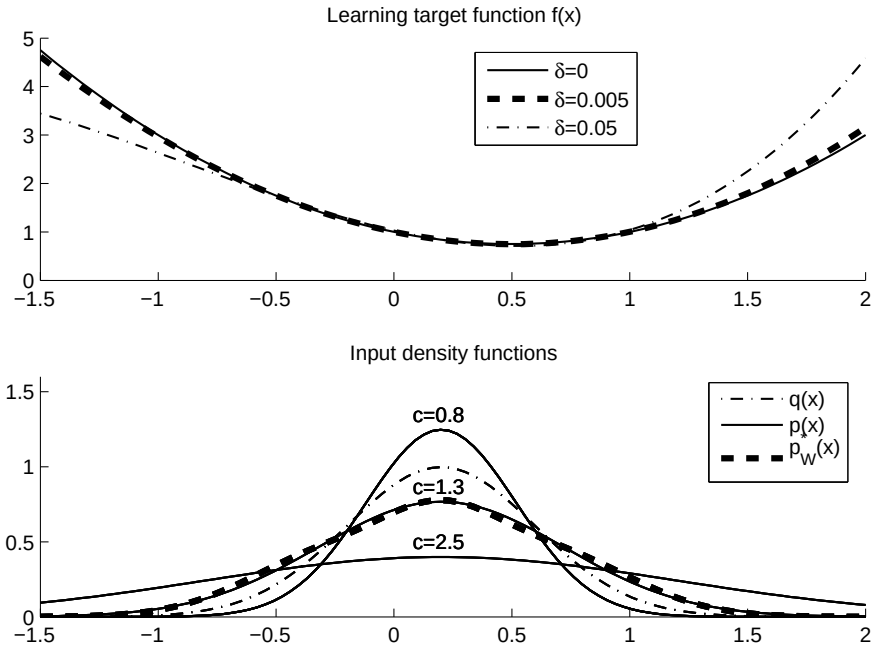


Figure 5: Learning target function and input density functions.

(W): c is determined so that J_W given by Eq.(17) is minimized. WLS is used for estimating the parameters.

(W*): $p^*_W(x)$ given by Eq.(18) is used as the training input density. The profile of $p^*_W(x)$ under the current setting is illustrated in the bottom graph of Figure 5, showing that $p^*_W(x)$ is similar to the Gaussian density with $c = 1.3$. WLS is used for estimating the parameters.

(OW): First, ℓ training input points are created following the test input density $q(x)$, and corresponding training output values are observed. Based on the ℓ training examples, c is determined so that J_{OW} given by Eq.(21) is minimized. Then $n-\ell$ remaining training input points are created following the determined input density. The combination of OLS and WLS given by Eq.(22) is used for estimating the parameters. We set $\ell = 25$, which we experimentally confirmed to be a reasonable choice in this illustrative simulation.

(O): c is determined so that J_O given by Eq.(13) is minimized. OLS given by Eq.(7) is used for estimating the parameters.

(Passive): Following the test input density $q(x)$, training input points $\{x_i\}_{i=1}^n$ are created. OLS is used for estimating the parameters.

For (W*), we generate the random number following $p^*_W(x)$ by the rejection method (see e.g., Knuth, 1998). We run this simulation 1000 times for each δ in Eq.(24).

Accuracy of Generalization Error Prediction: First, we evaluate the accuracy of J, J_W, J_{OW}, and J_O as predictors of the generalization error. Note that J and J_W are predictors of G_W. J_{OW} is also derived as a predictor of G_W, but the finally obtained generalization error by (OW) is G_{OW}, which

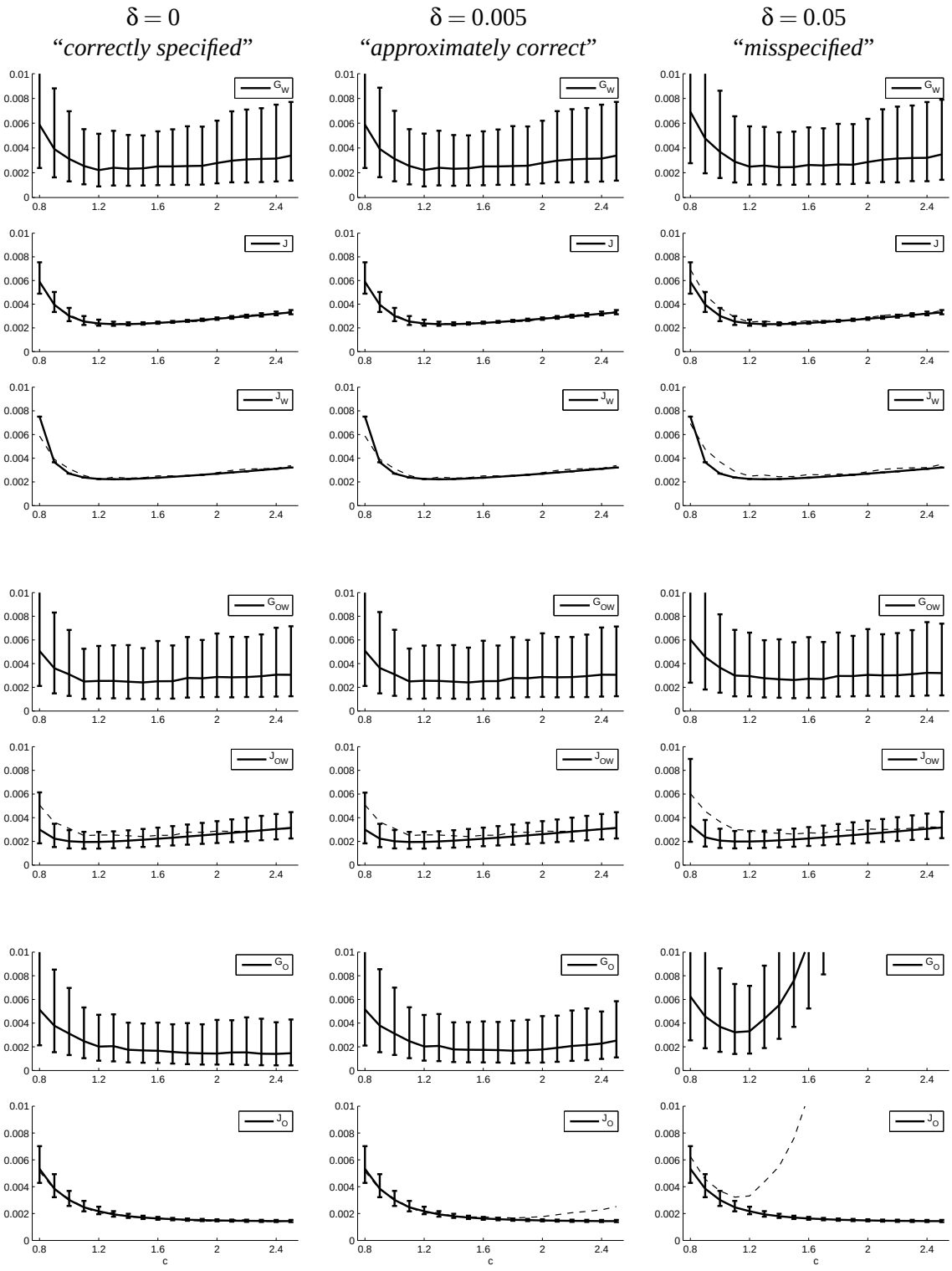


Figure 6: The means and (asymmetric) standard deviations of G_W, J, J_W, G_{OW}, J_{OW}, G_O, and J_O over 1000 runs as functions of c. The dashed curves show the means of the generalization error that corresponding active learning criteria are trying to predict.

is the generalization error G for the learned function obtained by the combination of OLS and WLS (see Eq.(22)). Therefore, J_{OW} should be evaluated as a predictor of G_{OW}. J_O is a predictor of G_O.

In Figure 6, the means and standard deviations of G_W, J, J_W, G_{OW}, J_{OW}, G_O, and J_O over 1000 runs are depicted as functions of c by the solid curves. Here the upper and lower error bars are calculated separately since the distribution is not symmetric. The dashed curves show the means of the generalization error that corresponding active learning criteria are trying to predict. Note that J, J_W, and J_O are multiplied by $\sigma^2 = (0.3)^2$ so that comparison with G_W and G_O are clear. By definition, G_W, G_{OW}, and G_O do not include the constant C defined by Eq.(6). The values of C for $\delta = 0$, 0.005, and 0.05 are 0, 2.32×10^{-5}, and 2.32×10^{-3}, respectively.

These graphs show that when $\delta = 0$ ("*correctly specified*"), J and J_W give accurate predictions of G_W. Note that J_W does not depend on the training input points $\{x_i\}_{i=1}^n$ so it does not fluctuate over 1000 runs. J_{OW} is slightly biased toward the negative direction for small c. We conjecture that this is caused by the small sample effect. However, the profile of J_{OW} still roughly approximates that of G_{OW}. J_O gives accurate predictions of G_O. When $\delta = 0.005$ ("*approximately correct*"), J, J_W, and J_{OW} work similarly to the case with $\delta = 0$, i.e., J and J_W are accurate and J_{OW} is negatively biased. On the other hand, J_O behaves slightly differently: it tends to be biased toward the negative direction for large c. Finally, when $\delta = 0.05$ ("*misspecified*"), J and J_W still give accurate predictions, although they slightly have a negative bias for small c. J_{OW} still roughly approximates G_{OW}, while J_O gives totally different profile from G_O.

These results show that as approximations of the generalization error, J and J_W are accurate and robust against the misspecification of models. J_{OW} is also reasonably accurate, although it tends to be rather inaccurate for small c. J_O is accurate in the correctly specified case, but it becomes totally inaccurate once the correctness of the model is violated.

Note that, by definition, J, J_W and J_O do not depend on the learning target function. Therefore, in the simulation, they give the same values for all δ (J and J_O depend on the realization of $\{x_i\}_{i=1}^n$ so they may have a small fluctuation). On the other hand, the generalization error, of course, depends on the learning target function even if the constant C is not included, since the training output values depend on it. Note that the bias depends on δ, but the variance does not. The simulation results show that the profile of G_O changes heavily as the degree of model misspecification increases. This would be caused by the increase of the bias since OLS is not unbiased even asymptotically. On the other hand, J_O stays the same as δ increases. As a result, J_O becomes a very poor predictor for a large δ. In contrast, the profile of G_W appears to be very stable against the change in δ, which is in good agreement with the theoretical fact that WLS is asymptotically unbiased. Thanks to this property, J and J_W are more accurate than J_O for misspecified models.

Obtained Generalization Error: In Table 1, the mean and standard deviation of the generalization error obtained by each method are described. The best method and comparable ones by the *t-test* (e.g., Henkel, 1979) at the significance level 5% are indicated with boldface. In Figure 7, the box-plot expression of the obtained generalization error is depicted. Note that the values described in Figure 6 correspond to G (the constant C is not included), while the values in Table 1 and Figure 7 correspond to G' which includes C (see Eq.(1)).

When $\delta = 0$, (O) works significantly better than other methods. Actually, in this case, training input densities that approximately minimize G_W, G_O, and G_{OW} were successfully found by (ALICE), (W), (OW), and (O). This implies that the difference in the error is caused not by the quality of the active learning criteria, but by the difference between WLS and OLS: WLS generally has

	$\delta = 0$	$\delta = 0.005$	$\delta = 0.05$
(ALICE)	2.08 ± 1.95	$\mathbf{2.10 \pm 1.96}$	$\mathbf{4.61 \pm 2.12}$
(W)	2.40 ± 2.15	2.43 ± 2.15	4.89 ± 2.26
(W*)	2.32 ± 2.02	2.35 ± 2.02	4.84 ± 2.14
(OW)	3.09 ± 3.03	3.13 ± 3.00	5.95 ± 3.58
(O)	$\mathbf{1.31 \pm 1.70}$	2.53 ± 2.23	124 ± 67.4
(Passive)	3.11 ± 2.78	3.14 ± 2.78	6.01 ± 3.43

All values in the table are multiplied by 10^3.

Table 1: The mean and standard deviation of the generalization error obtained by each method for the toy data set. Here we describe the value G' that includes the constant C (see Eq.(6)). The best method and comparable ones by the t-test at the significance level 5% are indicated with boldface. The value of (O) for $\delta = 0.05$ is extremely large but it is not a typo.

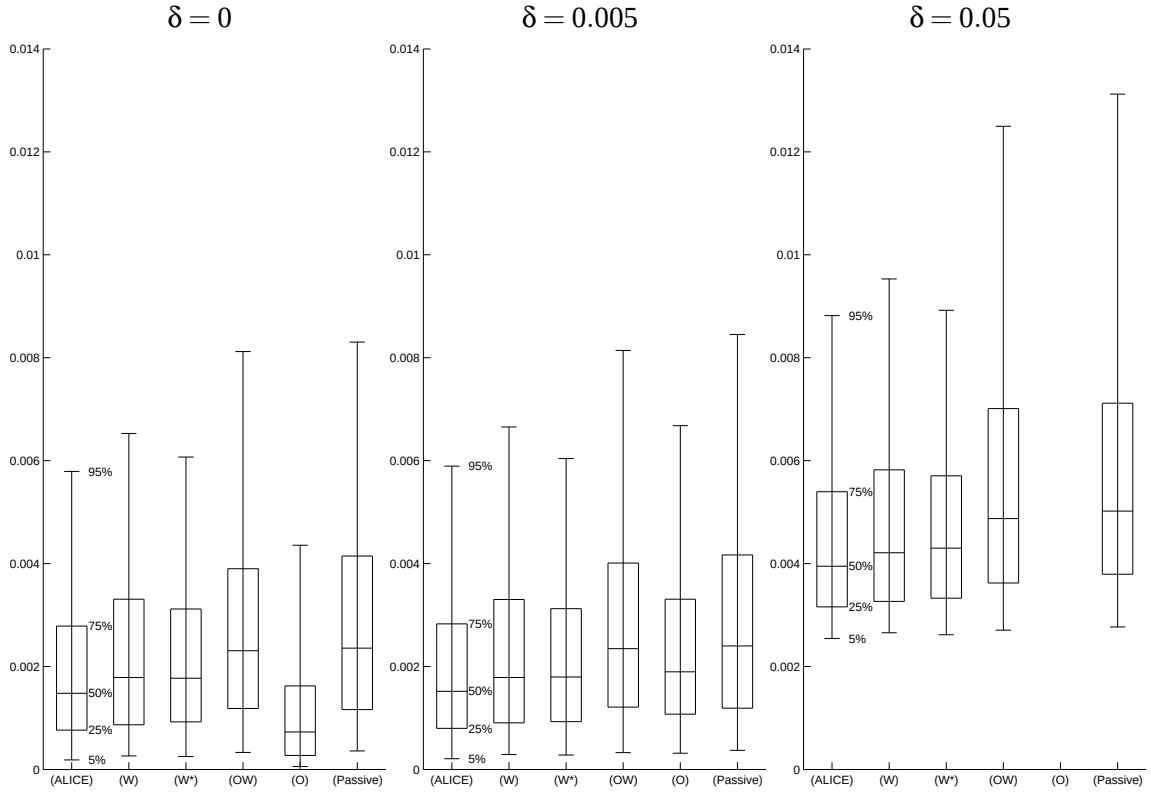


Figure 7: Box-plots of the generalization error obtained by each method for the toy data set. Here we plot the value G' that includes the constant C (see Eq.(6)). The value of (O) for $\delta = 0.05$ is not plotted because it is extremely large.

larger variance than OLS (Shimodaira, 2000). Therefore, when $\delta = 0$, OLS would be more accurate than WLS since both WLS and OLS are unbiased. Although (ALICE), (W), (W*), and (OW) are outperformed by (O), they still work better than (Passive). Note that (ALICE) is significantly better than (W), (W*), (OW), and (Passive) by the t-test. The box-plot shows that (ALICE) outperforms (W), (W*), and (OW) particularly in upper quantiles.

When $\delta = 0.005$, (ALICE) gives significantly smaller errors than other methods. All the methods except (O) work similarly to the case with $\delta = 0$, while (O) tends to perform poorly. This result is surprising since the learning target functions with $\delta = 0$ and $\delta = 0.005$ are visually almost the same, as illustrated in the top graph of Figure 5. Therefore, it intuitively seems that the result when $\delta = 0.005$ is not much different from the result when $\delta = 0$. However, this slight difference appears to make (O) unreliable.

When $\delta = 0.05$, (ALICE) again works significantly better than others. (W) and (W*) still work reasonably well. The box-plot shows that (ALICE) is better than (W) and (W*) particularly in upper quantiles. The performance of (OW) is slightly degraded, although it is still better than (Passive). (O) gives extremely large errors.

The above results are summarized as follows. For all three cases ($\delta = 0, 0.005, 0.05$), (ALICE), (W), (W*), and (OW) work reasonably well and consistently outperform (Passive). Among them, (ALICE) appears to be better than (W), (W*), and (OW) for all three cases. (O) works excellently in the correctly specified case, although it tends to perform poorly once the correctness of the model is violated. Therefore, (ALICE) is found to work well overall and is robust against the misspecification of models for this toy data set.

4.2 Benchmark Data Sets

Here we use eight regression benchmark data sets provided by DELVE (Rasmussen et al., 1996): *Bank-8fm, Bank-8fh, Bank-8nm, Bank-8nh, Kin-8fm, Kin-8fh, Kin-8nm,* and *Kin-8nh*. Each data set includes 8192 samples, consisting of 8-dimensional input points and 1-dimensional output values. For convenience, every attribute is normalized into $[0,1]$.

Suppose we are given all 8192 *input* points (i.e., unlabeled samples). Note that output values are kept unknown at this point. From this pool of unlabeled samples, we choose $n = 300$ input points $\{x_i\}_{i=1}^n$ for training and observe the corresponding output values $\{y_i\}_{i=1}^n$. The task is to predict the output values of all 8192 unlabeled samples.

In this experiment, the test input density $q(x)$ is unknown. So we estimate it using the uncorrelated multi-dimensional Gaussian density:

$$q(x) = \frac{1}{(2\pi\widehat{\gamma}_{MLE}^2)^{\frac{d}{2}}} \exp\left(-\frac{\|x-\widehat{\mu}_{MLE}\|^2}{2\widehat{\gamma}_{MLE}^2}\right),$$

where $\widehat{\mu}_{MLE}$ and $\widehat{\gamma}_{MLE}$ are the maximum likelihood estimates of the mean and standard deviation obtained from all 8192 unlabeled samples. Let $b = 50$ and the basis functions be Gaussian basis functions with variance 1:

$$\varphi_i(x) = \exp\left(-\frac{\|x-t_i\|^2}{2}\right) \qquad \text{for } i = 1,2,\ldots,b,$$

where $\{t_i\}_{i=1}^b$ are template points randomly chosen from the pool of unlabeled samples.

	Bank-8fm	Bank-8fh	Bank-8nm	Bank-8nh
(ALICE)	2.10 ± 0.17	6.83 ± 0.44	$\mathbf{1.11 \pm 0.09}$	4.19 ± 0.29
(W)	2.26 ± 0.21	7.21 ± 0.52	1.22 ± 0.12	4.40 ± 0.38
(OW)	2.31 ± 0.25	7.39 ± 0.63	1.25 ± 0.15	4.52 ± 0.39
(O)	$\mathbf{1.91 \pm 0.16}$	$\mathbf{6.20 \pm 0.24}$	1.32 ± 0.14	$\mathbf{4.02 \pm 0.21}$
(Passive)	2.31 ± 0.26	7.45 ± 0.61	1.26 ± 0.14	4.51 ± 0.38

	Kin-8fm	Kin-8fh	Kin-8nm	Kin-8nh
(ALICE)	$\mathbf{1.62 \pm 0.58}$	$\mathbf{3.50 \pm 0.63}$	$\mathbf{34.97 \pm 1.90}$	$\mathbf{47.21 \pm 1.97}$
(W)	$\mathbf{1.70 \pm 0.62}$	$\mathbf{3.64 \pm 0.73}$	36.60 ± 2.05	49.15 ± 2.88
(OW)	$\mathbf{1.73 \pm 0.63}$	3.73 ± 0.78	37.29 ± 2.94	49.64 ± 3.11
(O)	3.03 ± 1.60	4.85 ± 1.96	38.65 ± 3.09	48.86 ± 2.66
(Passive)	$\mathbf{1.77 \pm 0.68}$	3.73 ± 0.79	37.38 ± 3.05	49.69 ± 3.06

All values in the table are multiplied by 10^3.

Table 2: The means and standard deviations of the test error for DELVE data sets. The best method and comparable ones by the t-test at the significance level 5% are indicated with boldface.

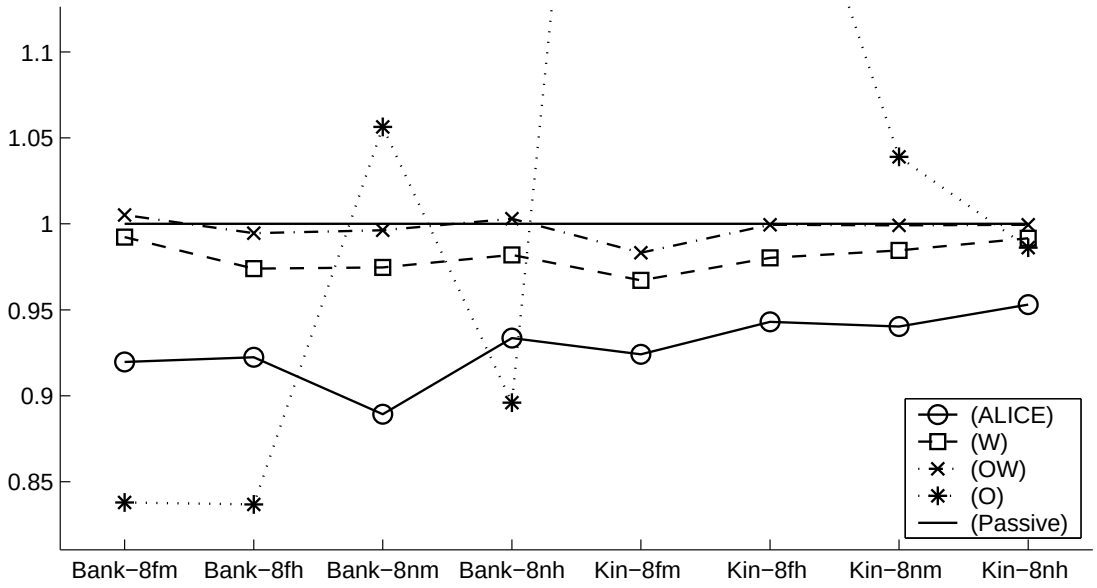


Figure 8: The means of the test error of (ALICE), (W), (OW), and (O) normalized by the test error of (Passive).

We select the training input density $p(x)$ from the set of uncorrelated multi-dimensional Gaussian densities with mean $\widehat{\mu}_{MLE}$ and standard deviation $c\widehat{\gamma}_{MLE}$, where

$$c = 0.7, 0.75, 0.8, \ldots, 2.4.$$

We again compare the active learning methods tested in Section 4.1. However, we do not test (W*) here because we could not efficiently generate random numbers following $p^*_W(x)$ by the rejection method. For (OW), we set $\ell = 100$ which we experimentally confirmed to be reasonable.

In this simulation, we can not create the training input points in an arbitrary location because we only have 8192 samples in the pool. Here, we first create provisional input points following the determined training input density, and then choose the input points from the pool of unlabeled samples that are closest to the provisional input points. In this simulation, the expectation over the test input density $q(x)$ in the matrix U is calculated by the empirical average over all 8192 unlabeled samples since the true test error is also calculated as such. For each data set, we run this simulation 100 times, by changing the template points $\{t_i\}_{i=1}^b$ in each run.

The means and standard deviations of the test error over 100 runs are described in Table 2. This shows that (ALICE) works very well for five out of eight data sets. For the other three data sets, (O) works significantly better than other methods. (W) works well and is comparable to (ALICE) for two data sets, but is outperformed by (ALICE) for the other six data sets. (OW) is overall comparable to (Passive).

Figure 8 depicts the means of the test error of (ALICE), (W), (OW), and (O) normalized by the test error of (Passive): For each run, the test errors of (ALICE), (W), (OW), and (O) are divided by the test error of (Passive), and then the values are averaged over 100 runs. This graph shows that (ALICE) is better than (W), (OW), and (Passive) for all eight data sets. (O) works very well for three data sets, but it is comparable or largely outperformed by (Passive) for the other five data sets. (W) also works reasonably well, although it is outperformed by (ALICE) overall. (OW) is on par with (Passive). Overall, (ALICE) is shown to be stable and works well for the benchmark data sets.

We also carried out similar simulations for Gaussian basis functions with variance 0.5 and 2. The results had similar tendencies, i.e., (ALICE) is overall shown to be stable and works well, so we omit the detail.

5. Conclusions

In this paper, we proposed a new active learning method based on the importance-weighted least-squares learning. The numerical study showed that the proposed method works well overall and compares favorably to existing WLS-based methods and the passive learning scheme. Although the proposed method is outperformed by the existing OLS-based method when the model is correctly specified, the existing OLS-based method tends to perform very poorly once the correctness of the model is violated. Therefore, the existing OLS-based method may not be reliable in practical situations where the correctness of the model may not be fulfilled. On the other hand, the proposed method is shown to be robust against the misspecification of models and therefore reliable.

Our criterion is shown to be a variant of the criterion proposed by Wiens (2000). Indeed, we showed that they are asymptotically equivalent. However, an important difference is that we predict the conditional expectation of the generalization error given training input points, while in Wiens (2000), the full expectation of the generalization error is predicted. As described in Section 1, the conditional-expectation approach conceptually gives a finer choice of the training input density

than the full-expectation approach. Theoretically, we proved that the proposed criterion is a better estimate of the single-trial generalization error than Wiens's criterion (see Section 3.2).

An advantage of Wiens's criterion is that the optimal training input density can be obtained analytically, while we do not yet have such an analytic solution for the proposed criterion. In the current paper, we resorted to a naive optimization scheme: prepare a finite set of input densities and choose the best one from the set. The performance of this naive optimization scheme depends heavily on the choice of the set of densities. In practice, using a set of input densities which consist of the optimal density analytically found by Wiens's criterion and its variants would be a reasonable choice. It is also important to devise a better optimization strategy for the proposed active learning criterion, which currently remains open.

In theory, we assumed that the test input density is known. However, this may not be satisfied in practice. In experiments with benchmark data sets, the test input density is indeed unknown and is approximated by a Gaussian density. Although the simulation results showed that the proposed method consistently outperforms the passive learning scheme (given unlabeled samples), a more detailed analysis should be carried out to see how approximating the test input density affects the performance.

We discussed the active learning problem for *weakly* misspecified models. A natural extension of the proposed method is to be applicable to *strongly* misspecified models, as achieved in Kanamori and Shimodaira (2003). However, when the model is totally misspecified, even learning with the optimal training input points may not work well because of the model error. In such cases, it is important to carry out *model selection* (Akaike, 1974; Schwarz, 1978; Rissanen, 1978; Vapnik, 1998). In most of the active learning research—including the current paper, the location of the training input points are designed for a *single* model at hand. That is, the model should have been chosen *before* active learning is carried out. However, in practice, we may want to select the model as well as the location of the training input points. Devising a method for simultaneously optimizing the model and the location of the training input points would therefore be a more important and promising future direction. In Sugiyama and Ogawa (2003), a method of *active learning with model selection* has been proposed for the trigonometric polynomial models. However, its range of application is rather limited. We expect that the results given in this paper form a solid basis for further pursuing this challenging issue.

Acknowledgments

The author would like to thank anonymous reviewers for their helpful comments, which highly helped him to improve the manuscript. Particularly, the normalization of the residual function is pointed out by one of the reviewers. He also acknowledges Dr. Motoaki Kawanabe for fruitful discussions on the accuracy of generalization error estimators. Special thanks also go to the members of Fraunhofer FIRST.IDA for their comments on various aspects of the proposed method when the author gave a talk at the seminar. This work is supported by MEXT (Grant-in-Aid for Young Scientists 17700142).

Appendix A. Proof of Lemma 1

A simple calculation yields that B and V are expressed as

$$B = \langle U(\mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha} - \alpha^*), \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha} - \alpha^* \rangle,$$
$$V = \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \langle U(\widehat{\alpha} - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}), \widehat{\alpha} - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha} \rangle.$$

Let

$$z_g = (g(x_1), g(x_2), \ldots g(x_n))^\top,$$
$$z_r = (r(x_1), r(x_2), \ldots r(x_n))^\top.$$

By definition, it holds that

$$z_g = X\alpha^*.$$

Then we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_W - \alpha^* &= L_W(z_g + \delta z_r) - \alpha^* \\
&= (\tfrac{1}{n} X^\top D X)^{-1} \tfrac{1}{n} X^\top D (X\alpha^* + \delta z_r) - \alpha^* \\
&= \delta (\tfrac{1}{n} X^\top D X)^{-1} \tfrac{1}{n} X^\top D z_r.
\end{aligned}$$

By the law of large numbers (Rao, 1965), we have

$$\begin{aligned}
\lim_{n\to\infty} [\tfrac{1}{n} X^\top D X]_{i,j} &= \lim_{n\to\infty} \left(\frac{1}{n} \sum_{k=1}^{n} \frac{q(x_k)}{p(x_k)} \varphi_i(x_k) \varphi_j(x_k) \right) \\
&= \int_{\mathcal{D}} \frac{q(x)}{p(x)} \varphi_i(x) \varphi_j(x) p(x) dx \\
&= O_p(1).
\end{aligned}$$

Furthermore, by the central limit theorem (Rao, 1965), it holds for sufficiently large n,

$$\begin{aligned}
[\tfrac{1}{n} X^\top D z_r]_i &= \frac{1}{n} \sum_{k=1}^{n} r(x_k) \varphi_i(x_k) \frac{q(x_k)}{p(x_k)} \\
&= \int_{\mathcal{D}} r(x) \varphi_i(x) \frac{q(x)}{p(x)} p(x) dx + O_p(n^{-\frac{1}{2}}) \\
&= O_p(n^{-\frac{1}{2}}),
\end{aligned}$$

where the last equality follows from Eq.(4). Therefore, we have

$$\begin{aligned}
B_W &= \langle U(\mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_W - \alpha^*), \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_W - \alpha^* \rangle \\
&= O_p(\delta^2 n^{-1}).
\end{aligned}$$

It holds that $U = O_p(1)$ and

$$\begin{aligned}
L_W L_W^\top &= (\tfrac{1}{n} X^\top D X)^{-1} \tfrac{1}{n^2} X^\top D^2 X (\tfrac{1}{n} X^\top D X)^{-1} \\
&= O_p(n^{-1}).
\end{aligned}$$

Then we have

$$\begin{aligned} V_W &= \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \langle U(\widehat{\alpha}_W - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_W), \widehat{\alpha}_W - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_W \rangle \\ &= \sigma^2 \mathrm{tr}(U L_W L_W^\top) \\ &= O_p(n^{-1}), \end{aligned}$$

which concludes the proof. ■

Appendix B. Proof of Lemma 2

It holds that

$$\begin{aligned} \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_O - \alpha^* &= L_O(z_g + \delta z_r) - \alpha^* \\ &= (\tfrac{1}{n} X^\top X)^{-1} \tfrac{1}{n} X^\top (X\alpha^* + \delta z_r) - \alpha^* \\ &= \delta (\tfrac{1}{n} X^\top X)^{-1} \tfrac{1}{n} X^\top z_r. \end{aligned}$$

By the law of large numbers, we have

$$\begin{aligned} \lim_{n\to\infty} [\tfrac{1}{n} X^\top X]_{i,j} &= \lim_{n\to\infty} \left(\frac{1}{n} \sum_{k=1}^{n} \varphi_i(x_k) \varphi_j(x_k) \right) \\ &= \int_{\mathcal{D}} \varphi_i(x) \varphi_j(x) p(x) dx \\ &= O_p(1). \end{aligned}$$

Furthermore, by the central limit theorem, it holds for sufficiently large n,

$$\begin{aligned} [\tfrac{1}{n} X^\top z_r]_i &= \frac{1}{n} \sum_{k=1}^{n} r(x_k) \varphi_i(x_k) \\ &= \int_{\mathcal{D}} r(x) \varphi_i(x) p(x) dx + O_p(n^{-\frac{1}{2}}) \\ &= O_p(1). \end{aligned}$$

Therefore, we have

$$\begin{aligned} B_O &= \langle U(\mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_O - \alpha^*), \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_O - \alpha^* \rangle \\ &= O_p(\delta^2). \end{aligned}$$

It holds that $U = O_p(1)$ and

$$\begin{aligned} L_O L_O^\top &= (\tfrac{1}{n} X^\top X)^{-1} \tfrac{1}{n^2} X^\top X (\tfrac{1}{n} X^\top X)^{-1} \\ &= O_p(n^{-1}). \end{aligned}$$

Then we have

$$\begin{aligned} V_O &= \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \langle U(\widehat{\alpha}_O - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_O), \widehat{\alpha}_O - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} \widehat{\alpha}_O \rangle \\ &= \sigma^2 \mathrm{tr}(U L_O L_O^\top) \\ &= O_p(n^{-1}), \end{aligned}$$

which concludes the proof. ■

Appendix C. Proof of Lemma 3

The central limit theorem (see e.g., Rao, 1965) asserts that

$$L_W L_W^\top = \tfrac{1}{n} U^{-1} T U^{-1} + O_p(n^{-\frac{3}{2}}),$$

from which we have Eq.(19) ■

Appendix D. Proof of Lemma 4

It holds that

$$\begin{aligned}
\mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J_W - G_W)^2 &= \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J_W - \sigma^2 J + \sigma^2 J - G_W)^2 \\
&= (\sigma^2 J_W - \sigma^2 J)^2 + \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J - G_W)^2 \\
&\quad + 2 \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J_W - \sigma^2 J)(\sigma^2 J - G_W). \qquad (25)
\end{aligned}$$

Eq.(19) implies

$$(\sigma^2 J_W - \sigma^2 J)^2 = O_p(n^{-3}).$$

Eqs.(19) and (10) imply

$$\begin{aligned}
2 \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J_W - \sigma^2 J)(\sigma^2 J - G_W) &= 2(\sigma^2 J_W - \sigma^2 J)(\sigma^2 J - \mathop{\mathbb{E}}_{\{\varepsilon_i\}} G_W) \\
&= -2(\sigma^2 J_W - \sigma^2 J) B_W \\
&= O_p(\delta^2 n^{-\frac{5}{2}}). \qquad (26)
\end{aligned}$$

If $\delta = o_p(n^{-\frac{1}{4}})$ and the term of order $o_p(n^{-3})$ (i.e., Eq.(26)) is ignored in Eq.(25), we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J_W - G_W)^2 &= (\sigma^2 J_W - \sigma^2 J)^2 + \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J - G_W)^2 \\
&\geq \mathop{\mathbb{E}}_{\{\varepsilon_i\}} (\sigma^2 J - G_W)^2,
\end{aligned}$$

which concludes the proof. ■

References

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–723, 1974.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

K. Fukumizu. Statistical active learning in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 11(1):17–26, 2000.

R. E. Henkel. *Tests of Significance*. SAGE Publication, Beverly Hills, 1979.

T. Kanamori and H. Shimodaira. Active learning algorithm using the maximum weighted log-likelihood estimator. *Journal of Statistical Planning and Inference*, 116(1):149–162, 2003.

J. Kiefer. Optimum experimental designs. *Journal of the Royal Statistical Society, Series B*, 21: 272–304, 1959.

D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Massachusetts, 1998.

D. J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

F. Pukelsheim. *Optimal Design of Experiments*. John Wiley & Sons, 1993.

C. R. Rao. *Linear Statistical Inference and Its Applications*. Wiley, New York, 1965.

C. E. Rasmussen, R. M. Neal, G. E. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani. The DELVE manual, 1996. URL `http://www.cs.toronto.edu/~delve/`.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

M. Sugiyama and H. Ogawa. Incremental active learning for optimal generalization. *Neural Computation*, 12(12):2909–2940, 2000.

M. Sugiyama and H. Ogawa. Active learning for optimal generalization in trigonometric polynomial models. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A(9):2319–2329, 2001.

M. Sugiyama and H. Ogawa. Active learning with model selection — Simultaneous optimization of sample points and models for trigonometric polynomial models. *IEICE Transactions on Information and Systems*, E86-D(12):2753–2763, 2003.

V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc., New York, 1998.

D. P. Wiens. Robust weights and designs for biased regression models: Least squares and generalized M-estimation. *Journal of Statistical Planning and Inference*, 83(2):395–412, 2000.

Journal of Machine Learning Research 7 (2006) 167–189 Submitted 3/04; Revised 9/05; Published 2/06

MinReg: A Scalable Algorithm for Learning Parsimonious Regulatory Networks in Yeast and Mammals

Dana Pe'er DPEER@GENETICS.MED.HARVARD.EDU
Genetics Department
Harvard Medical School
Boston, MA 02115, USA

Amos Tanay AMOS@POST.TAU.AC.IL
Computer Science Department
Tel Aviv University
Tel Aviv, Israel

Aviv Regev AREGEV@CGR.HARVARD.EDU
Bauer Center for Genomics Research
Harvard University
Cambridge, MA 02138, USA

Editor: Tommi Jaakkola

Abstract

In recent years, there has been a growing interest in applying Bayesian networks and their extensions to reconstruct *regulatory networks* from gene expression data. Since the gene expression domain involves a large number of variables and a limited number of samples, it poses both computational and statistical challenges to Bayesian network learning algorithms. Here we define a constrained family of Bayesian network structures suitable for this domain and devise an efficient search algorithm that utilizes these structural constraints to find high scoring networks from data. Interestingly, under reasonable assumptions on the underlying probability distribution, we can provide performance guarantees on our algorithm. Evaluation on real data from yeast and mouse, demonstrates that our method cannot only reconstruct a high quality model of the yeast regulatory network, but is also the first method to scale to the complexity of mammalian networks and successfully reconstructs a reasonable model over thousands of variables.

Keywords: Bayesian networks, structure learning, gene networks, gene expression, approximation algorithms

1. Introduction

Learning Bayesian network structure from data (Cooper and Herskovits, 1992; Heckerman et al., 1994) and its application to reconstruct *gene regulatory networks* from biological data (Friedman et al., 2000; Pe'er et al., 2001; Hartemink et al., 2002; Ong et al., 2002; Imoto et al., 2002; Yoo et al., 2002) is a subject of current research.

Regulatory networks control the expression of thousands of genes in a living cell, modulating the expression levels of individual genes based on external and internal conditions. To regulate

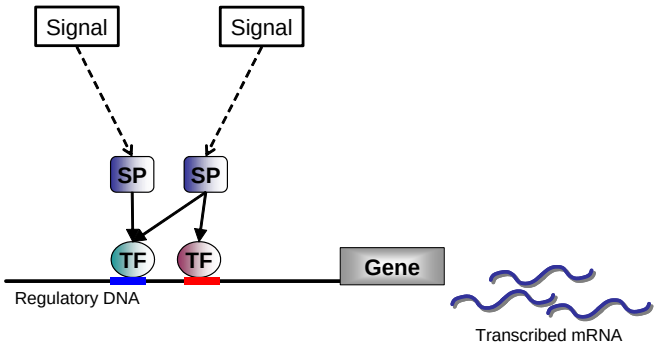


Figure 1: Biological regulation: Signals activate signaling molecules (SM), which in turn activate transcription factors (TF). When activated, these bind to DNA regulatory sequences. Combinations of such binding events control the levels of mRNA transcription in a combinatorial manner.

the expression of a gene, specialized proteins called *transcription factors (TFs)* bind to regulatory sequences on the DNA of *target genes* and work in a combinatorial fashion to ensure the correct amount is being transcribed (Figure 1). The behavior of those transcription factors is in turn controlled by the cell's environment through the action of *signaling proteins (SPs)*. The combined network of transcription factors and signaling proteins forms a regulatory program controlling the expression of individual genes directly (by regulator TFs) and indirectly (by regulator SPs). Since these networks serve as the information processing devices of cells, it is of great interest to uncover their structure and the regulation functions that they encode.

How can we learn such regulatory programs? An experimental technique, called *DNA microarrays* allows us to simultaneously measure the expression of thousands of genes under various conditions and perturbations, providing biologists with global observations of the workings of the cell. Importantly, microarrays measure not only the expression levels of target genes, but also of genes encoding regulators - TFs and SPs. As has been previously demonstrated (Pe'er et al., 2001, 2002; Segal et al., 2003), in many cases a TF's expression level is a good proxy to its activity, allowing us to construct a network that relates the gene expression of a target gene to the gene expression of its regulators. However, there are also numerous cases where a TF's activity is not determined by its expression level, but rather by other types of biochemical events, that that are unobserved in microarray data. Fortunately, in some of these cases, a change in the expression of indirect regulators (such as SPs that control the TF's activity) may be observed in microarray measurements, allowing us to detect an indirect regulatory relation in lieu of the direct event.

Following these biological considerations, it is expected that regulatory interactions between the genes would often result in corresponding statistical dependencies between random variables representing their expression. Thus, a Bayesian network approach to regulatory network reconstruction treats the expression level of each gene as a random variable and attempts to estimate the structural features of the dependencies in their joint probability distribution from data. Bayesian networks are particularly well suited for this domain, as has been demonstrated by early studies (Friedman et al., 2000; Pe'er et al., 2001; Hartemink et al., 2002). First, experimental evidence indicates that

the regulatory network is sparse, such that only a few genes directly control the transcription of a given target (Martinez-Antonio and Collado-Vides, 2003; Shen-Orr et al., 2002; Lee et al., 2002). Second, microarray measurements are typically noisy, necessitating a probabilistic model. Finally, biological networks contains many important hidden variables (*e.g.* the actual activity level of the regulators) which can be handled well in a Bayesian networks framework.

Nevertheless, the biological domain raises several important challenges for learning Bayesian networks. The central difficulty is that contrary to previous applications, microarrays measure thousands of variables (genes) across at most a few hundred samples. Thus, even if a search for the optimal solution (over a prohibitively large space) was possible, statistical noise is likely to lead to spurious dependencies, resulting in models that significantly overfit the data.

This problem becomes even more pronounced when considering the complex regulatory networks of mammals. While Bayesian network approaches have been relatively successful in tackling networks of a unicellular model organism, the Baker's yeast *Saccharomyces cerevisiae*, they have yet to achieve similar success in mammalian systems, such as human or mouse cells. These organisms have considerably more complex regulatory systems, with a larger number of regulators and target genes, and much more complex combinatorial regulatory functions. Deciphering these networks can have significant implications to the understanding of animal development and common diseases. A central question toward these important applications is finding a parsimonious set of *major* regulators at the center of a given response, and distinguishing them from additional redundant regulators or by product effects.

In this paper, we propose a novel approach to address these issues. We enforce biologically-motivated restrictions to limit the search to simple network structures that significantly reduce the space of possible networks, while highlighting the most relevant biological information. We devise a search algorithm that utilizes these structural constraints to efficiently find high scoring networks. Furthermore, under reasonable assumptions on the underlying probability distribution, we provide guarantees on our algorithm's performance, thus providing an approximation algorithm for a certain class of Bayesian networks. This is of particular interest, because approximation algorithms for learning Bayesian networks have only been developed for polytrees(Dasgupta, 1999).

We evaluate the performance of our algorithm on synthetic and real gene expression data sets for both yeast and mammals. Our results show good structure reconstruction on synthetic data and that the model learned from gene expression data generalizes well to unseen test data. Importantly, our results also illustrate the ability of the learned models to successfully reconstruct biologically correct regulatory relations in complex mammalian systems.

2. Regulation Model

Our gene regulation model is a Bayesian network that describes regulatory relations between genes. In this network, each random variable corresponds to the gene expression level of a specific gene. If gene Y is a parent of gene X in the Bayesian network, we interpret this as "*Y regulates X*". We denote by $\mathbf{Pa}_X$ the set of all regulators (parents) of the gene (variable) X. Any gene that "regulates" in our model is termed a *regulator*. The key point behind to our approach is that we enforce a number of biologically motivated constraints to limit these regulators and the graph structure.

Unlike a standard Bayesian network, we limit the possible regulators (parents) in the network to a set of candidate regulators $\mathcal{C}$. Our candidate set $\mathcal{C}$ is chosen based on prior biological knowledge, and contains known and putative regulators in the organism being studied. Note, that while finding

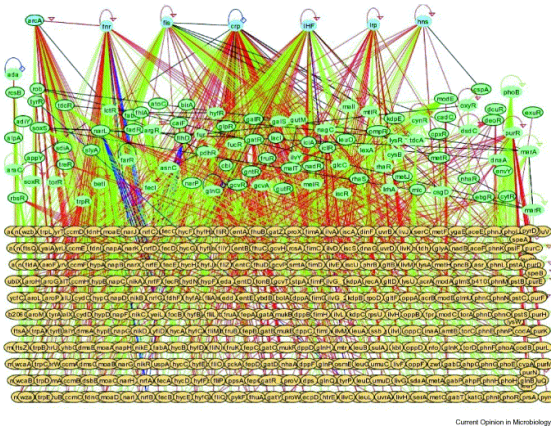


Figure 2: A literature reconstruction of the Bacterial *E. Coli* regulation network from (Martinez-Antonio and Collado-Vides, 2003). Notice this includes small top layer of regulators and many targets for each.

which genes in a genome may function as regulators in general is often tractable, finding which regulators are *active* in a data set is difficult. Our inference will focus on this latter question.

In fact, previous biological studies indicate that only a small fraction of all potential regulators may be active in a given data set. Accordingly, we also constrain the structural properties of the graph, seeking a Bayesian network in which only a limited number of genes are regulators, *i.e.*, have an outdegree greater than zero. Moreover, extensive studies in both bacteria and yeast (Shen-Orr et al., 2002; Martinez-Antonio and Collado-Vides, 2003; Lee et al., 2002) (Figure 2) indicate that each such "master regulatory gene" may affect the transcription of many genes (indeed, only 3-6% of the yeast and human genes respectively encode TFs). Thus, we expect each regulator to have a high out-degree. These constraints result in a graph of small depth, in which layers containing a small number of regulators control a large bottom layer of target genes, consistent with current biological understanding.

In addition to its biological relevance, this network structure has an obvious statistical motivation: Only when a gene consistently scores high as a parent for many genes, do we believe it indicates a true signal. An occasional high score as a parent of a single gene is attributed to spurious chance. Since learning an accurate genetic network is not possible in the current data paucity in the gene expression domain, our restrictions represent a reasonable first order approximation of the network which preserves its biological relevance. In fact, for most biological applications, false positives are significantly more "costly" than false negatives, and finding a robust set of key regulators whom are most strongly supported by the data (as offered by our model) is a more important goal then discovering their complete set of targets.

We now provide a formal definition of our model: A *regulation graph* is a Bayesian network with the following restrictions on its structure.

Definition 1 *Given a set of random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$, a set of* candidate regulators- *$\mathcal{C}$ and the constants d and k, we define a* regulation graph, *$\mathcal{G}$ to be a Bayesian network over $\mathcal{X}$ so that:*

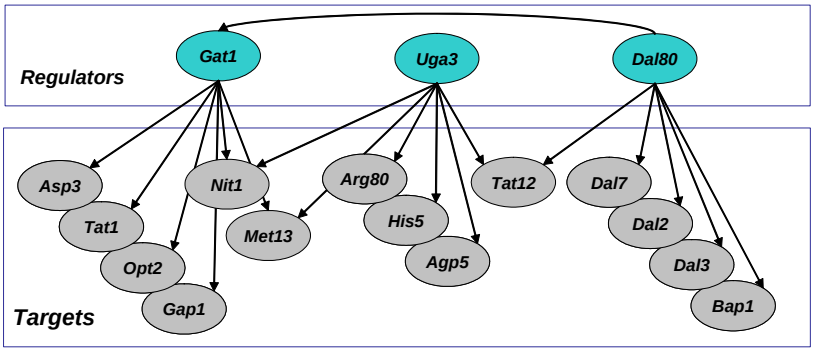


Figure 3: Regulation graph. The top layer is associated with the regulators and the bottom layer is associated with all other variables. The key concept behind the regulation graph is a small number of regulators, each with many targets. Note, the illustrated nitrogen catabolism response was automatically inferred by our algorithm from a gene expression data set.

- *All parents belong to the candidate set:* $\forall X, \mathbf{Pa}_X \subset \mathcal{C}$.
- *The number of parents for each variable (indegree) is bounded by* d*:* $\forall X, |\mathbf{Pa}_X| \leq d$.
- *The total number of parents in the model is bounded by* k*: We term the the union of all parent sets in the network to be the graph's* regulators*, denoted by* $\mathcal{R}$*, thus we constrain* $|\mathcal{R}| \leq k$.

∎

The graph structure is best visualized as a graph with a shallow depth: in the top layers, a small set of regulators (chosen from a large set $\mathcal{C}$), possibly regulating each other and in the bottom layer, all other variables (see Figure 3).

2.1 Optimization Problem

Since a regulation graph is a Bayesian network, the straightforward approach to learning its structure would be to use the typical heuristic greedy hill-climbing search (Heckerman, 1998) used for this task. This involves traversing the space of legal models in a greedy fashion using local operators such as adding, removing or reversing a single edge. At each step, the operation that best improves the score is chosen.

Unfortunately, the standard approach is likely to fail as the limited number of regulators specified by the regulation graph could be quickly used up. For example, we may wish to search for a regulation graph over 2000 variables, limited to 30 regulators. We begin with the empty graph and in a greedy fashion add the optimal edge at each iteration. In many of these iterations, a new regulator is added to the regulator set $\mathcal{R}$. Therefore, after little over 30 iterations, no new regulators could be added to $\mathcal{R}$ and all subsequent legal steps would only involve adding edges from regulators already in $\mathcal{R}$. While these regulators might be the best parents for a small set of variables, thousands of other variables remain unexplained in the model.

In fact, contrary to learning regular Bayesian networks, the choice of parents for a variable X in our model is no longer independent of the parents chosen for other variables. Since the total number

of parents in the model is limited to k, choosing a parent for one variable can limit the choice of parents for other variables. Thus, a regulator should be added to $\mathcal{R}$ only when it is a good parent for many variables concurrently. Any search algorithm we design must take this into account and add a regulator to a *set* of target genes in each greedy step.

Fortunately, once a regulator set $\mathcal{R}$ is given, finding the optimal regulation graph constrained to $\mathcal{R}$ is polynomial, and for most practical cases efficient. For any given variable, there is a small number, $\binom{k}{d}$, of possible parent sets (compared to $\binom{n}{d}$ possible parent sets for Bayesian networks bounded by an indegree of d).[1] Thus, it is quick to calculate the local score, denoted $Score(X;\mathbf{P})$, for all possible parents sets and choose the highest scoring parents.

$$\mathbf{Pa}_X = \text{argmax}_{\mathbf{P}\subset\mathcal{R},|\mathbf{P}|\leq d} Score(X;\mathbf{P}) \tag{1}$$

When $\mathcal{R}$ is not given, we can use use this property to efficiently evaluate the quality of any potential set of regulators.

Definition 2 *We define the* utility, $F(\mathbf{R})$ *of a regulator set* $\mathbf{R}$ *as:*

$$F(\mathbf{R}) = \sum_{i=1}^{n} \max_{\mathbf{P}\subset\mathbf{R},|\mathbf{P}|\leq d} Score(X_i;\mathbf{P}) \tag{2}$$

∎

The utility of a regulator set $\mathbf{R}$ can be computed quickly and closely approximates the score of the optimal network constrained to $\mathbf{R}$, denoted SCORE($\mathbf{R}$). $F(\mathbf{R})$ scores a graph structure resulting from independently choosing the optimal parents for each variable, and is therefore an upper bound on SCORE($\mathbf{R}$). Note, that an independent choice of parents for each variable may lead to a structure containing cycles. Thus, a legal Bayesian network might have a some parent sets that score sub-optimally. However, since cycles can form only between the variables in $\mathbf{R}$, which is a relatively small part of the entire network ($k << n$), we expect that SCORE($\mathbf{R}$) is usually only slightly less than $F(\mathbf{R})$. Furthermore, acyclicity can be resolved within a subgraph involving only k nodes (the complexity of solving this problem is constant in n, though exponential in k). In most practical cases, only a few short cycles form and the optimal solution can be easily found. In comparison, resolving cyclicity in a Bayesian network can be exponential in n ($k << n$), since cycles can form among any n variables.[2]

We treat $F(\mathbf{R})$ as a scoring function that measures the quality of regulator sets. This implies a new optimization problem to find a small set of regulators, $\mathcal{R}$, which maximize this score:

Definition 3 *The* Best Regulator Set *problem: Given a set of variables $\mathcal{X}$, a data set of samples D, a set of candidate parents $\mathcal{C}$, and the constants d and k, we wish to find*

$$\mathcal{R} = \text{argmax}_{\mathbf{R}\subset\mathcal{C},|\mathbf{R}|\leq k} F(\mathbf{R}) \tag{3}$$

∎

1. We typically use d ranging between 3 to 5 and k ranging between 30 to 70, whereas n is in the thousands and $|\mathcal{C}|$ is in the hundreds.
2. In our typical setting where k ranges between 30 to 70 and $n \geq 2000$, the difference in score after breaking the cycles is negligible. With high probability this difference would not change our choice of regulating set $\mathcal{R}$. Therefore, for the remainder of this paper, we ignore the issue of cyclicity.

This problem is conceptually similar to the *Set Cover* problem, a classical hard problem. The challenge is that the regulator set $\mathcal{R}$ must be chosen from a much larger candidate set $\mathcal{C}$ and there are $\binom{|\mathcal{C}|}{k}$ possible regulator sets. While there does not seem to be any efficient algorithm to find an optimal solution, we next present an efficient algorithm that attempts to approximate it.

3. MinReg Learning Algorithm

We now turn to the task of learning a regulation model (specified by a set of regulators $\mathcal{R}$, and a parent structure on the variables in $\mathcal{X}$) from a training set ($D = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$, consisting of M instances drawn independently from an unknown distribution $P(\mathcal{X})$). Our goal is to choose the set of regulators $\mathcal{R}$ and learn the regulatory graph structure that best explains this distribution. We take a *score-based approach* to this learning task and we define a scoring function that measures how well each candidate model fits the observed data.

Given a scoring function, our task is to devise a search algorithm capable of efficiently finding a high scoring model. As discussed in Section 2.1, the hard part of the search is to find the optimal set of regulators, $\mathcal{R}$. Our novel greedy algorithm for this task, MinReg (sketched in Figure 4), begins with an empty set of regulators and an empty graph structure. At each iteration, for each possible candidate, we construct an increment regulator set by adding that candidate to the current regulator set. We calculate the score for each of the increment regulator sets and choose the one that gives the largest gain. Each time $\mathcal{R}$ is updated, we calculate the optimal regulation graph restricted to the current regulator set $\mathcal{R}$. We continue to iterate until some stopping criterion is reached.

A crucial point is to correctly define the gain of a given regulator at each iteration. We calculate a local score between a variable and its regulating *set*. When considering a new candidate regulator $c \in \mathcal{C}$ as a parent for a variable X, we measure not how well c scores for X, but how much additional gain c gives to X's local score. Thus, each of the regulating parents provide a distinct contribution to the score.

Definition 4 *We define the* marginal utility *of adding a regulator set* $\mathbf{C}$ *to an already chosen regulator set* $\mathbf{R}$ *as*

$$F(\mathbf{C} \mid \mathbf{R}) = F(\mathbf{C} \cup \mathbf{R}) - F(\mathbf{R}) \tag{4}$$

∎

Thus, at each iteration, we add the candidate regulator with the largest marginal utility.

3.1 α-modularity and Performance Guarantees

MinReg is a greedy algorithm, which at each iteration, adds to the model the best single regulator according to some local criterion This greedy approach does not necessarily lead to a global optimum. Can we characterize the situations in which the MinReg algorithm is lead astray? Consider the case where $Score(X;A) + Score(X;B)$ is significantly less than $Score(X;A \cup B)$. In this situation, neither A nor B would be attractive enough to get selected by themselves in any of the greedy steps, whereas their joint contribution may be significantly higher than any other combination of regulators. Thus, the greedy algorithm is misled to choose an inferior regulator set. Importantly, this is a biologically-plausible scenario, since synergy between regulators is a well-documented phenomenon.

MinReg Algorithm
Begin with an empty regulator set $\mathcal{R}$ and an empty graph
In each iteration find the candidate regulator with the highest marginal utility:
$c^* = \text{argmax}_{c \in \mathcal{C}} F(c \mid \mathcal{R})$
  Iterate over each candidate regulator $c \in \mathcal{C}$
    calculate $F(c \mid \mathcal{R})$
    Iterate over each variable $X = X_1, \ldots, X_n$
      approximate its best parents restricted to $\mathcal{R} \cup c$
      $\max_{\mathbf{P} \subset \mathcal{R} \cup c, |\mathbf{P}| \leq d} Score(X_i; \mathbf{P})$
      Add local score of X_i to utility of c
  Add best candidate regulator to $\mathcal{R}$ and update the regulation graph
until *stopping criterion* (3.3.1)

Figure 4: Overview of the MinReg algorithm. The algorithm consists of two nested greedy loops. The external loop finds the optimal set $\mathcal{R}$ of k regulators. For each $X \in \mathcal{X}$ an internal loop finds an optimal set of parents $\mathbf{Pa}_X$.

We argue that this characterizes the only situation in which our algorithm fails. We show that if we can bound the severity of such effects, we can derive a worst case error bound on the algorithm's performance: in this case, MinReg is an *approximation algorithm*, guaranteed to find a solution which is not too far from optimal. To formally prove this guarantee, we introduce the notation of α-*modular* functions.

Definition 5 *Let f be a function defined over subsets of $\mathcal{C}$. f is monotone increasing if forall subsets $\mathbf{A}, \mathbf{B}$ s.t. $\mathbf{A} \subseteq \mathbf{B}$, the following holds*

$$f(\mathbf{A}) \leq f(\mathbf{B})$$

∎

Definition 6 *(Lehmann et al., 2001) Let f be a function defined over subsets of $\mathcal{C}$. f is α-modular ($\alpha \geq 1$) if and only if for all subsets $\mathbf{A}, \mathbf{R}$ and for all singletons Z, the following holds:*

$$f(\mathbf{A} \cup Z|\mathbf{R}) \leq f(\mathbf{A}|\mathbf{R}) + \alpha f(Z|\mathbf{R})$$

∎

Note that this is a generalization of sub-modular functions, f is sub-modular for $\alpha = 1$. One might consider α as some measure on the convexity of f over the space of subsets from $\mathcal{C}$. For larger α, more "synergy" can be gained by joining sets together. We will show that if we can bound the amount of "synergy" between regulators, we can bound the error of our greedy algorithm accordingly.

Lemma 7 *The following are equivalent definitions of* α*-modular functions: ((Lehmann et al., 2001))*

1. *For any subsets* $\mathbf{S} \subseteq \mathbf{T}$ *and singleton* $Z \notin \mathbf{T}$ *we have* $f(Z|\mathbf{S}) \geq \alpha f(Z|\mathbf{T})$.
2. *For any subsets* $\mathbf{S} \subseteq \mathbf{T}$ *and subset* $\mathbf{V}$ *we have* $f(\mathbf{V}|\mathbf{S}) \geq \alpha f(\mathbf{V}|\mathbf{T})$.
3. *For any subsets* $\mathbf{A}, \mathbf{B}$*, we have* $f(\mathbf{A}) + \alpha f(\mathbf{B}) \geq f(\mathbf{A} \cup \mathbf{B}) + \alpha f(\mathbf{A} \cap \mathbf{B})$

These equivalent formulations offer us another perspective: the marginal utilities of α-modular functions are "almost" (up to a factor of α) monotone decreasing. This fits our intuition that as $\mathcal{R}$ grows larger, the utility of adding new regulators diminishes.

Theorem 3.1: If F is an α-modular and monotone increasing function,[3] then the MinReg algorithm (presented in Figure 4) is a polynomial time approximation algorithm for the Best Regulator Set: Denote by OPT_k the optimal k regulator set (i.e. that maximizes F) and by MINREG_k the regulator set found by the MinReg algorithm, then

$$(\alpha+1)F(\text{MINREG}_k) \geq F(\text{OPT}_k) \tag{5}$$

This theorem provides assurance that while MinReg is a very quick algorithm that greedily takes locally optimal steps, the score of the regulator set found by MinReg is not too far away (at most a factor of $1+\alpha$) from the optimal solution reached by exhaustively enumerating all possible regulator sets.

Proof: Our proof is by induction. For $k = 1$, the optimal solution is the best single regulator and this is exactly the regulator found by the MinReg algorithm therefore $\text{MINREG}_1 = \text{OPT}_1$. We assume that $(\alpha+1)F(\text{MINREG}_{k-1}) \geq F(\text{OPT}_{k-1})$ and prove it for k.

Set $J = \text{argmax}_{I \in \mathcal{C}} F(I)$, the best single regulator in $\mathcal{C}$ and $\hat{J} = \text{argmax}_{I \in \text{OPT}_k} F(I)$, the best single regulator in OPT_k. Note, J is the first regulator chosen by the MinReg algorithm.

We define the following sub-problem imitating the behavior of the greedy algorithm. Let $\hat{F}(\mathbf{Y}) = F(\mathbf{Y} \cup \{J\}) - F(J)$, our goal is to find a set of $k-1$ regulators that optimize $\hat{F}$ on $\mathcal{C} \setminus \{J\}$. This is exactly what MinReg does after it chooses J in the first iteration. We denote by $\hat{\text{OPT}}_{k-1}$ and $\hat{\text{MINREG}}_{k-1}$ the optimal and greedy solutions respectively to this new sub problem. It is easy to see that $\hat{F}$ is a α-modular function and that our induction holds for $\hat{F}$ as well, that is, $(\alpha+1)\hat{F}(\hat{\text{MINREG}}_{k-1}) \geq \hat{F}(\hat{\text{OPT}}_{k-1})$.

By the inductive hypothesis, it suffices to show that the increment is α-modular, i.e.:

$$\begin{aligned} F(\text{OPT}_k) - \hat{F}(\hat{\text{OPT}}_{k-1}) &\leq (\alpha+1)F(\text{MINREG}_k) - \hat{F}(\hat{\text{OPT}}_{k-1}) \Rightarrow \\ F(\text{OPT}_k) &\leq (\alpha+1)F(\text{MINREG}_k) \end{aligned}$$

simply by subtraction of the same value on both sides. By the induction hypothesis on $\hat{F}$ we have that

$$\begin{aligned} F(\text{OPT}_k) - \hat{F}(\hat{\text{OPT}}_{k-1}) &\leq (\alpha+1)(F(\text{MINREG}_k) - \hat{F}(\hat{\text{MINREG}}_{k-1})) \\ &\leq (\alpha+1)F(\text{MINREG}_k) - \hat{F}(\hat{\text{OPT}}_{k-1}) \end{aligned}$$

3. While the marginal utilities should be almost monotone decreasing, we want the function itself to be monotone increasing.

because $(\alpha+1)\hat{F}(\widehat{\text{MINREG}}_{k-1}) \geq \hat{F}(\widehat{\text{OPT}}_{k-1})$. Note that

$$F(\text{MINREG}_k) - \hat{F}(\widehat{\text{MINREG}}_{k-1}) =$$
$$F(\text{MINREG}_k) - F(\widehat{\text{MINREG}}_{k-1} \cup \{J\}) + F(J) = F(J)$$

since $\widehat{\text{MINREG}}_{k-1} \cup \{J\} = \text{MINREG}_k$ by the very way in which the MinReg algorithm works. Thus to prove the induction for k it is enough to show that:

$$F(\text{OPT}_k) - \hat{F}(\widehat{\text{OPT}}_{k-1}) \leq (\alpha+1)F(J) \tag{6}$$

Since $\widehat{\text{OPT}}_{k-1}$ is at least as good as any solution of size $k-1$, by definition:

$$\hat{F}(\widehat{\text{OPT}}_{k-1}) \geq \hat{F}(\text{OPT}_k \setminus \{\hat{J}\}) = F(\text{OPT}_k \setminus \{J'\} \cup \{J\}) - F(J) \tag{7}$$

By α-modularity of F:

$$F(\text{OPT}_k) \leq F(\text{OPT}_k \setminus \{\hat{J}\}) + \alpha F(\hat{J}) \tag{8}$$

Subtracting (7) from (8) gives:

$$F(\text{OPT}_k) - \hat{F}(\widehat{\text{OPT}}_{k-1}) \leq [F(\text{OPT}_k \setminus \{\hat{J}\}) - F(\text{OPT}_k \setminus \{\hat{J}\} \cup \{J\})] + [\alpha F(\hat{J}) + F(J)] \tag{9}$$

Monotonicity of F implies that $F(\text{OPT}_k \setminus \{\hat{J}\}) \leq F(\text{OPT}_k \setminus \{\hat{J}\} \cup \{J\})$, therefore, the first bracket gives a negative contribution. Maximality of J implies that $F(J) \geq F(\hat{J})$, therefore the second bracket is $\leq (\alpha+1)F(J)$. ∎

It remains to show that F is both monotone and α-modular. Recall, F is a sum of maximizations of local scoring functions: $F = \sum_{i=1}^{n} \max_{\mathbf{P} \subset \mathbf{R}, |\mathbf{P}| \leq d} \mathit{Score}(X_i; \mathbf{P})$. The local maximizations are clearly monotone, if $\mathbf{S} \subset \mathbf{T}$, then $\forall X, \max_{\mathbf{P} \subset \mathbf{T}} \mathit{Score}(X; \mathbf{P}) \geq \max_{\mathbf{P} \subset \mathbf{S}} \mathit{Score}(X; \mathbf{P})$. Thus F, being the sum of monotone functions, is monotone as well.

Empirically we observe that F is α-modular, usually for relatively small α (see 4.1.1). At first this might sound surprising, since as mentioned above, synergy is known to play an important role in biological regulation, and we do not expect $\mathit{Score}(X; \mathbf{P})$ to be α-modular in the gene expression domain. Fortunately, while regulators are synergistic for specific targets, F is a sum over thousands of variables. Even if the synergy between two regulators is very high, this synergy would need to hold for many targets, otherwise it would average out when summing over all of $\mathcal{X}$. We empirically tested the synergy between regulators and groups of regulators in both yeast and mammalian data sets, the worst factor we encountered was 1.2. Therefore, we make the assumption of α-modularity of F with $\alpha = 2$ in the gene expression domain.

3.2 Scoring Function

To define the local score $\mathit{Score}(X; \mathbf{P})$, we adopt the Bayesian paradigm and use the Bayesian BDe scoring function (Heckerman et al., 1994; Heckerman, 1998) commonly used for learning Bayesian networks. The Bayesian score evaluates the posterior probability of the graph given the data:

$$\text{score}_{\mathcal{B}}(\mathcal{G} : \mathcal{D}) \quad = \quad \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

where $P(\mathcal{D} \mid \mathcal{G})$ takes into consideration our uncertainty over the parameters and averages the probability of the data over all possible parameter assignments to $\mathcal{G}$.

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \theta) P(\theta \mid \mathcal{G}) d\theta$$

The particular choice of the priors $P(\mathcal{G})$ and $P(\theta \mid \mathcal{G})$ determines the exact Bayesian score.

The BDe score refers to a certain class of priors with several desirable properties (as detailed in (Heckerman, 1998))In particular, the BDe score of an entire regulation graph $\mathcal{G}$, decomposes into sum over the local scores for each variable.

$$\text{score}(\mathcal{G} : D) = \sum_i Score(X_i; \mathbf{Pa}_{X_i}) \tag{10}$$

We use these local decomposed scores as the local score for our algorithm.

3.3 MinReg Implementation

A general overview of the MinReg algorithm was presented in Section 3. Several details in MinReg's implementation lead to substantial speed-up of the naïve algorithm, such that our implementation can generate a model over thousands of genes within few minutes.

First, we define a function f_X for each variable X, $f_X(\mathcal{R}) = \max_{\mathbf{P} \subset \mathcal{R}, |\mathbf{P}| \leq d} Score(X; \mathbf{P})$. This is the optimal contribution of X to F, restricted to a regulator set $\mathcal{R}$. Thus, we have 3 levels of scoring functions: *Score* - for a particular variable and its parents, f_X - the optimal score of a single variable X, and $F(\mathcal{R}) = \sum_{X \in \mathcal{X}} f_X(\mathcal{R})$.

The naïve greedy algorithm has k iterations. In each iteration, $F(c|\mathcal{R})$ is calculated for all $c \in \mathcal{C}$. Since each calculation of $F(c|\mathcal{R})$ requires calculating $f_X(c|\mathcal{R})$ for all $X \in \mathcal{X}$, f_X is calculated $k|\mathcal{C}||\mathcal{X}|$ times. Calculation of f_X requires calculating *Score* for each of the $\binom{k}{d}$ possible sets of parents. While this is constant in n, in practice k^d could be very large. We devise a number of heuristics based on α-modularity to reduce the number of times we need to calculate each of the 3 functions F, f_X, and *Score*.

We employ a branch and bound approach to $F(c|\mathcal{R})$, using the α-modularity of F to filter out candidates with little potential. In the first iteration, for all $c \in \mathcal{C}$ we calculate $\text{Util}(c) = F(c)$. We store the candidate regulators in a heap sorted by $\text{Util}(c)$. At any given time, $\text{Util}(c) = F(c|\mathbf{A})$, for some $\mathbf{A} \subseteq \mathcal{R}$. The α-modularity of F ensures that $\alpha\text{Util}(c) \geq F(c|\mathcal{R})$ (see Lemma 7). In most cases, we expect the regulator with the highest marginal utility to be toward the top of the heap.

Once a new regulator is added to $\mathcal{R}$, the marginal utilities change and need to be recalculated. In each subsequent iteration, we traverse down the heap and only re-evaluate candidates for whom $\alpha * \text{Util}(c)$ is greater than the best marginal valuation found thus far, denoted c^*. Each time $F(c|\mathcal{R})$ is calculated, we use this value to update $\text{Util}(c)$ in the heap. Once we reach a candidate such that $\alpha * \text{Util}(c) < F(c^*|\mathcal{R})$ we stop traversing the heap, as α-modularity ensures that none of the candidates beyond this point can be better than c^*. While this branch and bound does not change the worst case complexity, for most practical cases, only the few topmost candidates are examined in each iteration.

While the previous speed-up came at no loss in the quality of the final solution, the next two heuristics reduce accuracy. These heuristics are based on the assumption that *Score* is α-modular in most cases (though probably by a larger factor). This assumption is reasonable (albeit not always accurate). While regulation is sometimes synergistic, functions such as XOR are rare in biology and even when synergy exists, it is bounded by a reasonable constant. More importantly, we expect the synergistic pairs are themselves uncommon.

Similarly to how $\text{Util}(c)$ approximates $F(c|\mathcal{R})$, we cache $\text{Util}_X(c)$ as an approximation of $f_X(c|\mathcal{R})$. Whenever $F(c|\mathcal{R})$ is calculated, we do so only approximately: $F(c|\mathcal{R}) = \sum_{X \in \mathcal{X}} \text{Util}_X(c)$.

In the first iteration we initialize $\text{Util}_X(c) = f_X(c)$, for each $c \in \mathcal{C}$ and $X \in \mathcal{X}$. In subsequent iterations, we only recalculate $f_X(c|\mathcal{R})$ (and update $\text{Util}_X(c)$) for those X's whose parent set $\mathbf{Pa}_X$ changed in the previous iteration. This is especially effective in later iterations where $\mathbf{Pa}_X$ rarely changes.

Finally, instead of calculating f_X exactly, we approximate it using a greedy algorithm similar to Figure 4. We start with no parents and at each iteration add best parent, $argmax_{c\in\mathcal{R}} f_X(c|\mathbf{Pa}_X)$ to $\mathbf{Pa}_X$. This only requires $d|\mathcal{R}|$ calculations of *Score* instead of $\binom{|\mathcal{R}|}{d}$.

3.3.1 STOPPING CRITERION

Formally, the best regulator set problem requires a predefined constant k, specifying the number of regulators in the model. However, there are no obvious biological grounds for choosing a particular "good" k, as there is a trade-off between fine resolution (offered by a larger number of regulators) and statistical robustness (from a small number of regulators).

We address this fine balance by taking an adaptive approach. We devise a *stopping criterion* for the addition of new regulators to our model. We continue to add regulators as long as their contribution to the score is significantly better than random regulators. We generate a set of m random regulators with similar properties to the real candidate regulators in our data. We construct these by sampling regulators with replacement from the original candidate regulator set. For each sampled regulator, we randomly permute the order of its samples. Thus, the random regulators have the same distribution over their values, but these are independent of the target variables. We calculate the score for these random regulators in a manner similar to the real candidate set and store these in a heap. This provides us with an empirical distribution for the score of a random regulator.

We continue to add regulators to our model as long as they score greater than the random candidates. We update the scores in the candidate heap in a similar manner to the real candidates, pruning the heap using α-modularity. We stop once a random regulator scores better than any real regulator.

4. Experimental Results

We evaluated our algorithm on two data sets, a compendium of yeast expression profiles and a data set of mouse B-lymphocyte expression profiles. The distinct two data sets provide us each with a different realistic evaluation context. The yeast *S. cerevisiae* is the most extensively studied organism on a genomic scale, and has the most extensively characterized regulatory system among eukaryotes. In addition to an extensive amount of microarray data, identifying cis-regulatory elements (Bussemaker et al., 2001) and TF binding events (ChIP-chip experiments) (Lee et al., 2002; Harbison et al., 2004) is tractable on a genomics scale. Finally, decades of careful studies on individual gene functions are documented in a genome database (Cherry et al., 2001). Together, these data sources will allow us to carefully evaluate the success of our method in light of current biological knowledge.

Mammalian regulatory systems, such as those of the laboratory mouse, are notoriously difficult to elucidate, both experimentally and computationally. First, these networks are significantly more complex, involving a larger number of regulators, binding to long promoter and enhancer sequences. In particular, different cell types and cell states employ different regulatory networks to process signals. Furthermore, this complexity renders both genomics studies of regulatory events (such as ChIP-chip experiments) and computational ones (such as discovery of *cis*-elements) dif-

ficult or intractable. In fact, the sole available source of relevant data in mammalian systems is typically microarray measurements of expression profiles. Importantly, no successful method was demonstrated to date for reconstructing regulatory networks from mammalian expression profiles. Even partial success of MinReg in reconstructing mammalian regulation would constitute a significant scientific advance.

The yeast data set contained 358 samples combined from the Compendium (Hughes et al., 2000) and stress (Gasch et al., 2000) data sets.[4] We compiled a set $\mathcal{C}$ of 466 candidate regulators for yeast, which includes any gene with a potential regulatory role based on annotation or sequence homology. The expression profiles were discretized into 3 values: *down-regulated*, *no change* and *up-regulated*.[5] We included only 3755 genes with significant change in gene expression in at least 15 samples. We set the maximal indegree, $d = 3$, a reasonable estimate for the regulation of most yeast genes (in particular under a limited number of condition).[6] We conservatively set $\alpha = 2$ based on empirical evaluation of the data (see Section 4.1.1 below). We applied our MinReg algorithm to this data set resulting in a yeast regulation model with 44 key regulators.

The mouse data set consisted of 204 samples from purified spleenic B-lymphocytes (Sambrano et al., 2002), subjected to a number of stimuli (ligands) and combinations of these stimuli. We compiled a list of 684 candidate regulators using criterion similar to the yeast candidate regulator set.[7] We discretized the data as in yeast, except that the data was discretized into 5 levels: strongly down regulated, weakly down regulated, no change, weakly upregulated and strongly upregulated. We included only the 4373 genes that significantly changed in at least 18 samples. We ran the MinReg algorithm on this data and inferred a regulation model with 75 key regulators.

We employed both statistical and biological criteria to evaluate the performance of the algorithm. We examine our assumptions of α-modularity, the ability of our algorithm to generalize to unseen data, and the accuracy of the reconstruction on synthetic data. To demonstrate the accuracy of our algorithm in reconstructing the real yeast and mammalian regulatory network, we devise an approach to infer regulator function from our model, and compare that to the known central regulators in the relevant biological processes.

4.1 Statistical Evaluation

In this section we well evaluate the statististical robustness of the MinReg algorithm. We focus on two issues, is the assumption of alpha-modularity a reasonable one for our gene expression domain, and how well does our learned model generalize to unseen test data.

4. The Compendium (Hughes et al., 2000) contains 276 deletion mutants from various functional classes and the Stress data set (Gasch et al., 2000) contains 82 samples of responses to 12 different stress conditions.
5. The Bayesian score is based on a multinomial distribution. Exact continuous measurement is a very noisy estimate of the actual gene expression and in our experience, discrete states better represent gene activity. We used a soft discretization based on a linear piecewise step function for each level of activity.
6. While some genes might have more regulators, there is not enough data to learn such complex regulatory function from so few samples. Importantly, our goal is to robustly learn the key regulatory relations, not the full detailed network.
7. Mouse has many more known regulators, but only 1/3 the mouse genome was printed on the microarray and only these genes were included in the analysis.

4.1.1 ALPHA MODULARITY

MinReg employs a greedy approach, evaluating only the addition of single regulators to the model at each iteration, potentially missing a better combination of regulators.[8] Based on the assumption of α-modularity of F, Theorem 3.1 ensures that the score of the greedy solution is not much worse than the score of the optimal solution. Furthermore, to improve the speed performance, the α-modularity of F is used strongly by the implementation to bound the number of the regulators that are evaluated from the heap at each iteration.[9]

To empirically evaluate the α-modularity of F in the two data sets used, we calculated the pairwise gain in score for all pairs of regulators in the candidate set $\mathcal{C}$. Thus, for each pair of candidates c_1 and c_2, we calculated the worst α using $F(c_1 \cup c_2)$, $F(c_1)$ and $F(c_2)$. In addition, we calculated the worst α for 10,000 pairs of random subsets, $\mathbf{C}_1, \mathbf{C}_2 \subset \mathcal{C}$, ranging between 2 to 8 regulators each, using $F(\mathbf{C}_1 \cup \mathbf{C}_2)$,$F(\mathbf{C}_1)$ and $F(\mathbf{C}_2)$. For the yeast and B-lymphocyte data, the worst α empirically encountered were 1.184 and 1.229, respectively. We used $\alpha = 2$ as a conservative overestimation to determine when to stop evaluating candidates in the heap at each iteration.

To further boost speed, we make a weak (but inaccurate) assumption that *Score* is close to α-modular, allowing MinReg to reconstruct a large network over thousands of genes in a few minutes, rather than overnight. We evaluated the effect of this additional modification on MinReg's performance by comparing its affect on the likelihood of test data in cross validation, as well on the enrichment of GO annotations in target sets (see sections 4.1.2 and 4.2.1) Indeed, based on these two criteria, assuming α-modularity of *Score* does not hurt the algorithm's performance.

4.1.2 CROSS VALIDATION

To evaluate the statistical robustness of our learned model and its ability to generalize to unseen data, we tested MinReg's performance in 5-fold cross validation. We randomly split the data into 5 equal parts, and ran MinReg 5 times. Each time using 4/5 of the samples as training samples to to learn both the structure and parameters of the regulation model, and withholding 1/5th of the data samples as a testing set. We then used the inferred model and gene expression of inferred regulators in the test data to predict the expression levels of all 3755 variables in each test sample. That is, given the expression of regulators in the mth sample, we use $P(X \mid \mathbf{Pa}_X[m])$ to predict the value of X in that sample.

We compared the likelihood of test data in several different models. As a baseline, we used the marginal probability of each variable to predict its value. Since most of the variables had a high frequency of the value 0 (their corresponding gene's expression remained unchanged most of the time), even this simple predictor scored well (Figure 5, crosses). As competition to our MinReg algorithm, we generated 44 clusters using standard k-means clustering (Duda and Hart, 1973; Tavazoie et al., 1999) and randomly chose from within each cluster a gene $r \in \mathcal{C}$ as its "regulator". For each cluster we used $P(X \mid r)$ as our predictor. While cluster representatives somewhat improved the prediction over the baseline (0.06 log-loss/instance, Figure 5, circles), our MinReg algorithm clearly provided the best predictions (0.11 log-loss/instance, Figure 5, triangles). In conclusion, our cross-validation demonstrates that the model generated by the MinReg algorithm performs well on

8. This assumption is made implicitly by the classic and widely used greedy Bayesian network learning algorithm (Heckerman, 1998), that considers greedy moves of adding, removing and reversing a single edge at each step.
9. Typically, after the first few iterations, only a few regulators are evaluated at each iteration.

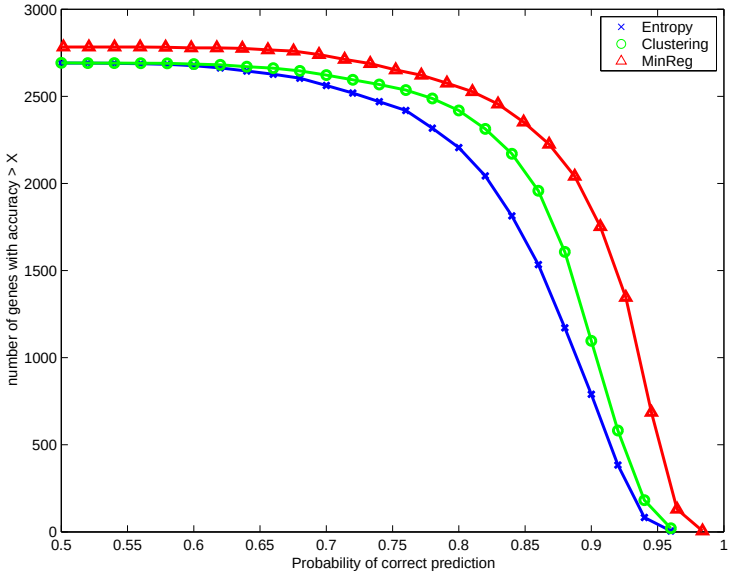


Figure 5: Cross validation of the predictive capabilities of our model on test data. The graph measures the number of variables correctly predicted at each probability. We compare our model (triangles) to the null model (crosses) that uses the marginal distribution of each variable and to a model based on cluster representatives (circles)

test data and that most of the information in an entire microarray can be captured by a small set of key regulators.

4.1.3 SYNTHETIC DATA

To evaluate the accuracy of the MinReg algorithm in a controlled setting, we generated synthetic data from a known regulation network. This gives a known ground truth to which we can compare the learned models. To make the data realistic, we generated synthetic data from the regulation model inferred from the yeast gene expression data above. While the inferred network is less complex than a true biological network, both share the same underlying probability distribution of the discretized data. We randomly sampled 10 data sets from this regulation model, each set consisting of 358 samples (same number of samples as the original data set). We tested MinReg's ability to reconstruct the correct network independently on each of these 10 synthetic data sets.

Our first test evaluated MinReg's choice of regulators. On average (over 10 repeats), MinReg correctly reconstructed 84% (39) of the generating 44 regulators. The worst case was 80% (35) and best case 91% (40). As for false positives, on average 74% of the reconstructed regulators were correct (worst case 71% and best case 77%).

Next we evaluated the detailed model itself. The generating model contained 6616 edges and we checked how many of those were correctly recovered. On average 70% of these were recovered (worst case 69% and best case 72%). In each model the percent of correct edges was a bit higher, with an average of 74% (worst case 72% and best case 77%). Even when we did not limit to

candidate regulators (setting $\mathcal{C} = \mathcal{X}$) our reconstruction of regulators was surprisingly good, 42/47 of the regulators were correct and 76% of the individual edges were correctly reconstructed.

In summary, using only a small number of samples, MinReg is capable of learning a model over thousands of variables, reconstructing most of the relationships correctly.

4.2 Biological Evaluation

The crucial test for the success of our approach is in reconstructing the main aspects of a real regulatory program. The true underlying biology is vastly more complex than the simple regulation model that generated the synthetic data. In a real biological system, there are more regulators working together in more complex functions, feedback loops, and unobserved events. Furthermore, the expression data probing these is noisy. Unfortunately, since our knowledge on the principles and specifics of real regulatory networks is limited, so is our ability to test our model based on "realistic" simulated data. Due to this lack of biological knowledge we also do not have a gold standard network in any organism.

Fortunately, while evaluating each specific connection is impossible at the moment, we can estimate whether our model has correctly captured the overall biological regulatory events in the system. To do this, we rely on the functional annotations available for many genes, which describe (using the controlled vocabulary of the Gene Ontology (Consortium, 2000)) the molecular function, biological process and cellular location of individual genes. Thus, as described below, we will evaluate our reconstructed model by the ability to use it to correctly deduce the functional annotation of regulators, and by the fit of these annotations to the relevant biological system. Importantly, such functional characterization of key regulators is a critical biological task in its own right.

4.2.1 ANNOTATING REGULATORS

Our approach is based on the understanding that the biological function of a regulator is mediated by its set of targets. Therefore, the common shared function of its set of targets (*e.g.* enzymes involved in amino acid (AA) metabolism) characterizes the overall biological process it regulates (*e.g.* amino acid metabolism). Continuing with the AA metabolism example, if our reconstructed model is good, we expect a regulator of AA metabolism to have many inferred targets involved in AA metabolism. More generally, we expect that the function associated with a regulator based on its set of targets in the model (as reflected by significant enrichment for a particular annotation) will fit with the known function of this regulator (as reflected by its own annotation).

More formally, we denote by $\mathcal{X}_r$ the set of targets of a regulator r in our network structure. For each annotation term A, we calculate the fraction of genes in $\mathcal{X}_r$ associated with A and use the hyper-geometric distribution to calculate a p-value for this fraction. We report for each regulator, all the significant annotation terms with which it was associated and compare them to the known annotations for that regulator, and to the main functions expected in the biological systems we examine.

4.2.2 EVALUATING THE YEAST REGULATORY NETWORK

Based on this test, our reconstructed yeast network corresponds well to previous findings. Specifically, the model derived functions for 8 of the top 10 regulators (sorted by p-value) coincided with their known biological roles. Of the remaining two regulators, we were able to assign a putative role

to one previously uncharacterized gene, but failed to identify the correct role of the other.[10] Furthermore, we examined numerous individual edges underlying these derived associations demonstrating that they are indeed supported by previously described regulator-target relations, lending support to our global analysis.

Additionally, the sequence motif representing binding site preference, is known for many yeast transcription factors. As an additional source of validation for MinReg's target sets, we use a putative map that uses these motif models to predict the gene targets of these transcription factors (Harbison et al., 2004). Similarlly to testing enrichment for GO annotations, we tested for enrichment of motifs in the promoter regions in each target set. In the case of signaling proteins, we tested for enrichment of the known transcription factor target activated by the signaling protein (see figure 1). A correct match between regulator and motif was found for 6 of the top 8 regulators,[11] Sst2 (Ste12), Met28, Uga3, Slt2 (Rlm1), Tpk2 (Msn2/4), and Tec1. Together the correct functional annotation and the occurance of the known motif in the regulatory regions, strongly support the quality of our reconstructed network. A more detailed and biologically oriented analysis of a simplified version of the proposed algorithm has been published in (Pe'er et al., 2002).

4.2.3 THE IMPORTANCE OF CANDIDATE REGULATORS

The pre-defined set of candidate regulators, C, is the only source of prior biological knowledge to our algorithm. In addition to focusing the model on regulatory relations, it narrows the search space and significantly reduces the running time of the algorithm (which is quadratic in the size of candidate set).

To assess the impact of this prior knowledge on MinReg's success, we examined MinReg's biological accuracy when run on the yeast data set, in the absence of a pre-defined candidate regulator set (*i.e.* $\mathcal{C} = \mathcal{X}$, such that any gene can be chosen as a regulator).[12] MinReg chose 35 regulators, only 6 of which were in the original candidate regulator set. This lack of "true" regulators suggests the expression of co-regulated genes is often at least as predictive (and sometimes even more) than that of the true regulating gene. While this model might be highly predictive and even generalizes well to new data, it does not reconstruct biological regulation, and is difficult to interpret.

Nevertheless, while $\frac{6}{35}$ is only a small fraction of the chosen regulators, this is a significant enrichment compared to their fraction in the candidate set (pvalue 0.003). The fact that our procedure results in a statistically significant enhancement of regulators is encouraging. We speculate that in complex organisms, where combinatorial regulation is expected to play a bigger role, this approach will be even more successful in detecting genes with a regulatory function.

4.3 Analysis of Mouse Data

In contrast to the significant success of several methods in reconstructing yeast regulatory networks, no algorithm has so far successfully reconstructed a mammalian regulatory network. To examine

10. The results for the top 20 regulators are of similar quality: the associations for 13 regulators correspond their known function,four regulators were previously uncharacterized and the associations for three regulators are unsupported.
11. These are 8/10 regulators we evaluated for gene function above, excluding 2 regulators for which no motif is currently known.
12. Since the algorithm is quadratic in $\mathcal{C}$, we reduced $\mathcal{X}$ by including only genes whose expression significantly changed in ≥ 18 samples (versus 15 samples). This resulted in a set of 2828 genes for both $\mathcal{X}$ and $\mathcal{C}$. It is important to note that in our data set, the expression of many candidate regulators remains almost constant. Only 148 genes from our original candidate set of known and putative regulators were included in the 2828 genes.

whether MinReg can scale up to the challenge of a mammalian system, we evaluated the biological accuracy of its reconstruction of a B-lymphocyte regulatory network.

We first examine whether the key regulators identified by the algorithm are known to participate in the main biological process taking place in B lymphocytes under the tested stimuli- the decision between cell proliferation and cell death. Indeed, the inferred regulator set $\mathcal{R}$ includes the top five genes - Trp53, Nfkb1, Jun, Fos and Bak1 - known to play a pivotal role in this decision. 16 additional inferred regulators are known to be directly involved in the regulation of proliferation and cell death[13] and seven others are involved in the regulation of the cell division cycle.[14] Overall, 28/75 regulators are known to participate in regulation of the central process occurring in these cells. Importantly, in multi-cellular organisms such as mouse, each cell type is characterized by a distinct regulatory network (although some of the sub-systems may be used in different types of cells). Indeed, 28 of the 75 inferred regulators are known to be involved in lymphocyte regulation: 7 genes (Nfkb1, Jun, Fos, Daxx, Syk, Gnai2, Csf1r) are known central regulators in B-lymphocytes, 15 genes are known to be active in the regulation of lymphocytes in general, and 6 others encode cytokines and their receptors (important in the regulation of immune cells, including lymphocytes). Taken together, this analysis indicates that 44 of the 75 inferred regulators are known regulators of lymphocytes, cell proliferation and death or both.[15] This suggests, that when applied to a complex mammalian data set, MinReg is able to identify the key regulatory genes active in this system. Finding such central regulatory genes is still a major biological task in most systems.

We next examined the quality of our model structure, based on its ability to predict the detailed function of individual regulators (as described above for yeast). For each regulator, we compared the 3 top significant annotation terms ($P < 0.05$) based on its predicted targets with its known annotation terms (typically 5-6 per regulator). We defined 5 different categories[16] and evaluated the significance of our results by comparing to the null model of randomly assigning each regulator with 3 GO annotations (out of 2694 annotations tested). Based on these criteria the predicted function for over half (45/75) the regulators had at least some support in prior biological knowledge. Specifically, the predicted functional annotation of 6/75 regulators was "very good" ($P < 10^{-18}$), "good" for 28/75 additional regulators ($P < 10^{-35}$), and "weak" for 11/75 genes. 12/75 genes had "no match" to any annotations, but many of them were genes encoding relatively uncharacterized regulators with little or no known annotations. Importantly, only 16 of 75 regulators were assigned no significant annotation, indicating the biological coherence of our reconstructed model, where regulators are associated with functionally related targets.

To illustrate the quality of our findings, we highlight several specific examples. First, we note that some of the "very good" annotations demonstrate that MinReg can provide an extensive biological characterization of the regulatory function of genes. For example, our model indicates that the protein Map3k1 functions in the MAP Kinase Signaling Pathway has a signal transducer activity and works in the Growth factor signaling pathway. Importantly, the model also identified several of Map3k1's targets, including Fos and Nfkb1. This is a highly accurate characterization of the

13. They are Aaft, Daxx, Foxo1, Gadd45g, Gnai2, Hipl2, Igbp1, Il2rh, Jund1, Itgb4, Map3k1, Rgs15, Rras2, Rsu1, Socs1, and Zmynd11.
14. They are Ax1, Camk2b, Csfir, Elk3, Maf, Tbl1, Rgs2, and Tbl1.
15. We expect that many of the other inferred regulators may be just as correct, and are simply not characterized by current biological knowledge. They suggest therefore novel biological hypotheses for experimental validation.
16. Our categories are: "Very good" (more than one exact match); "good" (1 exact match), "weak" (1 approximate match to a related term), "no match" (significant annotation were associated with the regulator but none match any known annotations) and "no p-value" (no significant annotations were associated with the regulator).

molecular function of this protein, and of its biological regulatory role. Since Map3k1 is a signaling protein (rather than a transcription factor) this is a particularly important achievement, since direct assays of regulatory function (based on cis-elements and protein-DNA binding) cannot help in this task. Indeed, our method detects and correctly associates a whole range of regulators - including transcription factors, kinases and phosphotases. For example, Aatf, the apoptosis antagonizing transcription factor, is correctly associated with the apoptotic pathway and the mitotic cell cycle. Dusp4, the dual specificity phosphatase 4, is correctly associated with protein-tyrosine-phosphatase activity and MAPK signaling pathway. Finally, we emphasize that the main benefit of our approach is in suggesting novel hypotheses for further research. Thus, "weak" and "no match" associations may present the most important biological leads emanating from MinReg's results. For example, the Jun oncogene (assigned to the "Weak" category), was predicted by our method to be involved in the Oncogene associated pathway and Cell proliferation and differentiation. While multiple abstracts in the published literature clearly and strongly support these two associations, Jun's current GO annotation includes neither.

4.3.1 COMPARISON TO MODULE NETWORKS

Does MinReg have significant benefits in reconstructing mammalian regulatory networks over other (related) reconstruction approaches? To address this question, we compared MinReg's performance on the B-lymphocyte data set to that of the Module Networks algorithm (Segal et al., 2003, 2005). Similar to MinReg, Module Networks associates a regulator to its targets based solely on dependencies in gene expression. However, while MinReg considers each target gene separately, Module Networks groups targets into sets ("modules"), such that all module genes share exactly the same regulatory program. In previous work (Segal et al., 2003), Module Networks was shown to be highly successful in reconstructing the yeast stress regulatory network. Here, we applied Module networks to the B-lymphocyte data and learn 75 modules and their associated regulation programs, involving 216 regulators overall. While many of the regulators overlapped those chosen by the MinReg algorithm,[17] these did not include 3 of the 5 known central regulators of cell proliferation and death (Nfkb1, Fos, nor Bak1) identified by MinReg.

For comparison, we can evaluate the Module Networks model by annotating each of its 206 regulators based on its associated targets (compiled across all 75 modules), resulting in 196 significantly annotated regulators. When evaluating the annotation quality of the top 75 regulators (sorted by p-value) by the same scale described above, we did not receive similarly significant results. In fact, only 13/75 genes had any support in prior biological knowledge (1/75 scored "very good", 7/75 scored "good", and 5/75 scored "weak"). Furthermore, when examining only regulators identified by both algorithms, MinReg's associations outperform Module networks on 23 regulators, while Module networks only found a better association for one regulator (Gnai13). Thus, in the specific task of characterizing the molecular function and biological process controlled by a regulator, MinReg overwhelmingly outperforms Module Networks in this mammalian data set. This suggests that the detailed network and regulatory targets identified by MinReg are more accurate than those discovered by Module Networks.

What may be the underlying reason for MinReg's success over Module Networks? The central goal of Module networks is to decompose the space of all genes into functionally coherent co-

17. Module networks learned on the real valued data, rather than the discretized expression values, further supporting the robustness of the overlapping set of regulators.

regulated modules, at the "cost" of constraining them to share exactly the same set of regulators. While this constraint increases the statistical robustness and biological coherence (leading to a major success on yeast data), it may be less suited to complex mammalian regulatory systems. In contrast, MinReg focuses on finding the most dominant regulators and their targets in the data. A regulator is only assigned to a target if that specific edge is sufficiently supported by the data [18] and each gene chooses its unique set of regulators. We believe these two reasons combined led to MinReg's superior performance in regulatory reconstruction.

5. Discussion and Conclusions

We have introduced the MinReg framework, a constrained Bayesian network for the reconstruction of regulatory networks. The framework limits the total number of parents in the model, thus focusing on only a small parsimonious set of key regulators. We exploit these constraints to devise a novel efficient approximation algorithm to search for a high scoring network from expression data. Under reasonable assumptions on the underlying probability distribution, we can prove guarantees on our algorithm's performance. To derive these guarantees, we introduce the notion of α-modularity, a convexity measure of the scoring function over the space of possible parent sets. Approximation algorithms with a performance guarantee rarely exist for Bayesian networks (Dasgupta, 1999) and we hope this measure can be used to derive addition performance bounds for other sub-classes of Bayesian networks.

Machine learning in the gene expression domain is especially challenging as it requires learning structures over thousands of variables using at most hundreds of samples. Our extensive experimental results on real expression data demonstrate that our framework is up to this challenge: we successfully infer regulatory relations over thousands of genes within minutes. Our results are validated by statistical criteria (synthetic data, cross-validation) and biological ones (our ability to correctly infer a correct set of key regulators and their detailed regulatory functions). Importantly, unlike previous approaches, our method scales well to complex mammalian systems, discovering key mammalian regulators (both signaling proteins and transcription factors) solely from expression data.

While constraining the number of regulators carries obvious statistical and computational advantages, what does it cost us in biological accuracy? We claim that the focus on a small and parsimonious regulatory set is as motivated biologically as it is statistically. Most importantly, any complex biological network involves a multitude of genes and proteins, but biologists' primary goal is most typically to find the central genes, that play the most important functional role in the system. In fact, a full and accurate model of the exact network at a given point, may fail to highlight those central genes. Rather, by focusing on a small set of key regulators, MinReg can provide clear critical leads for further research. Indeed, our analysis of the B lymphocyte data set indicates that MinReg is able to focus on the very key regulators of a complex process (cell proliferation and death) as well as on a significant number of cell specific regulators. Using an established "guilt-by-association" approach (Wu et al., 2002; Ihmels et al., 2002), we further capitalize on the learned structure, and identify the accurate functional roles of these proteins in regulating cellular processes. This is a major feat, never before accomplished by a computational algorithm for a mammalian system. Importantly, MinReg is not only superior to standard clustering, but it overwhelmingly outperforms in this task the recently published Module Network algorithm (Segal et al., 2005).

18. Many genes in the final model do not have any regulator, as none scored well enough.

Our method relies on the assumption that regulatory interactions between genes often result in corresponding statistical dependencies between random variables representing their expression. Recently, there have been a number of successful attempts to use other data sources - such as *cis*-elements (Bussemaker et al., 2001; Segal et al., 2002) and transcription factor binding events (Bar-Joseph et al., 2003) to infer regulatory relations in yeast. However, these successes cannot scale well to mammalian systems, in which computational detection of *cis*-elements is far less tractable (due to long and ill-defined promoters), and experimental detection of binding events is currently very limited (due to the genome size and the difficulty in carrying such experiments *in vivo*). In contrast, the collection of mammalian expression data is growing at an exponential rate, and methods such as MinReg that rely solely on gene expression for network reconstruction are direly needed.

MinReg lies between two graphical model based approaches for learning regulatory networks: unconstrained Bayesian networks and Module Networks Segal et al. (2005). While unconstrained Bayesian networks allow for a reconstruction of finer structure, they have only been successful at reconstructing small networks or subnetworks consisting of only a few variables Pe'er et al. (2001); Hartemink et al. (2002); Imoto et al. (2002). In contrast, MinReg and Module Networks can reconstruct a network over thousands of variables, based on the assumption that a small number of regulators can be chosen from a pre-defined candidate set. All three approaches, assume that regulator expression can be a proxy for its activity. Bioinformatics validation (of all approaches), and experimental validation (of Module Networks (Segal et al., 2003)) indicates that they can be at least partly successful in this task. This success is somewhat surprising, since actual protein activity depends on many biochemical events in addition to mRNA transcription.

What accounts for the significant success of MinReg compared to Module networks in mammalian network reconstruction? In the Module Network approach, genes are grouped into modules, thus losing their individual identity and distinction. MinReg provides a finer structure, allowing each gene an individual set of parents and regulatory function. Many recent biological papers stress the importance of modularity in biological networks (Hartwell et al., 1999; Ihmels et al., 2002; Segal et al., 2003; Bar-Joseph et al., 2003). Such organization facilitates orchestrating coordinated responses to external and internal signals by co-regulating genes that participate in a common function or task. While modularity may be a general organizing principle of regulatory networks, it may be too coarse grained on it own to represent the complex coordination between multiple genes and biological process. Rather, complex mammalian regulation is probably orchestrated by few key regulators, which combine together to regulate the genome, one target at a time through its unique regulatory program.

References

Z. Bar-Joseph, G. K. Gerber, T. I. Lee, N. J. Rinaldi, J. Y. Yoo, F. Robert, D. B. Gordon, E. Fraenkel, T. S. Jaakkola, R. A. Young, and D. K. Gifford. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21:1337–42, Nov 2003.

H. Bussemaker, H. Li, and E. Siggia. Regulatory element detection using correlation with expression. *Nature Genetics*, 27(2):167–171, 2001.

J. M. Cherry, C. Ball, K. Dolinski, S. Dwight, M. Harris, J. C. Matese, G. Sherlock, G. Binkley, H. Jin, S. Weng, and D. Botstein. *Saccharomyces* genome database. http://genome-www.stanford.edu/Saccharomyces/, 2001.

The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

S. Dasgupta. Learning polytrees. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*. 1999.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein, and P. O. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Mol. Bio. Cell*, 11:4241–4257, 2000.

C. T. Harbison, D. B. Gordon, T. I. Lee, N. J. Rinaldi, Macisaac K. D., T. W. Danford, N. M. Hannett, J. B. Tagne, D. B. Reynolds, J. Yoo, E. G. Jennings, J. Zeitlinger, D. K. Pokholok, M. Kellis, P. A. Rolfe, K. T. Takusagawa, E. S. Lander, D. K. Gifford, E. Fraenkel, and R. A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431:99–104, 2004.

A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Combining location and expression data for principled discovery of genetic regulatory networks. In *Pacific Symposium on Biocomputing*, pages 437–449, 2002.

L. H. Hartwell, J. J. Hopfield, S. Leibler, and Murray A. W. From molecular to modular cell biology. *Nature*, 2, Dec 1999.

D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 293–301. 1994.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–26, 2000.

J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31:370–7, Aug 2002.

S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using bayesian networks and nonparametric regression. In *Pacific Symposium on Biocomputing*, pages 185–186, 2002.

T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. Transcriptional regulatory networks in *saccharomyces cerevisiae*. *Science*, 298:799–804, 2002.

B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *ACM Conference on Electronic Commerce*, pages 18–28, 2001.

A. Martinez-Antonio and J. Collado-Vides. Identifying global regulators in transcriptional regulatory networks in bacteria. *Current Opinion Microbioly*, 6(5):482–9, 2003.

I. M. Ong, J. D. Glasner, and D. Page. Modelling regulatory pathways in *e. coli* from time series expression profiles. *Bioinformatics*, 18 Suppl 1:S241–S248, 2002.

D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 17 Suppl 1:S215–S224, 2001.

D. Pe'er, A. Regev, and A. Tanay. Minreg: Inferring an active regulator set. *Bioinformatics*, 18 Suppl 1:S258–S267, 2002.

G. R. Sambrano, G. Chandy, S. Choi, D. Decamp, R. Hsueh, K. M. Lin, D. Mock, N. O'Rourke, T. Roach, H. Shu, B. Sinkovits, M. Verghese, and H. Bourne. Unravelling the signal-transduction network in b lymphocytes. *Nature*, 420:708–710, Dec 2002.

E. Segal, Y. Barash, I. Simon, N. Friedman, and D. Koller. From promoter sequence to expression: A probabilistic framekwork. In *RECOMB*, pages 263–272. 2002.

E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. *Journal of Machine Learning Research*, 6:557–588, April 2005.

E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman. Module networks: identifying regulatory modules and their condition specific regulators from gene expression data. *Nature Genetics*, 34:166 – 176, 2003.

S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature Genetics*, 31(1):64–8, 2002.

S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nat Genet*, 22(3):281–5, 1999.

L. F. Wu, T. R. Hughes, A. P. Davierwala, M. D. Robinson, R. Stoughton, and S. J. Altschuler. Large-scale prediction of *saccharomyces cerevisiae* gene function using overlapping transcriptional clusters. *Nature Genetics*, 31:255–265, 2002.

C. Yoo, V. Thorsson, and G. F. Cooper. Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational dna microarray data. In *Pacific Symposium on Biocomputing*, pages 498–509, 2002.

Journal of Machine Learning Research 7 (2006) 191–246 Submitted 3/05; Revised 9/05; Published 2/06

Learning the Structure of Linear Latent Variable Models

Ricardo Silva[*] RBAS@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit
University College London
London, WC1N 3AR, UK

Richard Scheines SCHEINES@ANDREW.CMU.EDU
Clark Glymour CG09@ANDREW.CMU.EDU
Peter Spirtes PS7Z@ANDREW.CMU.EDU
Center for Automated Learning and Discovery (CALD) and Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: David Maxwell Chickering

Abstract

We describe anytime search procedures that (1) find disjoint subsets of recorded variables for which the members of each subset are d-separated by a single common unrecorded cause, if such exists; (2) return information about the causal relations among the latent factors so identified. We prove the procedure is point-wise consistent assuming (a) the causal relations can be represented by a directed acyclic graph (DAG) satisfying the Markov Assumption and the Faithfulness Assumption; (b) unrecorded variables are not caused by recorded variables; and (c) dependencies are linear. We compare the procedure with standard approaches over a variety of simulated structures and sample sizes, and illustrate its practical value with brief studies of social science data sets. Finally, we consider generalizations for non-linear systems.

Keywords: latent variable models, causality, graphical models

1. What We Will Show

In many empirical studies that estimate causal relationships, influential variables are unrecorded, or "latent." When unrecorded variables are believed to influence only one recorded variable directly, they are commonly modeled as noise. When, however, they influence two or more measured variables directly, the intent of such studies is to identify them and their influences. In many cases, for example in sociology, social psychology, neuropsychology, epidemiology, climate research, signal source studies, and elsewhere, the chief aim of inquiry is in fact to identify the causal relations of (often unknown) unrecorded variables that influence multiple recorded variables. It is often assumed on good grounds that recorded variables do not influence unrecorded variables, although in some cases recorded variables may influence one another.

When there is uncertainty about the number of latent variables, which measured variables they influence, or which measured variables influence other measured variables, the investigator who aims at a causal explanation is faced with a difficult discovery problem for which currently avail-

∗. This work was completed while Ricardo Silva was at the School of Computer Science, Carnegie Mellon University.

able methods are at best heuristic. Loehlin (2004) argues that while there are several approaches to automatically learn causal structure, none can be seem as competitors of exploratory factor analysis: the usual focus of automated search procedures for causal Bayes nets is on relations among observed variables. Loehlin's comment overlooks Bayes net search procedures robust to latent variables (Spirtes et al., 2000) and heuristic approaches for learning networks with hidden nodes (Elidan et al., 2000), but the general sense of his comment is correct. For a kind of model widely used in applied sciences – "multiple indicator models" in which multiple observed measures are assumed to be effects of unrecorded variables and possibly of each other – machine learning has provided no principled alternative to factor analysis, principal components, and regression analysis of proxy scores formed from averages or weighted averages of measured variables, the techniques most commonly used to estimate the existence and influences of variables that are unrecorded. The statistical properties of models produced by these methods are well understood, but there are no proofs, under any general assumptions, of convergence to features of the true causal structure. The few simulation studies of the accuracy of these methods on finite samples with diverse causal structures are not reassuring (Glymour, 1997). The use of proxy scores with regression is demonstrably not consistent, and systematically overestimates dependencies. Better methods are needed.

Yet the common view is that solving this problem is actually impossible, as illustrated by the closing words of a popular textbook on latent variable modeling (Bartholomew and Knott, 1999):

> When we come to models for relationships between latent variables we have reached a point where so much has to be assumed that one might justly conclude that the limits of scientific usefulness have been reached if not exceeded.

This view results from a commitment to factor analysis as *the* method to identify and measure unrecorded common causes of recorded variables. One aim of the following work is to demonstrate that such a commitment is unjustified, and to show that the pessimistic claim that follows from it is false.

We describe a two part method for this problem. The method (1) finds clusters of measured variables that are d-separated by a single unrecorded common cause, if such exists; and (2) finds features of the Markov Equivalence class of causal models for the latent variables. Assuming only multiple indicator structure and principles standard in Bayes net search algorithms, principles assumed satisfied in many domains, especially in the social sciences, the two procedures converge, probability 1 in the large sample limit, to correct information. The completeness of the information obtained about latent structure depends on how thoroughly confounded the measured variables are, but when, for each unknown latent variable, there in fact exists at least a small number of measured variables that are influenced only by that latent variable, the method returns the complete Markov Equivalence class of the latent structure. To complement the theoretical results, we show by simulation studies for several latent structures and for a range of sample sizes that the method identifies the unknown structure more accurately than does factor analysis and a published greedy search algorithm. We also illustrate and compare the procedures with applications to social science cases, where expert opinions about measurement are reasonably firm, but are less so about causal relations among the latent variables.

The focus is on linear models of continuous variables. Although most of our results do not make special assumptions about the choice of a probability family, for practical purposes we further assume in the experiments that variables are multivariate Gaussian. In the very end of the paper, we consider possible generalizations of this approach for non-linear, non-Gaussian and discrete models.

The outline of this paper is as follows:

- **Section 2: Illustrative principles** describes a few examples of the techniques we use to learn causal structure in the presence of latent variables;
- **Section 3: Related work** is a brief exposition of other methods used in latent variable learning. We note how the causal discovery problem cannot be reliably solved by methods created for probabilistic modeling only;
- **Section 4: Notation, assumptions and definitions** contains all relevant definitions and assumptions used throughout this paper for the convenience of the reader;
- **Section 5: Procedures for finding pure measurement models** describes the method we use to solve the first half of the problem, discovering which latents exist and which observed variables measure them;
- **Section 6: Learning the structure of the unobserved** describes the method we use to solve the second half of the problem, discovering the Markov equivalence class that contains the causal graph connecting the latent variables;
- **Section 7: Simulation studies** and **Section 8: Real data applications** contain empirical results with simulated and real data;
- **Section 9: Generalizations** is a brief exposition of related work describing how the methods here introduced could be used to discover partial information in certain other classes models;
- **Section 10: Conclusion** summarizes the contribution of this paper and suggests several avenues of research;

Proofs of theorems and implementation details are given in the Appendix.

2. Illustrative Principles

One widely cited and applied approach to learning causal graphs rely on comparing models that entail different conditional independence constraints in the observed marginal (Spirtes et al., 2000). When latent variables are common causes of all observed variables, as in the domains described in the introduction, no such constraints are expected to exist. Still, when such common causes are *direct* causes of just a few variables, there is much structure that can be discovered, although not by observable independencies. One needs instead a framework that distinguishes among different causal graphs from other forms of constraints in the marginal distribution of the observed variables. This section introduces the type of constraints we use through a few illustrative examples.

Consider Figure 1, where X variables are recorded and L variables (in ovals) are unrecorded and unknown to the investigator. The latent structure, the dependencies of measured variables on individual latent variables, and the linear dependency of the measured variables on their parents and (unrepresented) independent noises in Figure 1 imply a pattern of constraints on the covariance matrix among the X variables. For example, X_1, X_2, X_3 have zero covariances with X_7, X_8, X_9. Less

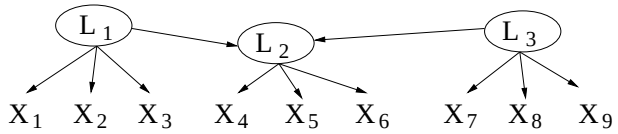


Figure 1: A latent variable model which entails several constraints on the observed covariance matrix. Latent variables are inside ovals.

obviously, for X_1, X_2, X_3 and any one of X_4, X_5, X_6, three quadratic constraints (*tetrad* constraints) on the covariance matrix are implied: e.g., for X_4

$$\rho_{12}\rho_{34} = \rho_{14}\rho_{23} = \rho_{13}\rho_{24} \tag{1}$$

where ρ_{12} is the Pearson product moment correlation between X_1, X_2, etc. (Note that any two of the three vanishing tetrad differences above entails the third.) The same is true for X_7, X_8, X_9 and any one of X_4, X_5, X_6; for X_4, X_5, X_6, and any one of X_1, X_2, X_3 or any one of X_7, X_8, X_9. Further, for any two of X_1, X_2, X_3 or of X_7, X_8, X_9 and any two of X_4, X_5, X_6, exactly one such quadratic constraint is implied, e.g., for X_1, X_2 and X_4, X_5, the single constraint

$$\rho_{14}\rho_{25} = \rho_{15}\rho_{24} \tag{2}$$

The constraints hold as well if covariances are substituted for correlations.

Statistical tests for vanishing tetrad differences are available for a wide family of distributions (Wishart, 1928; Bollen, 1990). Linear and non-linear models can imply other constraints on the correlation matrix, but general, feasible computational procedures to determine arbitrary constraints are not available (Geiger and Meek, 1999) nor are there any available statistical tests of good power for higher order constraints. Tetrad constraints therefore provide a practical way of distinguishing among possible candidate models, with a history of use in heuristic search dating from the early 20th century (see, e.g., references within Glymour et al., 1987). This paper describes a principled way of using tetrad constraints in search.

In particular, we will focus on a class of "pure" latent variable models where latents can be arbitrarily connected in a acyclic causal graph, but where observed variables have at most one latent parent.

Given a "pure" set of measured indicators of latent variables, as in Figure 1 – informally, a measurement model specifying, for each latent variable, a set of measured variables influenced only by that latent variable and individual, independent noises – the causal structure among the latent variables can be estimated by any of a variety of methods. Standard score functions of latent variable models (such as the chi-square test) can be used to compare models with and without a specified edge, providing indirect tests of conditional independence among latent variables. The conditional independence facts can then be input to a constraint based Bayes net search algorithm, such as PC or FCI (Spirtes et al., 2000), or used to guide a greedy search algorithm such as GES (Chickering, 2002).

This is not to say that we need to assume that the true underlying graph contains only pure measures of the latent variables. In Figure 1, the measured variables neatly cluster into disjoint sets of variables and the variables in any one set are influenced only by a single common cause and there

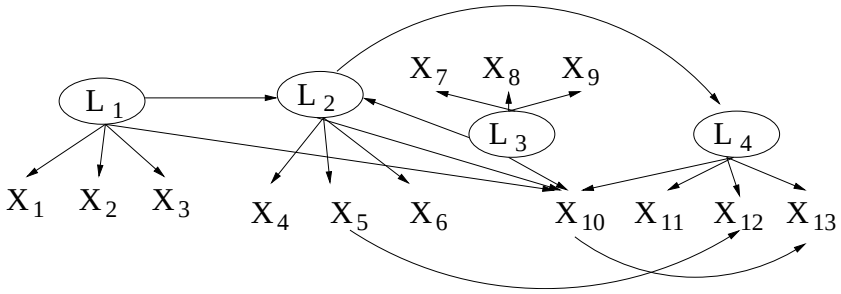


Figure 2: A latent variable model which entails several constraints on the observed covariance matrix. These constraints can be used to discover a submodel of the model given above.

are no influences of the measured variables on one another. In many real cases the influences on the measured variables do not separate so simply. Some of the measured variables may influence others (as in signal leakage between channels in spectral measurements), and some or many measured variables may be influenced by two or more latent variables. For example, the latent structure of a linear, Gaussian system shown in Figure 2 can be recovered by the procedures we propose by finding a *subset* of the given measures that are pure measures in the true graph. Our aim in what follows is to prove and use new results about implied constraints on the covariance matrix of measured variables to form measurement models that enable estimation of features of the Markov Equivalence class of the latent structure in a wide range of cases. We will develop the theory first for linear models (mostly for problems with a joint Gaussian distribution on all variables, including latent variables), and then consider possibilities for generalization.

3. Related Work

The traditional framework for discovering latent variables is factor analysis and its variants (see, e.g., Bartholomew et al., 2002). A number of factors is chosen based on some criterion such as the minimum number of factors that fit the data at a given significance level or the number that maximizes a score such as BIC. After fitting the data, usually assuming a Gaussian distribution, different transformations (rotations) to the latent covariance matrix are applied in order to satisfy some criteria of simplicity. The meaning of a latent variable is determined informally based on the magnitude of the coefficients relating each observed variable to each latent. This is, by far, the most common method used in several applied sciences (Glymour, 2002). Social science methodology also contains various beam searches that begin with an initial latent variable model and iteratively add or delete dependencies in a greedy search guided by significance tests of nested models. In simulation experiments (Glymour et al., 1987; Spirtes et al., 2000) these procedures have performed little better than chance from data generated by true models in which some measured variables are influenced by multiple latent varibles and by other measured variables.

In non-Gaussian cases, the usual methods are variations of independent component analysis, such as independent factor analysis (Attias, 1999) and tree-based component analysis (Bach and Jordan, 2003). These methods severely constrain the dependency structure among the latent vari-

ables. That facilitates joint density estimation or blind source separation, but it is of little use in learning causal structure.

In a similar vein, Zhang (2004) represents latent variable models for discrete variables (both observed and latent) with a multinomial probabilistic model. The model is constrained to be a tree and every observed variable has one and only one (latent) parent and no child. Zhang does not provide a search method to find variables satisfying the assumption, but assumes a priori the variables measured satisfy it.

Elidan et al. (2000) introduces latent variables as common causes of densely connected regions of a DAG learned through Bayesian algorithms for learning Bayesian network structures. Once one latent is introduced as the parent of a set of nodes originally strongly connected, the same search algorithm is applied using this modified graph as the initial graph. The process can be iterated to introduce multiple latents. Examples are given for which this procedure, called FINDHIDDEN, increases the fit over a latent-free graphical model, but for causal modeling the algorithm is not known to be correct in the large sample limit. In a relevant sense, the algorithm cannot be correct, because its output yields particular models from among an indistinguishable class of models that is not characterized.

For instance, consider Figure 3(a), a model of two latents and four observed variables. Two typical outputs produced by FINDHIDDEN given data generated by this model are shown in Figures 3(b) and 3(c). The choice of model is affected by the strength of the connections in the true model and the sample size. These outputs suggest correctly that there is a single latent condition on which all but one pair of observed variables are independent, although the suggestion of some direct causal connection among a pair of indicators is false. The main problem of FINDHIDDEN here is that each of these two models represents a different actual latent variable[1] which is not clear from the output. Graphs given Figures 3(b) and 3(c) are also generated by FINDHIDDEN when the true model has the graphical structure seen in Figure 3(d). In this case, one might be led to infer that there is a latent condition on which three of the indicators are independent, which is not true.

To report all possible structures indistinguishable by the data instead of an arbitrary one is the fundamental difference between purely probabilistically oriented applications (as the ones that motivate the FINDHIDDEN algorithm) and causally oriented applications, as those that motivate this paper. Algorithms such as the ones by Elidan et al. (2000) and Zhang (2004) are designed to effectively perform density estimation, which is a very different problem, even if good density estimators provide one possible causal model compatible with the data.

To tackle issues of sound identifiability of causal structures, we previously developed an approach to learning measurement models (Silva et al., 2003). That procedure requires that the true underlying graph has a "pure" submodel with three measures for each latent variable, which is a strong and generally untestable assumption. That assumption is not needed in the procedures described here, but the output is still a pure model.

One of the reasons why we focus on pure models instead of general latent variable models should be clear from the example in Figure 3: the equivalence class of all latent variable models that cannot be distinguished given the likelihood function might be very large. While, for instance, a Markov equivalence class for models with no latent variables can be neatly represented by a single graphical object known as "pattern" (Pearl, 2000; Spirtes et al., 2000), the same is not true for latent

1. Assuming T_1 in this Figure is the true latent that entails the same conditional independencies. In Figure 3(b), T_1 should correspond to L_2. In Figure 3(c), to L_1. In the first case, however, the causal direction of T_1 into both X_1 and X_2 is wrong and cannot be correctly represented without the introduction of another latent.

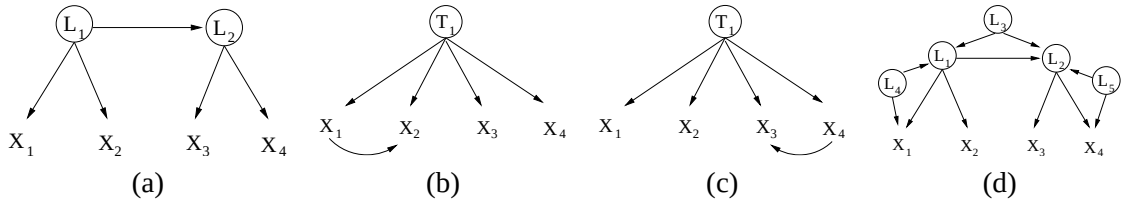


Figure 3: All four models above are undistinguishable in multivariate Gaussian families according to standard algorithms, but such algorithms do not report this fact.

variable models. The models in Figure 3 differ not only in the direction of the edges, but also in the adjacencies themselves ($\{X_1, X_2\}$ adjacent in one case, but not $\{X_3, X_4\}$; $\{X_3, X_4\}$ adjacent in another case, but not $\{X_1, X_2\}$) and the role of the latent variables (ambiguity about which latent d-separates which observed variables, how they are connected, etc.). A representation of such an equivalence class, as illustrated by this very small example, can be cumbersome and uninformative.

4. Notation, Assumptions and Definitions

Our work is in the framework of causal graphical models. Concepts used here without explicit definition, such as d-separation and I-map, can be found in standard sources (Pearl, 1988; Spirtes et al., 2000; Pearl, 2000). We use "variable" and "vertex/node" interchangeably, and standard kinship terminology ("parent," "child," "descendant," "ancestor") for directed graph relationships. Sets of variables are represented in bold, individual variables and symbols for graphs in italics. The Pearson partial correlation of X, Y controlling for Z is denoted by $\rho_{XY.Z}$. We assume i.i.d. data sampled from a subset $\mathbf{O}$ of the variables of a joint distribution D on variables $\mathbf{V} = \mathbf{O} \cup \mathbf{L}$, subject to the following assumptions:

A1 D factors according to the local Markov assumption for a DAG G with vertex set $\mathbf{V}$. That is, any variable is independent of its non-descendants in G conditional on any values of its parents in G.

A2 No vertex in $\mathbf{O}$ is an ancestor of any vertex in $\mathbf{L}$. We call this property the *measurement assumption*;

A3 Each variable in $\mathbf{V}$ is a linear function of its parents plus an additive error term of positive finite variance;

A4 The Faithfulness Assumption: for all $\{X, Y, Z\} \subseteq \mathbf{V}$, X is independent of Y conditional on each assignment of values to variables in Z if and only if the Markov Assumption for G entails such conditional independencies. For models satisfying A1-A3 with Gaussian distributions, Faithfulness is equivalent to assuming that no correlations or partial correlations vanish because of multiple pathways whose influences perfectly cancel one another.

Definition 1 (Linear latent variable model) *A model satisfying A1 − A4 is a linear latent variable model, or for brevity, where the context makes the linearity assumption clear, a latent variable model.*

A single symbol, such as G, will be used to denote both a linear latent variable model and the corresponding latent variable graph. Linear latent variable models are ubiquitous in econometric, psychometric, and social scientific studies (Bollen, 1989), where they are usually known as structural equation models.

Definition 2 (Measurement model) *Given a linear latent variable model G, with vertex set $\mathbf{V}$, the subgraph containing all vertices in $\mathbf{V}$, and all and only those edges directed into vertices in $\mathbf{O}$, is called the measurement model of G.*

Definition 3 (Structural model) *Given a linear latent variable model G, the subgraph containing all and only its latent nodes and respective edges is the structural model of G.*

Definition 4 (Linear entailment) *We say that a DAG G linearly entails a constraint if and only if the constraint holds in every distribution satisfying A1 - A4 for G with covariance matrix parameterized by Θ, the set of linear coefficients and error variances that defines the conditional expectation and variance of a vertex given its parents. We will assume without loss of generality that all variables have zero mean.*

Definition 5 (Tetrad equivalence class) *Given a set $\mathbf{C}$ of vanishing partial correlations and vanishing tetrad differences, a tetrad equivalence class $\mathcal{T}(\mathbf{C})$ is the set of all latent variable graphs each member of which entails all and only the tetrad constraints and vanishing partial correlations among the measured variables entailed by $\mathbf{C}$.*

Definition 6 (Measurement equivalence class) *An equivalence class of measurement models $\mathcal{M}(\mathbf{C})$ for $\mathbf{C}$ is the union of the measurement models graphs in $\mathcal{T}(\mathbf{C})$. We introduce a graphical representation of common features of all elements of $\mathcal{M}(\mathbf{C})$, analogous to the familiar notion of a pattern representing the Markov Equivalence class of a Bayes net.*

Definition 7 (Measurement pattern) *A measurement pattern, denoted $\mathcal{MP}(\mathbf{C})$, is a graph representing features of the equivalence class $\mathcal{M}(\mathbf{C})$ satisfying the following:*

- *there are latent and observed vertices;*
- *the only edges allowed in an MP are directed edges from latent variables to observed variables, and undirected edges between observed vertices;*
- *every observed variable in a MP has at least one latent parent;*
- *if two observed variables X and Y in a $\mathcal{MP}(\mathbf{C})$ do not share a common latent parent, then X and Y do not share a common latent parent in any member of $\mathcal{M}(\mathbf{C})$;*
- *if observed variables X and Y are not linked by an undirected edge in $\mathcal{MP}(\mathbf{C})$, then X is not an ancestor of Y in any member of $\mathcal{M}(\mathbf{C})$.*

Definition 8 (Pure measurement model) *A pure measurement model is a measurement model in which each observed variable has only one latent parent, and no observed parent. That is, it is a tree beneath the latents.*

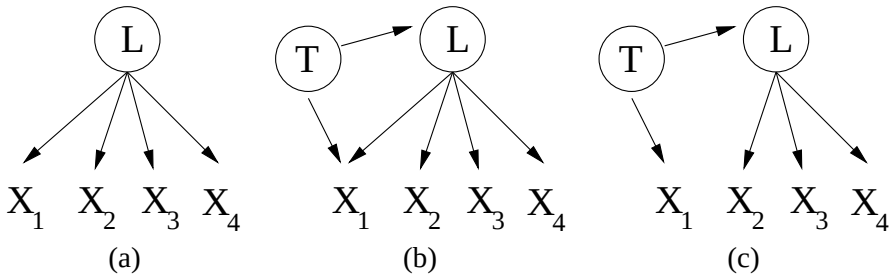


Figure 4: A linear latent variable model with any of the graphical structures above entails all possible tetrad constraints in the marginal covariance matrix of $X_1 - X_4$.

5. Procedures for Finding Pure Measurement Models

Our goal is to find pure measurement models whenever possible, and use them to estimate the structural model. To do so, we first use properties relating graphical structure and covariance constraints to identify a measurement pattern, and then turn the measurement pattern into a pure measurement model.

The key to solving this problem is a graphical characterization of tetrad constraints. Consider Figure 4(a). A single latent d-separates four observed variables. When this graphical model is linearly parameterized as

$$\begin{aligned} X_1 &= \lambda_1 L + \varepsilon_1 \\ X_2 &= \lambda_2 L + \varepsilon_2 \\ X_3 &= \lambda_3 L + \varepsilon_3 \\ X_4 &= \lambda_4 L + \varepsilon_4 \end{aligned}$$

it entails all three tetrad constraints among the observed variables. That is, any choice of values for coefficients $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ and error variances implies

$$\begin{aligned} \sigma_{X_1X_2}\sigma_{X_3X_4} &= (\lambda_1\lambda_2\sigma_L^2)(\lambda_3\lambda_4\sigma_L^2) = (\lambda_1\lambda_3\sigma_L^2)(\lambda_2\lambda_4\sigma_L^2) = \sigma_{X_1X_3}\sigma_{X_2X_4} \\ &= (\lambda_1\lambda_2\sigma_L^2)(\lambda_3\lambda_4\sigma_L^2) = (\lambda_1\lambda_4\sigma_L^2)(\lambda_2\lambda_3\sigma_L^2) = \sigma_{X_1X_4}\sigma_{X_2X_3} \end{aligned}$$

where σ_L^2 is the variance of latent variable L.

While this result is straightforward, the relevant result for a structure learning algorithm is the converse, i.e., establishing equivalence classes from observable tetrad constraints. For instance, Figure 4(b) and (c) are different structures with the same entailed tetrad constraints that should be accounted for. The main contribution of this paper is to provide several of such identification results, and sound algorithms for learning causal structure based on them. Such results require elaborate proofs that are left to the Appendix. What follows are descriptions of the most significant lemmas and theorems, and illustrative examples. This is the core section of the paper. Section 6 complements the approach by describing an algorithm for learning structural models.

5.1 Identification Rules for Finding Substructures of Latent Variable Graphs

We start with one of the most basic lemmas, used as a building block for later results. It is basically the converse of the observation above. Let G be a linear latent variable model with observed variables $\mathbf{O}$:

Lemma 9 *Let* $\{X_1, X_2, X_3, X_4\} \subset \mathbf{O}$ *be such that* $\sigma_{X_1X_2}\sigma_{X_3X_4} = \sigma_{X_1X_3}\sigma_{X_2X_4} = \sigma_{X_1X_4}\sigma_{X_2X_3}$. *If* $\rho_{AB} \neq 0$ *for all* $\{A, B\} \subset \{X_1, X_2, X_3, X_4\}$, *then there is a node P that d-separates all elements* $\{X_1, X_2, X_3, X_4\}$ *in G.*

It follows that, if no observed node d-separates $\{X_1, X_2, X_3, X_4\}$, then node P must be a latent node.

In order to learn a pure measurement model, we basically need two pieces of information: i. which sets of nodes are d-separated by a latent; ii. which sets of nodes do not share any common hidden parent. The first piece of information can provide possible indicators (children/descendants) of a specific latent. However, this is not enough information, since a set $\mathbf{S}$ of observed variables can be d-separated by a latent L, and yet $\mathbf{S}$ might contain non-descendants of L (one of the nodes might have a common ancestor with L and not be a descendant of L, for instance). This is the reason why we need to *cluster* observed variables into different sets when it is possible to show they cannot share a common hidden parent. We will show this clustering allows us to eliminate most possible non-descendants.

There are several possible combinations of observable tetrad constraints that allow one to identify such a clustering. Consider, for instance, the following case, in which it is determined that certain variables do not share a common latent. Suppose we have a set of six observable variables, X_1, X_2, X_3, Y_1, Y_2 and Y_3 such that:

1. there is some latent node that d-separates all pairs in $\{X_1, X_2, X_3, Y_1\}$ (Figure 5(a));
2. there is some latent node that d-separates all pairs in $\{X_1, Y_1, Y_2, Y_3\}$ (Figure 5(b));
3. there is no tetrad constraint $\sigma_{X_1X_2}\sigma_{Y_1Y_2} - \sigma_{X_1Y_2}\sigma_{X_2Y_1} = 0$;
4. no pairs in $\{X_1, \ldots, Y_3\} \times \{X_1, \ldots, Y_3\}$ have zero correlation;

Notice that is possible to empirically verify the first two conditions by using Lemma 9. Now suppose, for the sake of contradiction, that X_1 and Y_1 have a common hidden parent L. One can show that L should d-separate all elements in $\{X_1, X_2, X_3, Y_1\}$, and also in $\{X_1, Y_1, Y_2, Y_3\}$. With some extra work (one has to consider the possibility of nodes in $\{X_1, X_2, Y_1, Y_2\}$ having common parents with L, for instance), one can show that this implies that L d-separates $\{X_1, Y_1\}$ from $\{X_2, Y_2\}$. For instance, Figure 5(c) illustrates a case where L d-separates all of the given observed variables.

However, this contradicts the third item in the hypothesis (such a d-separation will imply the forbidden tetrad constraint, as we show in the formal proof) and, as a consequence, no such L should exist. Therefore, the items above correspond to an *identification rule* for discovering some d-separations concerning observed and hidden variables (in this case, we show that X_1 is independent of all latent parents of Y_1 given some latent ancestor of X_1). This rule only uses constraints that can be tested from the data.

Given such identification rules, what is needed is a principled way of combining the partial information they provide to build classes of latent variable models of interest. The following section explains the main rules and an algorithm for building an equivalence class of measurement models.

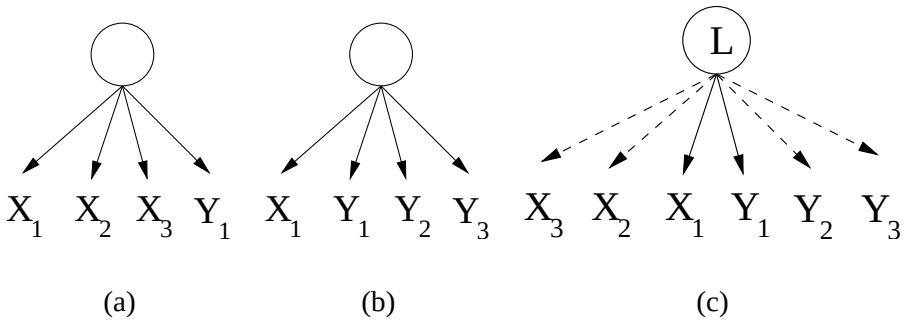


Figure 5: If sets $\{X_1, X_2, X_3, Y_1\}$ and $\{X_1, Y_1, Y_2, Y_3\}$ are each d-separated by some node (e.g., as in Figures (a) and (b) above), the existence of a common parent L for X_1 and Y_1 implies a common node d-separating $\{X_1, Y_1\}$ from $\{X_2, Y_2\}$, for instance (as exemplified in Figure (c)).

5.2 Algorithms for Finding Equivalence Classes of Latent Variable Graphs

We start with one of the most basic lemmas, used as a building block for later results. We discover a measurement pattern as an intermediate step before learning a pure measurement model. FINDPATTERN, given in Table 1, is an algorithm to learn a measurement pattern from an oracle for vanishing partial correlations and vanishing tetrad differences. The algorithm uses three rules, CS1, CS2, CS3, based on Lemmas that follow, for determining graphical structure from constraints on the correlation matrix of observed variables.

Let **C** be a set of linearly entailed constraints satisfied in the observed covariance matrix. The first stage of FINDPATTERN searches for subsets of **C** that will guarantee that two observed variables do not have any latent parent in common. Let G be the latent variable graph for a linear latent variable model with a set of observed variables **O**. Let $\mathbf{O}' = \{X_1, X_2, X_3, Y_1, Y_2, Y_3\} \subset \mathbf{O}$ such that for all triplets $\{A, B, C\}$, $\{A, B\} \subset \mathbf{O}'$ and $C \in \mathbf{O}$, we have $\rho_{AB} \neq 0, \rho_{AB.C} \neq 0$. Let τ_{IJKL} represent the tetrad constraint $\sigma_{IJ}\sigma_{KL} - \sigma_{IK}\sigma_{JL} = 0$ and $\neg\tau_{IJKL}$ represent the complementary constraint $\sigma_{IJ}\sigma_{KL} - \sigma_{IK}\sigma_{JL} \neq 0$. The following Lemma is a formal description of the example given earlier:

Lemma 10 (CS1 Test) *If constraints* $\{\tau_{X_1Y_1X_2X_3}, \tau_{X_1Y_1X_3X_2}, \tau_{Y_1X_1Y_2Y_3}, \tau_{Y_1X_1Y_3Y_2}, \neg\tau_{X_1X_2Y_2Y_1}\}$ *all hold, then* X_1 *and* Y_1 *do not have a common parent in* G.

"CS" here stands for "constraint set," the premises of a rule that can be used to test if two nodes do not share a common parent. Figure 6(a) illustrates one situation where X_1 and Y_1 can be identified to not measure a same latent. In that Figure, some variables are specified with unexplained correlations represented as bidirected edges between the variables (such edges could be due to independent hidden common causes, for instance). This illustrates that connections between elements of $\{X_2, X_3, Y_2, Y_3\}$ can occur.

Other sets of observable constraints can be used to reach the same conclusion. We call them CS2 and CS3. To see one of the limitations of CS1, consider Figure 6(b). There is no single latent that d-separates X_1, Y_1 and two other variables, as in CS1 cases. In Figure 6(c), there are no tetrad

Algorithm FINDPATTERN
Input: a covariance matrix Σ

1. Start with a complete undirected graph G over the observed variables.
2. Remove edges for pairs that are marginally uncorrelated or uncorrelated conditioned on a third observed variable.
3. For every pair of nodes linked by an edge in G, test if some rule CS1, CS2 or CS3 applies. Remove an edge between every pair corresponding to a rule that applies.
4. Let H be a graph with no edges and with nodes corresponding to the observed variables.
5. For each maximal clique in G, add a new latent to H and make it a parent to all corresponding nodes in the clique.
6. For each pair (A,B), if there is no other pair (C,D) such that $\sigma_{AC}\sigma_{BD} = \sigma_{AD}\sigma_{BC} = \sigma_{AB}\sigma_{CD}$, add an undirected edge $A - B$ to H.
7. Return H.

Table 1: Returns a measurement pattern corresponding to the tetrad and first order vanishing partial correlations of Σ.

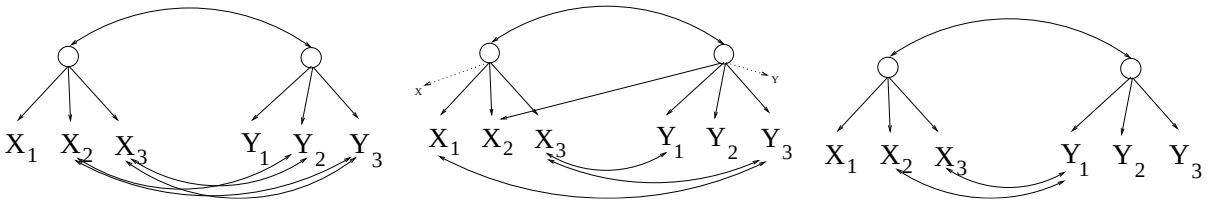
constraints simultaneously involving X_1, Y_1 and other observed variables that are children of the same latent parent of X_1. These extra rules are not as intuitive as CS1. To fully understand how these cases still generate useful constraints, some knowledge of the graphical implications of tetrad constraints is necessary. To avoid interrupting the flow of the paper, we describe these properties only in the Appendix along with formal proofs of correctness. In the next paragraphs, we just describe rules CS2 and CS3.

Let the predicate $Factor(X,Y,G)$ be true if and only if there exist two nodes W and Z in latent variable graph G such that τ_{WXYZ} and τ_{WXZY} are both linearly entailed by G, all variables in $\{W,X,Y,Z\}$ are correlated, and there is no observed C in G such that $\rho_{AB.C} = 0$ for $\{A,B\} \subset \{W,X,Y,Z\}$:

Lemma 11 (CS2 Test) *If constraints* $\{\tau_{X_1Y_1Y_2X_2}, \tau_{X_2Y_1Y_3Y_2}, \tau_{X_1X_2Y_2X_3}, \neg\tau_{X_1X_2Y_2Y_1}\}$ *all hold such that* $Factor(X_1,X_2,G) = true$, $Factor(Y_1,Y_2,G) = true$, X_1 *is not an ancestor of* X_3 *and* Y_1 *is not an ancestor of* Y_3, *then* X_1 *and* Y_1 *do not have a common parent in* G.

Lemma 12 (CS3 Test) *If constraints* $\{\tau_{X_1Y_1Y_2Y_3}, \tau_{X_1Y_1Y_3Y_2}, \tau_{X_1Y_2X_2X_3}, \tau_{X_1Y_2X_3X_2}, \tau_{X_1Y_3X_2X_3}, \tau_{X_1Y_3X_3X_2}, \neg\tau_{X_1X_2Y_2Y_3}\}$ *all hold, then* X_1 *and* Y_1 *do not have a common parent in* G.

The rules are not redundant: only one can be applied on each situation. For instance, in Figure 6(a) the latent on the left d-separates $\{X_1,X_2,X_3,Y_1\}$, which implies $\{\tau_{X_1Y_1Y_2Y_3}, \tau_{X_1Y_1Y_3Y_2}\}$. The latent on the right d-separates $\{X_1,Y_1,Y_2,Y_3\}$, which implies $\{\tau_{Y_1X_1Y_2Y_3}, \tau_{Y_1X_1Y_3Y_2}\}$. The constraint $\tau_{X_1X_2Y_2Y_1}$ can be shown not to hold given the assumptions. Therefore, this rule tells us information about the unobserved structure: X_1 and Y_1 do not have any common hidden parent.



(a) (b) (c)

Figure 6: Three examples with two main latents and several independent latent common causes of two indicators (represented by bidirected edges). In (a), CS1 applies, but not CS2 nor CS3 (even when exchanging labels of the variables); In (b), CS2 applies (assuming the conditions for X_1, X_2 and Y_1, Y_2), but not CS1 nor CS3. In (c), CS3 applies, but not CS1 nor CS2.

For CS2 (Figure 6(b)), nodes X and Y are depicted as auxiliary nodes that can be used to verify predicates *Factor*. For instance, $Factor(X_1, X_2, G)$ is true because all three tetrads in the covariance matrix of $\{X_1, X_2, X_3, X\}$ hold.

Sometime it is possible to guarantee that a node is not an ancestor of another, as required, e.g., to apply CS2:

Lemma 13 *If for some set* $\mathbf{O}' = \{X_1, X_2, X_3, X_4\} \subseteq \mathbf{O}$, $\sigma_{X_1X_2}\sigma_{X_3X_4} = \sigma_{X_1X_3}\sigma_{X_2X_4} = \sigma_{X_1X_4}\sigma_{X_2X_3}$ *and for all triplets* $\{A, B, C\}$, $\{A, B\} \subset \mathbf{O}', C \in \mathbf{O}$, *we have* $\rho_{AB.C} \neq 0$ *and* $\rho_{AB} \neq 0$, *then* $A \in \mathbf{O}'$ *is not a descendant in G of any element of* $\mathbf{O}' \setminus \{A\}$.

This follows immediately from Lemma 9 and the assumption that observed variables are not ancestors of latent variables. For instance, in Figure 6(b) the existence of the observed node X (linked by a dashed edge to the parent of X_1) allows the inference that X_1 is not an ancestor of X_3, since all three tetrad constraints hold in the covariance matrix of $\{X, X_1, X_2, X_3\}$.

We know have theoretical results that provide information concerning lack of common parents and lack of direct connections of nodes, given a set of tetrad and vanishing partial correlation **C**. The algorithm FINDPATTERN from Table 1 essentially uses the given lemmas to construct a measurement pattern, as defined in Section 4.

Theorem 14 *The output of* FINDPATTERN *is a measurement pattern* $\mathcal{MP}(\mathbf{C})$ *with respect to the tetrad and zero/first order vanishing partial correlation constraints* **C** *of* Σ.

The presence of an undirected edge does not mean that adjacent vertices in the pattern are actually adjacent in the true graph. Figure 7 illustrates this: X_3 and X_8 share a common parent in the true graph, but are not adjacent. Observed variables adjacent in the output pattern always share at least one parent in the pattern, but do not always share a common parent in the true DAG. Vertices sharing a common parent in the pattern might not share a parent in the true graph (e.g., X_1 and X_8 in Figure 7).

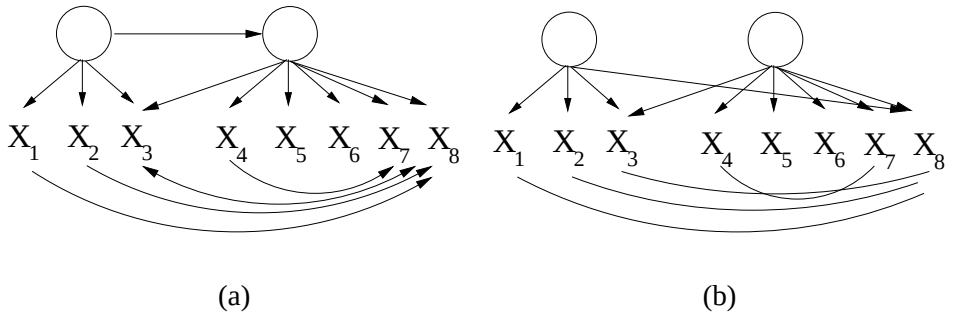


Figure 7: In (a), a model that generates a covariance matrix Σ. In (b), the output of FINDPATTERN given Σ. Pairs in $\{X_1, X_2\} \times \{X_4, \ldots, X_7\}$ are separated by CS2.

What is not obvious in the output of FINDPATTERN is how much more information it leaves implicit and how to extract a (pure) model out of an equivalence class. These issues are treated in the next section.

5.3 Completeness and Purification

The FINDPATTERN algorithm is sound, but not necessarily complete. That is, there might be graphical features shared by all members of the measurement model equivalence class that are not discovered by FINDPATTERN. For instance, there might be a CS4 rule that is not known to us. FINDPATTERN might be complete, but we conjecture it is not: we did not try to construct rules using more than 6 variables (unlike CS1, CS2, CS3), since the more variables a rule has, the more computational expensive and the less statistically reliable it is.[2] Learning a pure measurement model is a different matter. We can find a pure measurement model with the largest number of latents in the true graph, for instance.

A pure measurement model implies a *clustering* of observed variables: each cluster is a set of observed variables that share a common (latent) parent, and the set of latents defines a partition over the observed variables. The output of FINDPATTERN cannot, however, reliably be turned into a pure measurement pattern in the obvious way, by removing from H all nodes that have more than one latent parent and one of every pair of adjacent nodes, as attemped by the following algorithm:

- Algorithm TRIVIALPURIFICATION: remove all nodes that have more than one latent parent, and for every pair of adjacent observed nodes, remove an arbitrary node of the pair.

TRIVIALPURIFICATION is not correct. To see this, consider Figure 8(a), where with the exception of pairs in $\{X_3, \ldots, X_7\}$, every pair of nodes has more than one hidden common cause. Giving the covariance matrix of such model to FINDPATTERN will result in a pattern with one latent only (because no pair of nodes can be separated by CS1, CS2 or CS3), and all pairs that are connected by a double directed edge in Figure 8(a) will be connected by an undirected edge in the output pattern. One can verify that if we remove one node from each pair connected by an undirected edge in this pattern, the output with the maximum number of nodes will be given by the graph in Figure 8(b).

2. Under very general conditions, there are also no rules using fewer than 6 variables, as shown by Silva (2005).

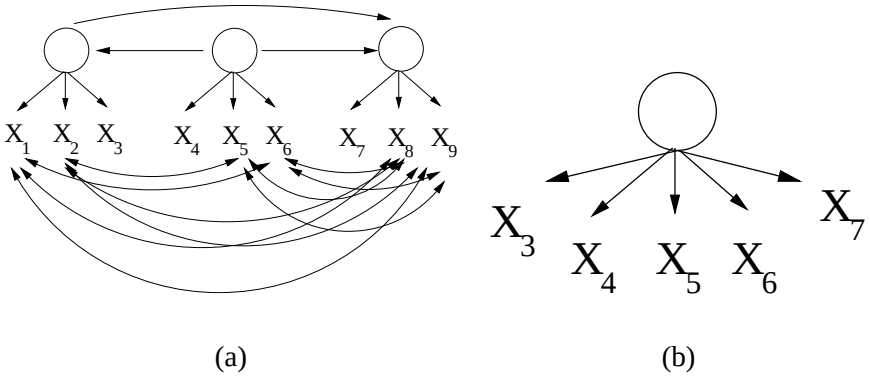


Figure 8: In (a), a model that generates a covariance matrix Σ. The output of FINDPATTERN given Σ contains a single latent variable that is a parent of all observed nodes, and several observed nodes that are linked by an undirected edge. In (b), the pattern with the maximum number of nodes that can be obtained by TRIVIALPURIFICATION. It is still not a correct pure measurement model for any latent in the true graph, since there is no latent that d-separates $\{X_3, \ldots, X_7\}$ in the true model.

The procedure BUILDPURECLUSTERS builds a pure measurement model using as input FINDPATTERN and an oracle for constraints. Unlike TRIVIALPURIFICATION, variables are removed whenever appropriate tetrad constraints are not satisfied. Table 2 presents a simplified version of the full algorithm. The complete algorithm is given only in Appendix A to avoid interrupting the flow of the text, since it requires the explanation of extra steps that are not of much relevance in practice. We also describe the choices made in the algorithm (Steps 2, 4 and 5) only in the implementation given in Appendix A. The particular strategy for making such choices is not relevant to the correctness of the algorithm.

The fundamental properties of BUILDPURECLUSTERS are clear from Table 2: it returns a model where each latent has at least three indicators, and such indicators are known to be d-separated by some latent. Nodes that are children of different latents in the output graph are known not to be children of a common latent in the true graph, as defined by the initial measurement pattern. However, it is not immediately obvious how latents in the output graph are related to latents in the true graph.

The informal description is: there is a labeling of latents in the output graph according to the latents in the true graph G, and in this relabeled output graph any d-separation between a measured node and some other node will hold in G. This is illustrated by Figure 9. Given the covariance matrix generated by the true model in Figure 9(a), BUILDPURECLUSTERS generates the model shown in Figure 9(b).

Since the labeling of the latents is arbitrary, we need a formal description of the fact that latents in the output should correspond to latents in the true model up to a relabeling. The formal graphical properties of the output of BUILDPURECLUSTERS (as given in Appendix A) are summarized by the following theorem:

Algorithm BUILDPURECLUSTERS-SIMPLIFIED
Input: a covariance matrix Σ

1. $G \leftarrow$FINDPATTERN(Σ).
2. Choose a set of latents in G. Remove all other latents and all observed nodes that are not children of the remaining latents and all clusters of size 1.
3. Remove all nodes that have more than one latent parent in G.
4. For all pairs of nodes linked by an undirected edge, choose one element of each pair to be removed.
5. If for some set of nodes $\{A,B,C\}$, all children of the same latent, there is a fourth node D in G such that $\sigma_{AB}\sigma_{CD} = \sigma_{AC}\sigma_{BD} = \sigma_{AD}\sigma_{BC}$ is *not* true, remove one of these four nodes.
6. Remove all latents with less than three children, and their respective measures;
7. if G has at least four observed variables, return G. Otherwise, return an empty model.

Table 2: A general strategy to find a pure measurement measurement model of a subset of the latents in the true graph. As explained in the body of the text, implementation details (such as the choices made in Steps 2 and 4) are left to Appendix A.

Theorem 15 *Given a covariance matrix Σ assumed to be generated from a linear latent variable model G with observed variables $\mathbf{O}$ and latent variables $\mathbf{L}$, let G_{out} be the output of* BUILDPURE-CLUSTERS(Σ) *with observed variables $\mathbf{O_{out}} \subseteq \mathbf{O}$ and latent variables $\mathbf{L_{out}}$. Then G_{out} is a measurement pattern, and there is an unique injective mapping $M : \mathbf{L_{out}} \rightarrow \mathbf{L}$ with the following properties:*

1. *Let $L_{out} \in \mathbf{L_{out}}$. Let X be a child of L_{out} in G_{out}. Then $M(L_{out})$ d-separates X from $\mathbf{O_{out}} \backslash X$ in G;*
2. *$M(L_{out})$ d-separates X from every latent L in G for which $M^{-1}(L)$ is defined;*
3. *Let $\mathbf{O'} \subseteq \mathbf{O_{out}}$ be such that each pair in $\mathbf{O'}$ is correlated. At most one element in $\mathbf{O'}$ has the following property: (i) it is not a descendant of its respective mapped latent parent in G or (ii) it has a hidden common cause with its respective mapped latent parent in G;*

For each group of correlated observed variables, we can guaranteee that at most one edge from a latent into an observed variable is incorrectly directed. By "incorrectly directed," we mean the condition defined in the third item of Theorem 15: although all observed variables are children of latents in the output graph, one of these edges might be misleading, since in the true graph one of the observed variables might not be a descendant of the respective latent. This is illustrated by Figure 10.

Notice also that we cannot guarantee that an observed node X with latent parent L_{out} in G_{out} will be d-separated from the latents in G not in G_{out}, given $M(L_{out})$: if X has a common cause with $M(L_{out})$, then X will be d-connected to any ancestor of $M(L_{out})$ in G given $M(L_{out})$. This is also illustrated by Figure 10.

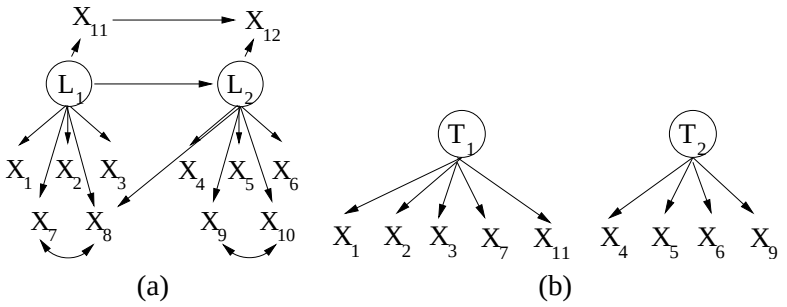


Figure 9: Given as input the covariance matrix of the observable variables $X_1 - X_{12}$ connected according to the true model shown in Figure (a), the BUILDPURECLUSTERS algorithm will generate the graph shown in Figure (b). It is clear there is an injective mapping $M(.)$ from latents $\{T_1, T_2\}$ to latents $\{L_1, L_2\}$ such that $M(T_1) = L_1$ and $M(T_2) = L_2$ and the properties described by Theorem 15 hold.

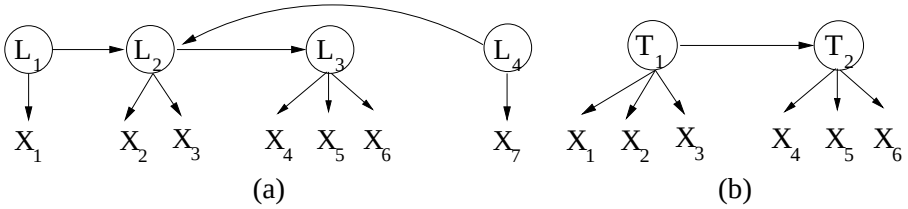


Figure 10: Given as input the covariance matrix of the observable variables $X_1 - X_7$ connected according to the true model shown in Figure (a), one of the possible outputs of BUILDPURECLUSTERS algorithm is the graph shown in Figure (b). It is clear there is an injective mapping $M(.)$ from latents $\{T_1, T_2\}$ to latents $\{L_1, L_2, L_3, L_4\}$ such that $M(T_1) = L_2$ and $M(T_2) = L_3$. However, in (b) the edge $T_1 \rightarrow X_1$ does not express the correct causal direction of the true model. Notice also that X_1 is not d-separated from L_4 given $M(T_1) = L_2$ in the true graph.

5.4 An Example

To illustrate BUILDPURECLUSTERS, suppose the true graph is the one given in Figure 11(a), with two unlabeled latents and 12 observed variables. This graph is unknown to BUILDPURECLUSTERS, which is given only the covariance matrix of variables $\{X_1, X_2, ..., X_{12}\}$. The task is to learn a measurement pattern, and then a purified measurement model.

In the first stage of BUILDPURECLUSTERS, the FINDPATTERN algorithm, we start with a fully connected graph among the observed variables (Figure 11(b)), and then proceed to remove edges according to rules CS1, CS2 and CS3, giving the graph shown in Figure 11(c). There are two maximal cliques in this graph: $\{X_1, X_2, X_3, X_7, X_8, X_{11}, X_{12}\}$ and $\{X_4, X_5, X_6, X_8, X_9, X_{10}, X_{12}\}$. They are distinguished in the figure by different edge representations (dashed and solid - with the edge $X_8 - X_{12}$ present in both cliques). The next stage takes these maximal cliques and creates an intermediate

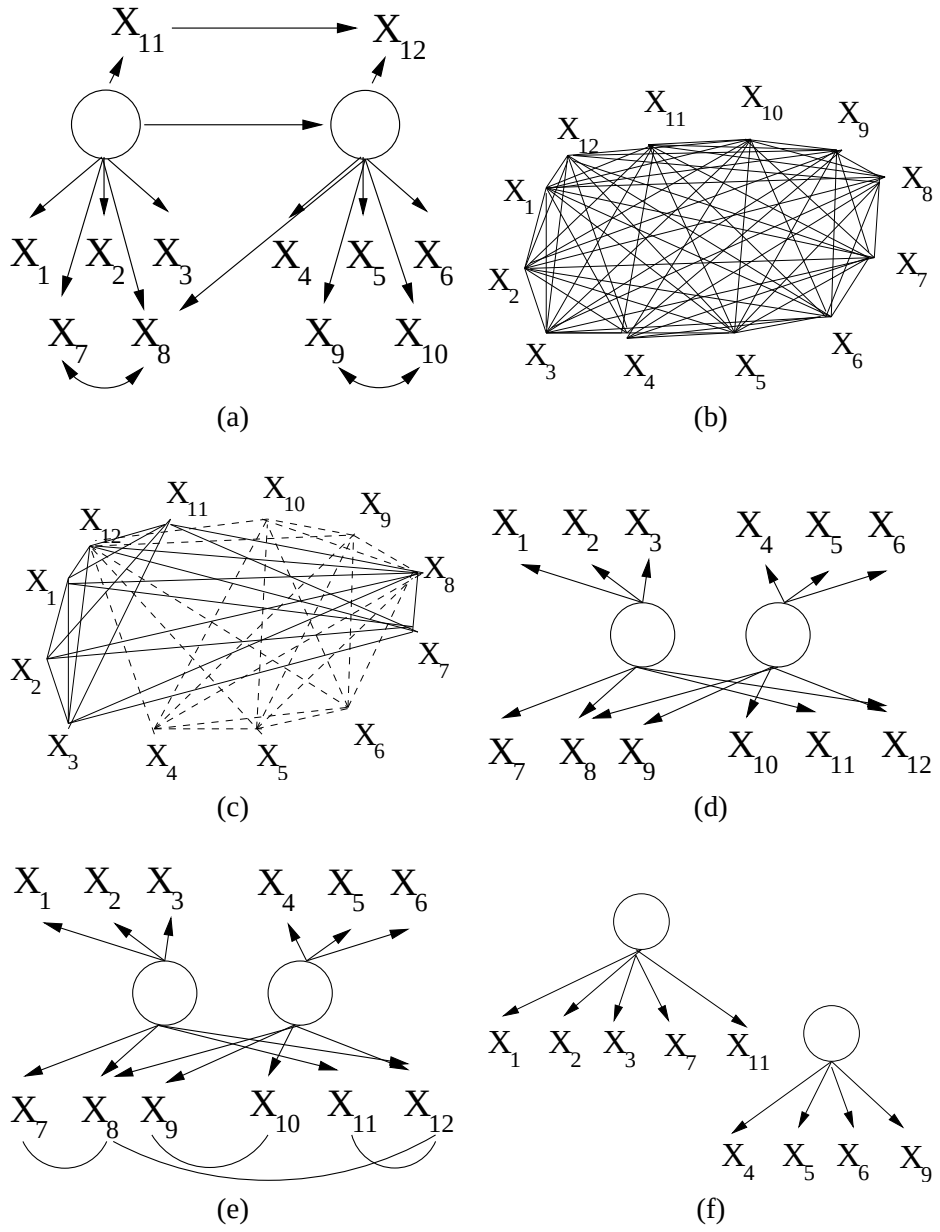


Figure 11: A step-by-step demonstration of how a covariance matrix generated by graph in Figure (a) will induce the pure measurement model in Figure (f).

graphical representation, as depicted in Figure 11(d). In Figure 11(e), we add the undirected edges $X_7 - X_8$, $X_8 - X_{12}$, $X_9 - X_{10}$ and $X_{11} - X_{12}$, finalizing the measurement pattern returned by FINDPATTERN. Finally, Figure 11(f) represents a possible purified output of BUILDPURECLUSTERS given this pattern. Another purification with as many nodes as in the graph in Figure 11(f) substitutes node X_9 for node X_{10}.

There is some superficial similarity between BUIDPURECLUSTERS and the FINDHIDDEN algorithm (Elidan et al., 2000) cited in Section 3. Both algorithms select cliques (or substructures close to a clique) and introduce a latent as a common cause of the variables in that clique. The algorithms are, however, very different: BUILDPURECLUSTERS knows that each selected clique should correspond to a latent,[3] and creates all of its latents at the same time. FINDHIDDEN creates one latent a time, and might backtrack if this latent is not supported by the data. More fundamentally, there is no clear description of what FINDHIDDEN actually learns (as illustrated in Section 3), and even if asymptotically it can always find a pure measurement submodel.[4]

5.5 Parameterizing the Output of BUILDPURECLUSTERS

Recall that so far we described only an algorithm for learning measurement models. Learning the structure among latents, as described in Section 6, requires exploring constraints in the covariance matrix of the observed variables. Since BUILDPURECLUSTERS returns only a marginal of the true model, it is important to show that this marginalized graph, when parameterized as a linear model, also represents the marginal probability distribution of the observed variables.

The following result is essential to provide an algorithm that is guaranteed to find a Markov equivalence class for the latents in $M(\mathbf{L_{out}})$ using the output of BUILDPURECLUSTERS, as in Section 6. It guarantees that one can fit a linear model using the structure given by BUILDPURECLUSTERS and have a consistent estimator of the observed covariance matrix (for the selected variables) in families such as Gaussian distributions. This is important, since the covariance matrix of the observed variables in the model is used to guide the search for a structure among latents, as discussed in Section 6.

Theorem 16 *Let $M(\mathbf{L_{out}}) \subseteq \mathbf{L}$ be the set of latents in G obtained by the mapping function $M()$. Let $\Sigma_{\mathbf{O_{out}}}$ be the population covariance matrix of $\mathbf{O_{out}}$. Let the DAG G^{aug}_{out} be G_{out} augmented by connecting the elements of $\mathbf{L_{out}}$ such that the structural model of G^{aug}_{out} is an I-map of the distribution of $M(\mathbf{L_{out}})$. Then there exists a linear latent variable model using G^{aug}_{out} as the graphical structure such that the implied covariance matrix of $\mathbf{O_{out}}$ equals $\Sigma_{\mathbf{O_{out}}}$.*

5.6 Computational Issues and Anytime Properties

A further reason why we do not provide details of some steps of BUILDPURECLUSTERS at this point is because there is no unique way of implementing it, and different purifications might be of interest. For instance, one might be interested in the pure model that has the largest possible number of latents. Another one might be interested in the model with the largest number of observed variables. However, some of these criteria might be computationally intractable to achieve. Consider for instance the following criterion, which we denote as $\mathcal{MP}^3$: given a measurement pattern, decide if there is some choice of observed nodes to be removed such that the resulting graph is a pure measurement model of all latents in the pattern and each latent has at least three children. This problem is intractable:

Theorem 17 *Problem $\mathcal{MP}^3$ is NP-complete.*

3. Some latents might be eliminated for not having enough indicators, though.
4. This, of course, bears no fundamental implication on the ability of FINDHIDDEN to generate a model that provides a good fit to the data, but it is a crucial limitation in causal analysis.

There is no need to solve a NP-hard problem in order to have the theoretical guarantees of interpretability of the output given by Theorem 15. For example, there is a stage in FINDPATTERN where it appears necessary to find all maximal cliques, but, in fact, it is not. Identifying more cliques increases the chance of having a larger output (which is good) by the end of the algorithm, but it is not required for the algorithms correctness. Stopping at Step 5 of FINDPATTERN before completion will not affect Theorems 15 or 16.

Another computational concern is the $O(N^5)$ loops in Step 3 of FINDPATTERN, where N is the number of observed variables.[5] Again, it is not necessary to compute this loop entirely. One can stop Step 3 at any time at the price of losing information, but not the theoretical guarantees of BUILDPURECLUSTERS. This anytime property is summarized by the following corollary:

Corollary 18 *The output of* BUILDPURECLUSTERS *retains its guarantees even when rules CS1, CS2 and CS3 are applied an arbitrary number of times in* FINDPATTERN *for any arbitrary subset of nodes and an arbitrary number of maximal cliques is found.*

It is difficult to assess how an early stopping procedure might affect the completeness of the output. In all of our experiments, we were able to enumerate all maximal cliques in a few seconds of computation. This is not to say that one should not design better ways of ordering the clique enumeration (using prior knowledge of which variables should not be clustered together, for instance), or using other alternatives to an anytime stop.

In case there are possibly too many maximal cliques to be enumerated in FINDPATTERN, an alternative to early stopping is to triangulate the graph, i.e., adding edges connecting some non-adjacent pair of nodes in a chordless cycle. This is repeated until no chordless cycles remain in the graph G constructed at the end of Step 3 of FINDPATTERN (Table 1). Different heuristics could be use to choose the next edge to be added, e.g., by linking the pair of nodes that is most strongly correlated. The advantage is that cliques in a triangulated graph can be found in linear time. For the same reasons that validate Corollary 18, such a triangulation will not affect the correctness of the output, since the purification procedure will remove all nodes that need to be removed. In general, adding undirected edges to graph G in FINDPATTERN does not compromise correctness. As a side effect, it might increase the robustness of the algorithm, since some edges of G are likely to be erroneously removed in small sample studies, although more elaborated ways of adding edges back would need to be discussed in detail and are out of the scope of this paper. Such a triangulation procedure, however, might still cause problems, since in the worst case we will obtain a fully connected (and uninformative) graph.[6]

6. Learning the Structure of the Unobserved

The real motivation for finding a pure measurement model is to obtain reliable statistical access to the relations among the latent variables. Given a pure and correct measurement model, even one involving a fairly small subset of the original measured variables, a variety of algorithms exist for finding a Markov equivalence class of graphs over the set of latents in the given measurement model.

5. This immediately follows from, e.g., the definition of CS1: we have to first find a foursome $\{X_1, X_2, Y_1, Y_2\}$ where $\sigma_{X_1X_2}\sigma_{Y_1Y_2} - \sigma_{X_1Y_1}\sigma_{X_2Y_2} \neq 0$, which is a $O(N^4)$ loop. Conditioned on this foursome, we have to find two independent (but distinct) X_3 and Y_3. This requires two (almost) independent loops of $O(N)$ within the $O(N^4)$ loop.

6. We would like to thank an anonymous reviewer for the suggestions in this paragraph.

6.1 Constraint-Based Search

Constraint-based search algorithms rely on decisions about independence and conditional independence among a set of variables to find the Markov equivalence class over these variables. Given a pure and correct measurement model involving at least 2 measures per latent, we can test for independence and conditional independence among the latents, and thus search for equivalence classes of structural models among the latents, by taking advantage of the following theorem (Spirtes et al., 2000):

Theorem 19 *Let G be a pure linear latent variable model. Let L_1, L_2 be two latents in G, and $\mathbf{Q}$ a set of latents in G. Let X_1 be a measure of L_1, X_2 be a measure of L_2, and $\mathbf{X_Q}$ be a set of measures of $\mathbf{Q}$ containing at least two measures per latent. Then L_1 is d-separated from L_2 given $\mathbf{Q}$ in G if and only if the rank of the correlation matrix of $\{X_1, X_2\} \cup \mathbf{X_Q}$ is less than or equal to $|\mathbf{Q}|$ with probability 1 with respect to the Lebesgue measure over the linear coefficients and error variances of G.*

We can then use this constraint to test[7] for conditional independencies among the latents. Such conditional independence tests can then be used as an oracle for constraint-satisfaction techniques for causality discovery in graphical models, such as the PC algorithm (Spirtes et al., 2000) or the FCI algorithm (Spirtes et al., 2000).

We define the algorithm PC-MIMBUILD[8] as the algorithm that takes as input a measurement model satisfying the assumption of purity mentioned above and a covariance matrix, and returns the Markov equivalence class of the structural model among the latents in the measurement model according to the PC algorithm. A FCI-MIMBUILD algorithm is defined analogously. In the limit of infinite data, it follows from the preceding and from the consistency of PC and FCI algorithms (Spirtes et al., 2000) that

Theorem 20 *Given a covariance matrix Σ assumed to be generated from a linear latent variable model G, and G_{out} the output of* BUILDPURECLUSTERS *given Σ, the output of* PC-MIMBUILD *or* FCI-MIMBUILD *given (Σ, G_{out}) returns the correct Markov equivalence class of the latents in G corresponding to latents in G_{out} according to the mapping implicit in* BUILDPURECLUSTERS.

For most common families of probabilities distributions (e.g., multivariate Gaussians) the sample covariance matrix is a consistent estimator of the population covariance matrix. This fact, combined with Theorem 20, shows we have a point-wise consistent algorithm for learning a latent variable model with a pure measurement model, up to the measurement equivalence class described in Theorem 15 and the Markov equivalence class of the structural model.

6.2 Score-Based Search

Score-based approaches for learning the structure of Bayesian networks, such as GES (Meek, 1997; Chickering, 2002) are usually more accurate than PC or FCI when there are no omitted common causes, or in other terms, when the set of recorded variables is causally sufficient. We know of

7. One way to test if the rank of a covariance matrix in Gaussian models is at most q is to fit a factor analysis model with q latents and assess its significance.
8. MIM stands for "multiple indicator model", a term in structural equation model literature describing latent variable models with multiple measures per latent.

no consistent scoring function for linear latent variable models that can be easily computed. This might not be a practical issue, since any structural model with a fixed measurement model generated by BUILDPURECLUSTERS has an unique maximum likelihood estimator, up to the scale and sign of the latents. That is, the set of maximum likelihood estimators is a single point, instead of a complicated surface. This sidesteps most of the problems concerning finding the proper complexity penalization for a candidate model (Spirtes et al., 2000).

We suggest using the Bayesian Information Criterion (BIC) function as a score function. Using BIC with STRUCTURAL EM (Friedman, 1998) and GES results in a computationally efficient way of learning structural models, where the measurement model is fixed and GES is restricted to modify edges among latents only. Assuming a Gaussian distribution, the first step of our STRUCTURAL EM implementation uses a fully connected structural model in order to estimate the first expected latent covariance matrix. That is followed by a GES search. We call this algorithm GES-MIMBUILD and use it as the structural model search component in all of the studies of simulated and empirical data that follow.

7. Simulation Studies

In the following simulation studies, we draw samples of three different sizes from 9 different latent variable models. We compare our algorithm against exploratory factor analysis and the DAG hill-climbing algorithm FINDHIDDEN (Elidan et al., 2000), and measure the success of each on the following discovery tasks:

DP1. Discover the number of latents in G.

DP2. Discover which observed variables measure each latent G.

DP3. Discover as many features as possible about the causal relationships among the latents in G.

Since factor analysis addresses only tasks DP1 and DP2, we compare it directly to BUILDPURECLUSTERS on DP1 and DP2. For DP3, we use our procedure and factor analysis to compute measurement models, then discover as much about the features of the structural model among the latents as possible by applying GES-MIMBUILD to the measurement models output by BPC and factor analysis.

We hypothesized that three features of the problem would affect the performance of the algorithms compared: sample size; the complexity of the structural model; and, the complexity and level of impurity in the generating measurement model. We use three different sample sizes for each study: 200, 1,000, and 10,000. We constructed nine generating latent variable graphs by using all combinations of the three structural models and three measurement models in Figure 12. For structural model SM3, the respective measurement models are augmented accordingly.

MM1 is a pure measurement model with three indicators per latent. MM2 has five indicators per latent, one of which is impure because its error is correlated with another indicator, and another because it measures two latents directly. MM3 involves six indicators per latent, half of which are impure.

SM1 entails one unconditional independence among the latents: L_1 is independent L_3. SM2 entails one first order conditional independence: $L_1 \perp L_3 | L_2$, and SM3 entails one first order conditional independence: $L_2 \perp L_3 | L_1$, and one second order conditional independence relation: $L_1 \perp L_4 | \{L_2, L_3\}$.

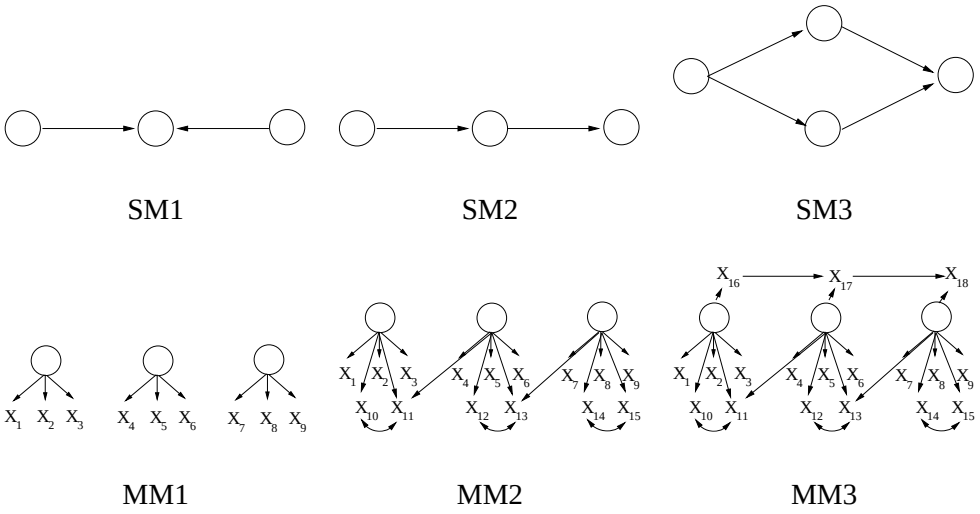


Figure 12: The Structural and Measurement models used in our simulation studies.

Thus the statistical complexity of the structural models increases from SM1 to SM3 and the impurity of measurement models increases from MM1 to MM3.

For each generating latent variable graph, we used the Tetrad IV program[9] with the following procedure to draw 10 multivariate normal samples of size 200, 10 at size 1,000, and 10 at size 10,000.

1. Pick coefficients for each edge in the model randomly from the interval $[-1.5, -0.5] \cup [0.5, 1.5]$.
2. Pick variances for the exogenous nodes (i.e., latents without parents and error nodes) from the interval $[1, 3]$.
3. Draw one pseudo-random sample of size N.

This choice of parameter values for simulations implies that, on average, half of the variance of the indicators of an exogenous latent is due to the error term, making the problem of structure learning more particularly difficult for at least some clusters.

We used three algorithms in our studies:

1. BPC: BUILDPURECLUSTERS + GES-MIMBUILD
2. FA: Factor Analysis + GES-MIMBUILD
3. FH: FINDHIDDEN, using the same sort of hill-climbing procedure used by Elidan et al. (2000)

BPC is the implementation of BUILDPURECLUSTERS and GES-MIMBUILD described in Appendix A. FA involves combining standard factor analysis to find the measurement model with GES-MIMBUILD to find the structural model. For standard factor analysis, we used `factanal`

9. Available at `http://www.phil.cmu.edu/projects/tetrad`.

from R 1.9 with the oblique rotation promax. FA and variations are still widely used and are perhaps the most popular approach to latent variable modeling (Bartholomew et al., 2002). We choose the number of latents by iteratively increasing its number until we get a significant fit above 0.05, or until we have to stop due to numerical instabilities.

Our implementation of FINDHIDDEN follows closely the implementation suggested by Elidan et al. (2000): in that implementation, a candidate latent is introduced as a common parent of the nodes in a dense subgraph of the current graph (such a subgraph is called *semiclique* by Elidan et al.). We implemented the most computational expensive version of FINDHIDDEN, where all semicliques are used to create new candidate graphs, and a full hill-climbing procedure with tabu search is performed to optimize each of them. The score function is BIC. The initial graph is a fully connected DAG among observed variables.[10]

We also added to FINDHIDDEN the prior knowledge that all edges should be directed from latents into observed variables, and we split the search into two main stages: first, only edges into observed variables are modified, while keeping a fully connected structural model. After finding the measurement model, we proceed to learn the structural model using the same type of hill-climbing procedure suggested by Elidan et al. Without these two modifications, FINDHIDDEN results are significantly worse.[11]

In order to compare the output of BPC, FA, and FH on discovery tasks DP1 (finding the correct number of underlying latents) and DP2 (measuring these latents appropriately), we must map the latents discovered by each algorithm to the latents in the generating model. That is, we must define a mapping of the latents in the G_{out} to those in the true graph G.

We do the mapping by first fitting each model by maximum likelihood to obtain estimates for the parameters. For each latent in the output model, we sum the absolute values of the edge coefficients of their observed children, grouping the sum according to their true latent parents. The group with the highest sum will define the label of the output latent. That is, for each latent L_{out} in the output model, the following procedure is performed:

- for all latents $L_1, \ldots, L_k$ in the true model, let $S_i = 0$, $1 \leq i \leq k$
- for every child O that measures L_{out} in the output model with edge coefficient λ_{LO}, such that O has a single parent L_i in the true model, increase S_i by $|\lambda_{LO}|$
- let M be such that S_M is maximum among $S_1, \ldots, S_k$. Label L_{out} as L_M.

For example, let L_{out} be a latent node in the output graph G_{out}. Suppose S_1 is the sum of the absolute values of the edge coefficients of the children of L_{out} that measure the true latent L_1, and S_2 is the respective sum for the measures of true latent L_2. If $S_2 > S_1$, we rename L_{out} as L_2. If two output latents are mapped to the same true latent, we label only one of them as the true latent by

10. Which is the true graph among observed variables in most simulations. We chose the initialization point to save computational costs of growing an almost fully connected DAG without hidden variables first.
11. Another important modification in our implementation was in the STRUCTURAL EM implementation: to escape out of bad local minima within STRUCTURAL EM, we do the following whenever the algorithm arrives in a local minimum: we apply the same search operators, but using the *true BIC score* evaluation instead of the STRUCTURAL EM-BIC score, which is a lower bound on the regular BIC score. This was also crucial to get better results with FINDHIDDEN, but considerably slowed down the algorithm, since computing the true score is computationally expensive and requires an evaluation of the whole model.

choosing the one that corresponds to the highest sum of absolute loadings. The other one remains unmapped and receives an arbitrary label.

We compute the following scores for the output model G_{out} from each algorithm,[12] where the true graph is labelled G:

- **latent omission**, the number of latents in G that do not appear in G_{out} divided by the total number of true latents in G;
- **latent commission**, the number of latents in G_{out} that could not be mapped to a latent in G divided by the total number of true latents in G;
- **mismeasurement**, the number of observed variables in G_{out} that are measuring at least one wrong latent divided by the number of observed variables in G;

To be generous to factor analysis, we considered only latents with at least three indicators. Even with this help, we still found several cases in which latent commission errors were more than 100%. We eliminated from FINDHIDDEN any latent that ended up with no observed children.

Table 3 evaluates all three procedures on the first two discovery tasks: DP1 and DP2. Each number is the average error across 10 trials with standard deviations in parentheses for sample sizes of 200, 1000, 10,000. Over all conditions, FA has very low rates of latent omission, but very high rates of latent commission. In particular, FA is very sensitive to the purity of the generating measurement model. With MM2, the rate of latent commission for FA was moderate; with MM3 it was abysmal. Because indicators are given too many latent parents in FA, many indicators are removed during purification, resulting in high indicator omission errors.

BPC does reasonably well on all measures in Tables 3 at all sample sizes and for all generating models. Our implementation of FINDHIDDEN also does well in most cases, but has issues with SM1.[13]

In the final piece of the simulation study, we applied the best causal model search algorithm we know of, GES, modified for this purpose as GES-MIMBUILD, to the measurement models output by BPC and FA. We evaluate FH both by 1. using its default structural model, which is obtained by a standard hill-climbing with tabu search, and by 2. fixing its measurement model and applying GES to re-learn the corresponding structural model.

If the output measurement model has no errors of latent omission or commission, then scoring the result of the structural model search is fairly easy. The GES-MIMBUILD search outputs an equivalence class, with certain adjacencies unoriented and certain adjacencies oriented. If there is an adjacency of any sort between two latents in the output, but no such adjacency in the true graph, then we have an error of edge commission. If there is no adjacency of any sort between two latents in the output, but there is an edge in the true graph, then we have an error of edge omission. For orientation, if there is an oriented edge in the output that is not oriented in the equivalence class for

12. Other types of errors, such as missing indicators that could have been preserved (in BPC) or adding edges among indicators when they should not exist (as in FINDHIDDEN) are not directly comparable and not as important with respect to the task of finding latents and causal relations among latents, and therefore not considered in this simulation study.

13. One possible explanation for the difficulties with SM1 is the fact that, in the intermediate stages of the algorithm, there will be paths connecting $\{X_1, X_2, X_3\}$ and $\{X_7, X_8, X_9\}$ due to latent variables, but such paths that have to amount to zero correlation in order to reproduce the marginal covariance matrix. This might be difficult to obtain with single edge modifications, considering that introducing an edge might cancel some correlations but increase others.

Evaluation of output measurement models									
	Latent omission			**Latent commission**			**Mismeasurements**		
Sample	**BPC**	**FA**	**FH**	**BPC**	**FA**	**FH**	**BPC**	**FA**	**FH**
$SM_1 + MM_1$									
200	0.10(.2)	0.00(.0)	0.50(.3)	0.00(.0)	0.00(.0)	0.00(.0)	0.01(.0)	0.41(.3)	0.52(.3)
1000	0.17(.2)	0.00(.0)	0.17(.3)	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.19(.2)	0.18(.3)
10000	0.07(.1)	0.00(.0)	0.23(.2)	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.14(.2)	0.23(.2)
$SM_1 + MM_2$									
200	0.00(.0)	0.03(.1)	0.27(.3)	0.03(.1)	0.77(.2)	0.00(.0)	0.01(.0)	0.92(.1)	0.47(.3)
1000	0.00(.0)	0.00(.0)	0.17(.2)	0.00(.0)	0.47(.2)	0.07(.1)	0.00(.0)	0.59(.1)	0.27(.3)
10000	0.00(.0)	0.00(.0)	0.27(.3)	0.03(.1)	0.33(.3)	0.07(.1)	0.02(.1)	0.55(.2)	0.33(.3)
$SM_1 + MM_3$									
200	0.00(.0)	0.00(.0)	0.10(.2)	0.07(.1)	1.13(.3)	0.07(.1)	0.03(.1)	0.90(.1)	0.36(.3)
1000	0.00(.0)	0.00(.0)	0.07(.1)	0.07(.1)	0.87(.3)	0.00(.0)	0.03(.1)	0.72(.1)	0.15(.2)
10000	0.03(.1)	0.00(.0)	0.23(.3)	0.00(.0)	0.70(.3)	0.03(.1)	0.00(.0)	0.60(.2)	0.30(.3)
$SM_2 + MM_1$									
200	0.10(.2)	0.00(.0)	0.27(.3)	0.00(.0)	0.00(.0)	0.00(.0)	0.06(.1)	0.43(.2)	0.28(.3)
1000	0.03(.1)	0.00(.0)	0.17(.3)	0.00(.0)	0.00(.0)	0.00(.0)	0.02(.1)	0.23(.2)	0.19(.3)
10000	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.11(.1)	0.00(.0)
$SM_2 + MM_2$									
200	0.03(.1)	0.00(.0)	0.17(.2)	0.07(.1)	0.80(.3)	0.00(.0)	0.06(.1)	0.85(.1)	0.32(.2)
1000	0.00(.0)	0.00(.0)	0.03(.1)	0.00(.0)	0.53(.3)	0.07(.1)	0.00(.0)	0.68(.1)	0.24(.2)
10000	0.00(.0)	0.00(.0)	0.03(.1)	0.00(.0)	0.27(.3)	0.03(.1)	0.00(.0)	0.53(.2)	0.08(.1)
$SM_2 + MM_3$									
200	0.00(.0)	0.03(.1)	0.03(.1)	0.00(.0)	1.13(.3)	0.07(.1)	0.01(.0)	0.91(.1)	0.29(.2)
1000	0.00(.0)	0.00(.0)	0.07(.1)	0.00(.0)	0.73(.3)	0.07(.1)	0.00(.0)	0.71(.2)	0.15(.1)
10000	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.97(.3)	0.03(.1)	0.00(.0)	0.78(.2)	0.03(.1)
$SM_3 + MM_1$									
200	0.12(.2)	0.02(.1)	0.40(.2)	0.00(.0)	0.05(.1)	0.00(.0)	0.05(.1)	0.66(.2)	0.43(.2)
1000	0.10(.2)	0.02(.1)	0.02(.1)	0.00(.0)	0.02(.1)	0.00(.0)	0.01(.0)	0.30(.2)	0.03(.1)
10000	0.05(.1)	0.00(.0)	0.05(.1)	0.00(.0)	0.00(.0)	0.00(.0)	0.00(.0)	0.21(.1)	0.07(.1)
$SM_3 + MM_2$									
200	0.02(.1)	0.05(.2)	0.10(.1)	0.10(.2)	0.62(.1)	0.02(.1)	0.03(.1)	0.89(.1)	0.31(.2)
1000	0.02(.1)	0.02(.1)	0.02(.1)	0.02(.1)	0.38(.2)	0.05(.1)	0.01(.0)	0.68(.2)	0.15(.1)
10000	0.00(.0)	0.05(.1)	0.05(.2)	0.00(.0)	0.35(.2)	0.02(.1)	0.00(.0)	0.72(.2)	0.15(.2)
$SM_3 + MM_3$									
200	0.02(.1)	0.02(.1)	0.02(.1)	0.05(.1)	0.98(.3)	0.02(.1)	0.04(.1)	0.91(.1)	0.24(.2)
1000	0.02(.1)	0.08(.2)	0.00(.0)	0.00(.0)	0.72(.3)	0.08(.1)	0.00(.0)	0.77(.1)	0.08(.1)
10000	0.00(.0)	0.08(.1)	0.00(.0)	0.00(.0)	0.60(.3)	0.05(.2)	0.00(.0)	0.70(.2)	0.04(.0)

Table 3: Results obtained with BUILDPURECLUSTERS (BPC), factor analysis (FA) and FindHidden (FH) for the problem of learning measurement models. Each number is an average over 10 trials, with standard deviation in parenthesis.

the true structural model, then we have an error of orientation commission. If there is an unoriented edge in the output which is oriented in the equivalence class for the true model, we have an error of orientation omission.

Evaluation of output structural models								
	Edge omission				Edge commission			
Sample	**BPC**	**FA**	**FH**	**FHG**	**BPC**	**FA**	**FH**	**FHG**
$SM_1 + MM_1$								
200	0.05 – 09	0.05 – 09	0.00 – 10	0.00 – 10	0.10 – 09	0.30 – 07	0.00 – 10	0.10 – 09
1000	0.05 – 09	0.10 – 08	0.00 – 10	0.00 – 10	0.20 – 08	0.30 – 07	0.60 – 04	0.10 – 09
10000	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10	0.30 – 07	0.00 – 10
$SM_1 + MM_2$								
200	0.00 – 10	0.15 – 07	0.00 – 10	0.00 – 10	0.00 – 10	0.40 – 06	0.40 – 06	0.10 – 09
1000	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10	0.10 – 09	0.40 – 06	0.40 – 06	0.00 – 10
10000	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10	0.20 – 08	0.50 – 05	0.50 – 05	0.10 – 09
$SM_1 + MM_3$								
200	0.00 – 10	0.25 – 05	0.00 – 10	0.05 – 09	0.20 – 08	0.70 – 03	0.50 – 05	0.30 – 07
1000	0.00 – 10	0.15 – 07	0.00 – 10	0.00 – 10	0.10 – 09	0.70 – 03	0.60 – 04	0.10 – 09
10000	0.00 – 10	0.05 – 09	0.05 – 09	0.00 – 10	0.00 – 10	0.40 – 06	0.50 – 05	0.10 – 09
$SM_2 + MM_1$								
200	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10	0.20 – 08	0.30 – 07	0.00 – 10	0.10 – 09
1000	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10	0.00 – 10	0.30 – 07	0.00 – 10	0.00 – 10
10000	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10	0.20 – 08	0.30 – 07	0.00 – 10	0.20 – 08
$SM_2 + MM_2$								
200	0.00 – 10	0.15 – 07	0.00 – 10	0.00 – 10	0.40 – 06	0.30 – 07	0.00 – 10	0.00 – 10
1000	0.00 – 10	0.10 – 09	0.05 – 09	0.05 – 09	0.10 – 09	0.60 – 04	0.10 – 09	0.20 – 08
10000	0.00 – 10	0.05 – 09	0.05 – 09	0.00 – 10	0.10 – 09	0.70 – 03	0.10 – 09	0.20 – 08
$SM_2 + MM_3$								
200	0.00 – 10	0.15 – 07	0.00 – 10	0.05 – 09	0.20 – 08	0.70 – 03	0.10 – 09	0.20 – 08
1000	0.00 – 10	0.15 – 07	0.00 – 10	0.00 – 10	0.20 – 08	0.40 – 06	0.00 – 10	0.30 – 07
10000	0.00 – 10	0.10 – 08	0.00 – 10	0.00 – 10	0.00 – 10	0.50 – 05	0.00 – 10	0.00 – 10
$SM_3 + MM_1$								
200	0.12 – 05	0.12 – 06	0.05 – 08	0.00 – 10	0.20 – 06	0.20 – 06	0.00 – 10	0.00 – 10
1000	0.05 – 08	0.08 – 08	0.10 – 06	0.00 – 10	0.15 – 08	0.10 – 08	0.55 – 03	0.20 – 07
10000	0.05 – 08	0.15 – 04	0.05 – 08	0.02 – 09	0.15 – 08	0.15 – 08	0.50 – 03	0.15 – 08
$SM_3 + MM_2$								
200	0.02 – 09	0.28 – 03	0.15 – 06	0.02 – 09	0.55 – 03	0.55 – 02	0.20 – 06	0.10 – 08
1000	0.00 – 10	0.12 – 07	0.08 – 07	0.00 – 10	0.25 – 07	0.75 – 02	0.60 – 02	0.15 – 08
10000	0.00 – 10	0.00 – 10	0.02 – 09	0.02 – 09	0.10 – 08	0.80 – 02	0.65 – 01	0.20 – 07
$SM_3 + MM_3$								
200	0.02 – 09	0.32 – 02	0.20 – 03	0.10 – 06	0.40 – 05	0.50 – 02	0.45 – 03	0.20 – 07
1000	0.08 – 07	0.02 – 09	0.10 – 07	0.05 – 08	0.30 – 06	0.65 – 02	0.45 – 04	0.25 – 06
10000	0.00 – 10	0.05 – 08	0.02 – 09	0.00 – 10	0.15 – 07	0.65 – 03	0.70 – 01	0.10 – 08

Table 4: Results obtained with the application of GES-MIMBUILD to the output of BUILDPURECLUSTERS and factor analysis, plus FINDHIDDEN and FINDHIDDEN + GES-MIMBUILD results, with an indication of the number of perfect solutions over these trials.

If the output measurement model has any errors of latent commission, then we simply leave out the excess latents in the measurement model given to GES-MIMBUILD. This helps FA primarily, as it was the only procedure of the three that had high errors of latent commission.

Evaluation of output structural models								
	Orientation omission				Orientation commission			
Sample	**BPC**	**FA**	**FH**	**FHG**	**BPC**	**FA**	**FH**	**FHG**
$SM_1 + MM_1$								
200	0.10 – 09	0.15 – 08	0.10 – 09	0.10 – 09	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
1000	0.20 – 08	0.00 – 10	0.60 – 04	0.10 – 09	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10
10000	0.00 – 10	0.00 – 10	0.30 – 07	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
$SM_1 + MM_2$								
200	0.00 – 10	0.20 – 07	0.40 – 06	0.10 – 09	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10
1000	0.10 – 09	0.20 – 07	0.40 – 06	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
10000	0.20 – 08	0.25 – 05	0.50 – 05	0.10 – 09	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
$SM_1 + MM_3$								
200	0.20 – 08	0.40 – 04	0.60 – 04	0.20 – 08	0.00 – 10	0.05 – 09	0.00 – 10	0.05 – 09
1000	0.10 – 09	0.10 – 09	0.70 – 03	0.10 – 09	0.00 – 10	0.10 – 08	0.00 – 10	0.00 – 10
10000	0.00 – 10	0.30 – 06	0.50 – 05	0.10 – 09	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
$SM_2 + MM_1$								
200	– – –	– – –	– – –	– – –	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
1000	– – –	– – –	– – –	– – –	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
10000	– – –	– – –	– – –	– – –	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
$SM_2 + MM_2$								
200	– – –	– – –	– – –	– – –	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
1000	– – –	– – –	– – –	– – –	0.00 – 10	0.10 – 09	0.00 – 10	0.00 – 10
10000	– – –	– – –	– – –	– – –	0.00 – 10	0.10 – 09	0.05 – 09	0.00 – 10
$SM_2 + MM_3$								
200	– – –	– – –	– – –	– – –	0.00 – 10	0.10 – 08	0.00 – 10	0.00 – 10
1000	– – –	– – –	– – –	– – –	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10
10000	– – –	– – –	– – –	– – –	0.00 – 10	0.05 – 09	0.00 – 10	0.00 – 10
$SM_3 + MM_1$								
200	0.15 – 08	0.00 – 10	0.00 – 10	0.00 – 10	0.22 – 07	0.35 – 06	0.10 – 09	0.00 – 10
1000	0.10 – 09	0.00 – 10	0.65 – 03	0.10 – 09	0.10 – 09	0.00 – 10	0.04 – 09	0.00 – 10
10000	0.05 – 09	0.00 – 10	0.65 – 03	0.05 – 09	0.04 – 09	0.00 – 10	0.04 – 09	0.04 – 09
$SM_3 + MM_2$								
200	0.50 – 05	0.30 – 06	0.20 – 07	0.10 – 09	0.08 – 09	0.16 – 07	0.08 – 09	0.08 – 09
1000	0.30 – 07	0.45 – 04	0.65 – 03	0.30 – 07	0.00 – 10	0.05 – 09	0.11 – 08	0.05 – 09
10000	0.20 – 08	0.40 – 06	0.85 – 01	0.25 – 07	0.00 – 10	0.00 – 10	0.00 – 10	0.00 – 10
$SM_3 + MM_3$								
200	0.50 – 04	0.15 – 08	0.85 – 01	0.35 – 05	0.19 – 06	0.14 – 08	0.20 – 07	0.48 – 02
1000	0.20 – 07	0.35 – 05	0.50 – 04	0.05 – 09	0.15 – 07	0.02 – 09	0.04 – 09	0.11 – 08
10000	0.00 – 10	0.35 – 05	0.85 – 01	0.10 – 09	0.00 – 10	0.00 – 10	0.04 – 09	0.00 – 10

Table 5: Results obtained with the application of GES-MIMBUILD to the output of BUILDPURECLUSTERS and factor analysis, plus FINDHIDDEN and FINDHIDDEN + GES-MIMBUILD results, with an indication of the number of perfect solutions over these trials.

If the output measurement model has errors of latent omission, then we compare the marginal involving the latents in the output model for the true structural model graph to the output structural model equivalence class. For each of the structural models we selected, SM1, SM2, and SM3, all marginals can be represented faithfully as DAGs. Our measure of successful causal discovery,

therefore, for a measurement model involving a small subset of the latents in the true graph is very lenient. For example, if the generating model was SM3, which involves four latents, but the output measurement model involved only two of these latents, then a perfect search result in this case would amount to finding that the two latents are associated.

In summary then, our measures for assessing the ability of these algorithms to correctly discover at least features of the causal relationships among the latents are as follows:

- **edge omission (EO)**, the number of edges in the structural model of G that do not appear in G_{out} divided by the possible number of edge omissions (2 in SM_1 and SM_2, and 4 in SM_3, i.e., the number of edges in the respective structural models);

- **edge commission (EC)**, the number of edges in the structural model of G_{out} that do not exist in G divided by the possible number of edge commissions (only 1 in SM_1 and SM_2, and 2 in SM_3);

- **orientation omission (OO)**, the number of arrows in the structural model of G that do not appear in G_{out} divided by the possible number of orientation omissions in G (2 in SM_1 and SM_3, 0 in SM_2);

- **orientation commission (OC)**, the number of arrows in the structural model of G_{out} that do not exist in G divided by the number of edges in the structural model of G_{out};

Tables 4 and 5 summarize the results. Along with each average we provide the number of trials where no errors of a specific type were made.

Factor analysis is particularly flawed. This is because FA infers so many latents, which leads to spurious dependence paths among the latents we scored. The default FINDHIDDEN is also sub-optimal in these small models, due to limitations in the hill-climbing procedure compared to GES: SM3 has a high proportion of "compelled" edges (Chickering, 2002), i.e., edges that are oriented in the pattern corresponding to the Markov equivalence class, which makes it harder for an algorithm that searches among DAGs instead of equivalence classes. Therefore, we included in Tables 4 and 5 a variation of FINDHIDDEN, labeled FHG, where we fix the measurement model given by FINDHIDDEN and learn the structural model using GES. Results are not significantly different from BPC + GES, except at sample size of 200, where FINDHIDDEN has a tendency to miss latents, inflating its performance in the structural model evaluation (since with fewer latents there is less chance of committing mistakes).

Figure 13 provides a summary evaluation of all algorithms, BPC, FA and FHG with respect to the number of perfect structural models obtained for each graphical structure (from 0 to 10). This includes not only getting the exact number of latents, but also the correct Markov equivalence class defined in the true model. Factor analysis is competitive when the true model is pure, but is completely ineffective otherwise. For models based on structural model SM3, FA does not get any fully correct structure when the measurement model is impure. Moreover, it is clear that while learning the measurement model can be reasonably performed by BUILDPURECLUSTERS and FINDHIDDEN with sample sizes of 200, learning the structural model is not an easy task unless more data is available.

In summary, factor analysis provides little useful information out of the given datasets that were not generated by pure models. In contrast, the combination of BUILDPURECLUSTERS and GES-

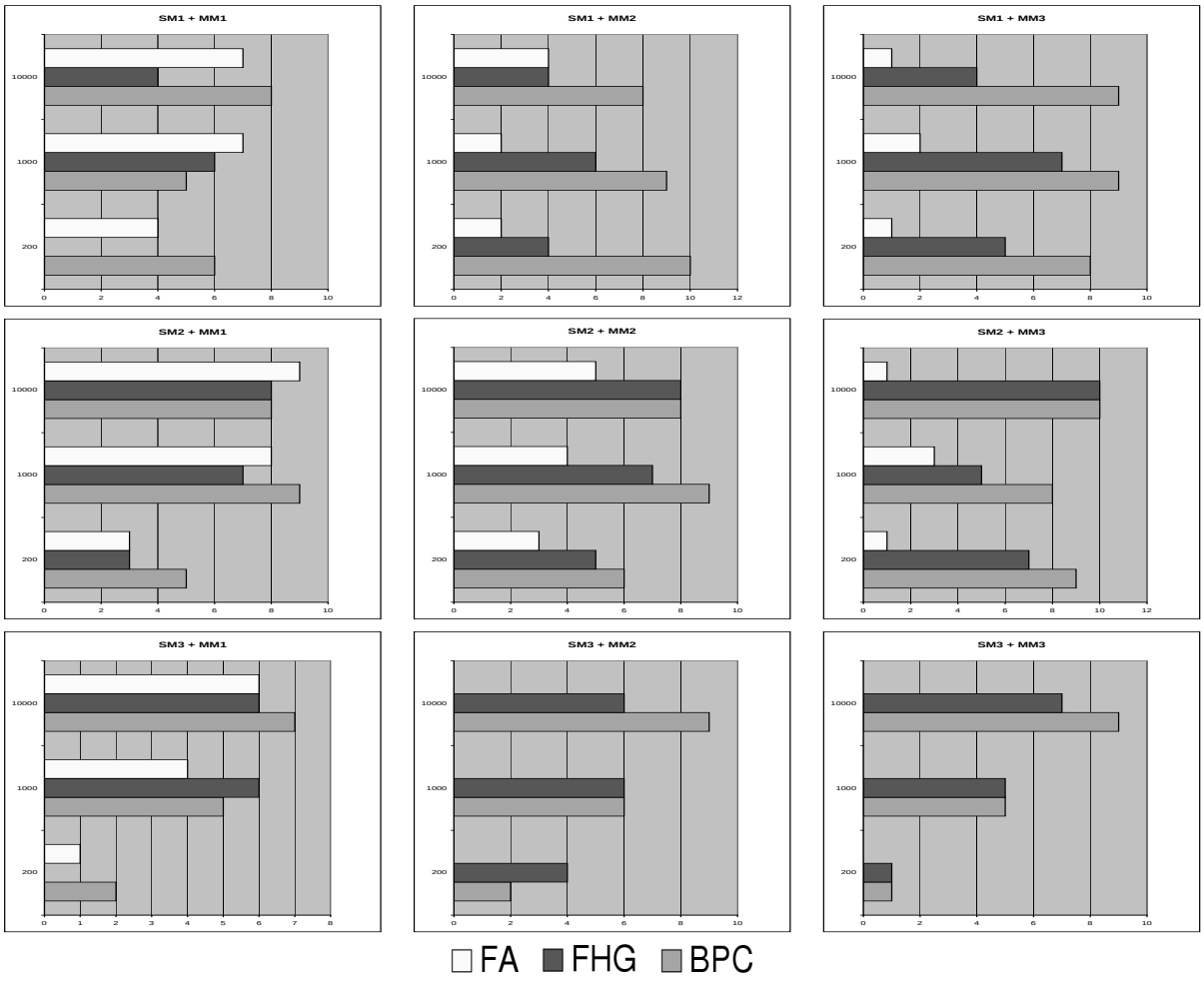


Figure 13: A comparison of the number of perfect solutions in all synthetic data sets.

MIMBUILD largely succeeds. FINDHIDDEN (with GES, i.e., FHG) has generally good results, although it behaves erractly with SM1.[14]

8. Real Data Applications

We now briefly present the results for two real data sets. Data collected from such domains may pose significant problems for exploratory data analysis since sample sizes are usually small and noisy, nevertheless they have a very useful property for our empirical evaluation. In particular, data obtained by questionnaires are designed to target specific latent factors (such as "stress", "job satisfaction", and so on) and a theoretical measurement model is developed by experts in the area to measure the desired latent variables. Very generally, experts are more confident about their choice of measures than about the structural model. Such data thus provide a basis for comparison with the output of our algorithm. The chance that various observed variables are not pure measures of their

14. This can probably be improved by adopting other schema of search initialization and extra heuristics for escaping local minima. However, it can also be a much slower algorithm than BPC, as discussed before.

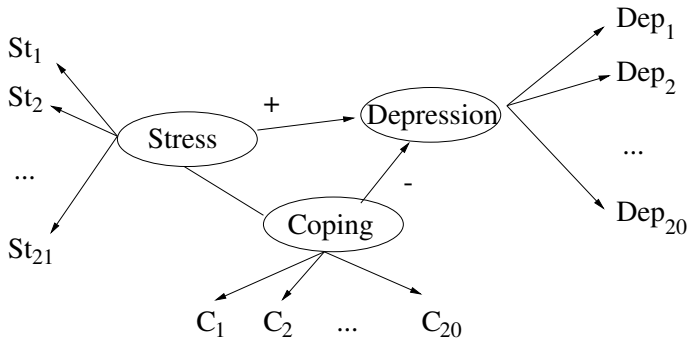


Figure 14: A theoretical model for the interaction of religious coping, stress and depression. The signs on the edges depicts the theoretical signs of the corresponding effects.

theoretical latents is high. Measures are usually discrete, but often ordinal with a Likert-scale that can be treated as normally distributed measures with little loss (Bollen, 1989). In the examples, we compare our procedures with models produced by domain researchers.

8.1 Stress Religious Coping and Depression

Bongjae Lee from the University of Pittsburgh conducted a study of religious/spiritual coping and stress in graduate students. In December of 2003, 127 students answered a questionnaire intended to measure three main factors: stress (measured with 21 items), depression (measured with 20 items) and religious/spiritual coping (measured with 20 items). The full questionnaire is given by Silva and Scheines (2004). Lee's model is shown in Figure 14.

This model fails a chi-square test: p = 0. The measurement model produced by BUILDPURECLUSTERS is shown in Figure 15(a). Note that the variables selected automatically are proper subsets of Lee's substantive clustering. The full model automatically produced with GESMIMBUILD with the prior knowledge that STRESS is not an effect of other latent variables is given in Figure 15(b). This model passes a chi square test, p = 0.28, even though the BPC algorithm itself does not try to directly maximize the fit of the algorithm.

Our FINDHIDDEN implementation took a couple of days to execture and did not perform produce a reasonable output if the theoretical model should be taken as the gold standard: five latents were found to have 20 indicators altogether, but they have no correspondence to the theoretical clustering. This is not unexpected, since the sample size is very small and FINDHIDDEN tries to create a model that includes all 61 variables. BUILDPURECLUSTERS can be seen as a way of doing feature selection by focusing on the easier, simpler pure models.

8.2 Test Anxiety

A survey of test anxiety indicators was administered to 335 grade 12 male students in British Columbia. The survey consisted of 20 measures on symptoms of anxiety under test conditions. The covariance matrix as well as a description of the variables is given by Bartholomew et al. (2002).[15]

15. The data are available online at http://multilevel.ioe.ac.uk/team/aimdss.html.

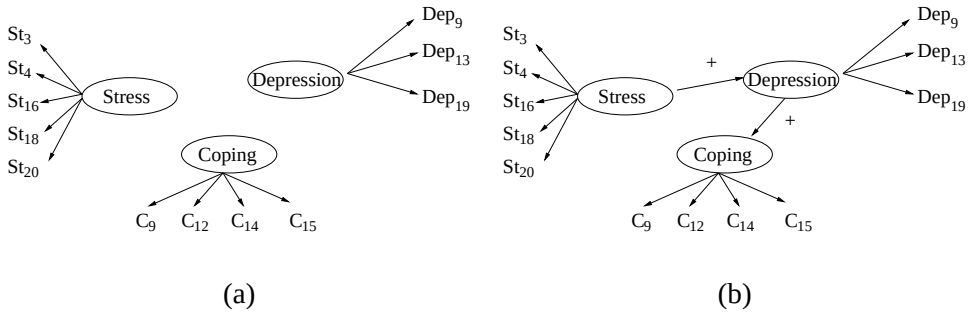


Figure 15: The output of BPC and GES-MIMBUILD for the coping study.

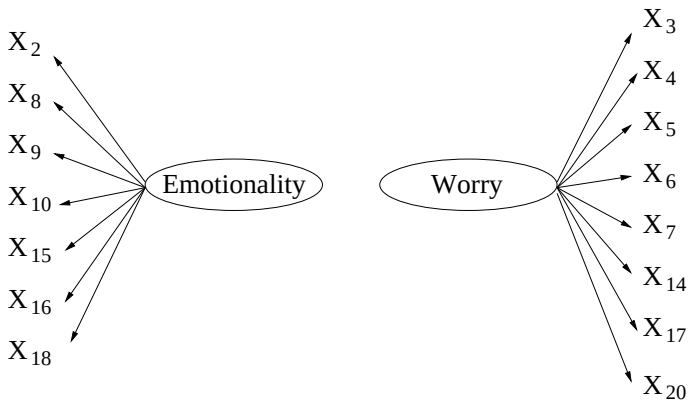


Figure 16: A theoretical model for psychological factors of test anxiety.

Using exploratory factor analysis, Bartholomew et al. concluded that two latent common causes underly the variables in this data set, agreeing with previous studies. The original study identified items $\{x_2, x_8, x_9, x_{10}, x_{15}, x_{16}, x_{18}\}$ as indicators of an "emotionality" latent factor (this includes physiological symptoms such as jittery and faster heart beatting), and items $\{x_3, x_4, x_5, x_6, x_7, x_{14}, x_{17}, x_{20}\}$ as indicators of a more psychological type of anxiety labeled "worry" by Bartholomew et al. No further description is given about the remaining five variables. Bartholomew et al.'s factor analysis with oblique rotation roughly matches this model. Bartholomew et al.'s exploratory factor analysis model for a subset of the variables is shown in Figure 16. This model is not intended to be pure. Instead, the figure represents which of the two latents is more "strongly" connected to each indicator. The measurement model itself is not constrained. Trying to fit this model as a pure model (i.e., using the graph in Figure 16 instead of a two-factor multivariate Gaussian model with a fully connected measurement model) gives a p-value of zero according to a chi-square test.

BPC provides the measurement model given in 17(a).[16] The labels in the latents were given to us and should be seen as our particular interpretation. Applying GES-MIMBUILD to the this measurement model results in the model shown in Figure 17(b). The model passes a chi-square

16. We allowed a latent with less than three indicators. It might correspond to more than one latent in the true model.

test handily, p = 0.47, even though we used constraint-satisfaction techniques that did not try to maximize the fitness of the model directly. To summarize, BPC provided a model supported by the data that is very close to a submodel of the theoretical model (variables $X_4, X_{15}, X_{17}, X_{20}$ were removed), except that:

- one of the latents is split in two. To see how this is supported by the data, trying to merge latents "Cares about achieving" and "Self-defeating" will result in a model of p-value of zero;
- variable X_{11} is used, which is not considered by Bartholomew et al.'s model;

What is remarkable in this case is the ability of reconstructing much of the theoretical model without using prior knowledge. The model is very simple, i.e., each indicator measures a single latent, while Bartholomew et al.'s model seems to artificially add edges from all latents into all indicators to get a model that fits the data. Escaping this artificiality is one of the motivations behind variable selection in factor analysis methods, such as the one proposed by Kano and Harada (2000): finding a submodel that is a pure model can provide a better understanding of the causal process being measured than allowing an impure model, whose extra edges might be no more than a patch to account for residual correlation among indicators, without necessarily existing in the true model. Kano and Harada's method, however, requires an initial measurement model to be "purified," while BPC works from scratch.

We applied FINDHIDDEN to this data set, obtaining the model shown in Figure 18(a). To simplify presentation, we removed nodes that were not children of any latent in the output model (e.g., X_3 was not a child of any of the latents, which results on its removal from the picture). Three latents, labeled by us as "Emotionality 1", "Emotionality 2" and "Worry" were generated. Both "Emotionality 1" and "Emotionality 2" seem to be a combination of some of the theoretical "Emotionality" indicators (Figure 16) plus some indicators not included the theoretical model of Figure 16. There are also lots of edges corresponding to impurities for which no equivalence class is known. As discussed in Section 3, these edges might correspond to very different causal mechanisms than they might suggest.

Since 5 of the variables are not present in the theoretical model, it is not so easy to compare this model against the theoretical model. Therefore, we also provide the result that is obtained from FINDHIDDEN when the data contains only the 15 indicators used in Figure 16. The result is the one shown in Figure 18(b), where we adopted the same latent labels used in BPC's output. The result is, surpringly, very different. The model has now a much closer resemblance to BPC's output, supporting the plausability of BPC's output. However, while it seems that BPC is able to find a pure model among all 20 indicators, FINDHIDDEN in this case was able to find an (almost) pure model *only* when variables were properly pre-selected.

9. Generalizations

In many social science studies, latent structure is represented by so called "non-recursive" structure. In graphical terms, the dependency graph is cyclic. Richardson (1996) has developed a consistent constraint based search for cyclic graphical models of linear systems, and our procedures for identifying measurement models can be combined with it to search for such structure.

The procedure we have described here can, however, straightforwardly be generalized to cases with measured variables taking a small finite range of values by treating the discrete variables as

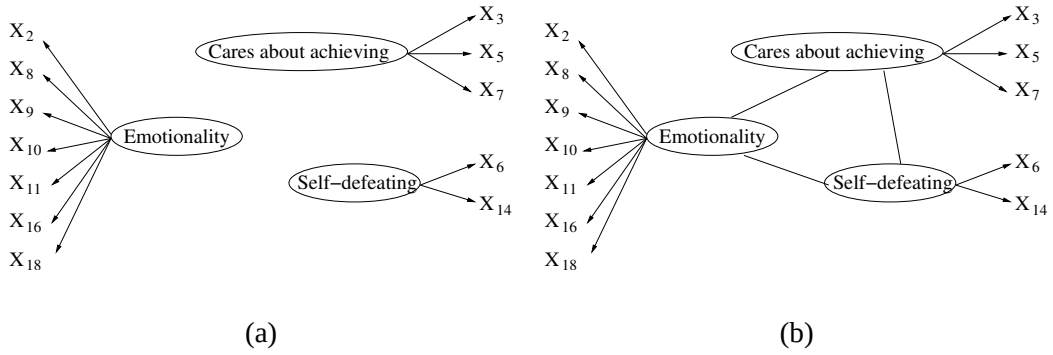


Figure 17: The output of BPC and GES-MIMBUILD for the test anxiety study.

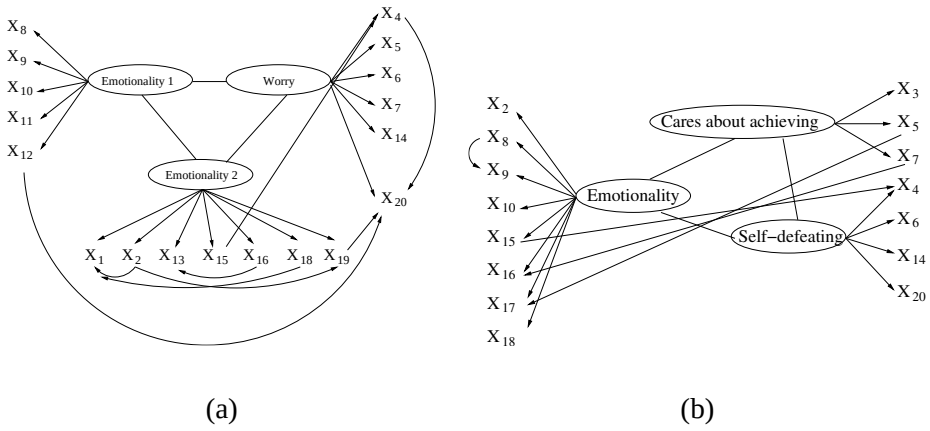


Figure 18: The output of FINDHIDDEN when using all 20 variables (a) and when using only the variables present in the theoretical model (b).

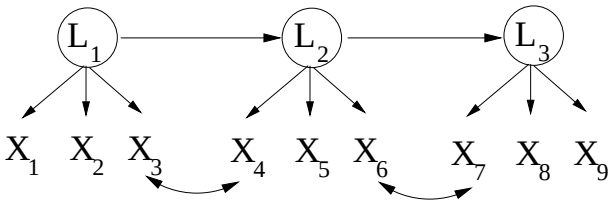


Figure 19: A model with no pure submodel with three indicators per latent.

projections from a Gaussian distribution. These are sometimes called latent trait models in the literature (Bartholomew and Knott, 1999). Much larger sample sizes are required than for linear, Gaussian measured variables.

In previous works (Silva et al., 2003; Silva and Scheines, 2005), we developed an approach to learn measurement models even when the functional relationships among latents are non-linear. In practice, that generality is of limited use because there are at present no consistent search methods available for structures with continuous, non-linear variables. A modified version of BUILDPURECLUSTERS, however, exists for the problem of learning equivalence classes of measurement models for non-linear structural models. Some of the results here developed cannot be carried on to the non-linear case (e.g., rule CS3). Others are substantially modified (Lemma 9). With extra prior knowledge, however, much of the measurement model for non-linear structural models can still be learned from data.

Finally, there are ways of reliably learning some types of impure models using the results discussed in this paper. For instance, only two of the three latents in the model in Figure 19 can be generated by BUILDPURECLUSTERS. A small modification of the algorithm, which would include an equivalence class accounting for some types of impurities, would be able to reconstruct all latents in this example. A more systematic exploration of such extensions will be treated in a future work.

10. Conclusion

This paper introduced a novel algorithm for learning causal structure in linear models which, to the best of our knowledge, presents the first published solution for the problem of learning causal models with latent variables in a principled way where observed conditional independencies are not expected to exist. It has the following properties:

- it was designed to learn multiple indicator models, i.e., models where observed variables are not causes of the hidden variables of interest, but which still encompass a large class of useful models;

- no assumptions about the number of hidden variables and how they are connected to observed variables are needed;

- it is a two-stage algorithm, which first learns equivalence classes of measurement models (i.e., which latents exist and which observed children they have) and, based on a choice of measurement model, returns an equivalence class of causal models among the latents;

- it is provably correct, in the sense that given the assumptions explicitly described in the paper and in the limit of infinite data, all causal claims made by the output graph hold in the population;

- it provides a framework which can be partially extended to cover other types of data (discrete, ordinal) and causal relations (non-linear, non-Gaussian);

Our experiments provide evidence that our procedures can be useful in practice, but there are certainly classes of problems where BUILDPURECLUSTERS will not be of practical value. For instance, learning the causal structure of general blind source separation problems, where measures are usually indicators of most of the latents (i.e., sources) at the same time.

A number of open problems invite further research, including these:

- completeness of the tetrad equivalence class of measurement models: can we identify all the common features of measurement models in the same tetrad equivalence class?

- using the more generic rank constraints of covariance matrices to learn measurement models, possibly identifying the nature of some impure relationships;

- better treatment of discrete variables. Bartholomew and Knott (1999) survey different ways of integrating factor analysis and discrete variables that can be readily adapted, but the computational cost of this procedure is high;

- finding non-linear causal relationships among latent variables given a fixed linear measurement model, and in other families of multivariate continuous distributions besides the Gaussian;

The fundamental point is that common and appealing heuristics (e.g., factor rotation methods) fail when the goal is structure learning with a causal interpretation. In many cases it is preferable to model the relationships of a subset of the given variables than trying to force a bad model over all of them (Kano and Harada, 2000). Better methods are available now, and further improvements will surely come from machine learning research.

Acknowledgments

We thank the anonymous reviewers for their comments, which greatly improved the presentation of this paper. Research for this paper was supported by NASA NCC 2-1377 to the University of West Florida, NASA NRA A2-37143 to CMU and ONR contract N00014-03-01-0516 to the University of West Florida.

Appendix A. BUILDPURECLUSTERS: Full Algorithm and Implementation

We now introduce the complete version of BUILDPURECLUSTERS. This version has additional steps that deal with exceptional, but arguably less relevant, situations. This requires removing additional nodes due to vanishing correlations, as well as merging some clusters. The full algorithm is given in Table 6.

Algorithm BUILDPURECLUSTERS
Input: a covariance matrix Σ

1. $G \leftarrow$FINDPATTERN(Σ).
2. Choose a set of latents in G. Remove all other latents and all observed nodes that are not children of the remaining latents and all clusters of size 1.
3. Remove all nodes that have more than one latent parent in G.
4. For all pairs of nodes linked by an undirected edge, choose one element of each pair to be removed.
5. If for some set of nodes $\{A,B,C\}$, all children of the same latent, there is a fourth node D in G such that $\sigma_{AB}\sigma_{CD} = \sigma_{AC}\sigma_{BD} = \sigma_{AD}\sigma_{BC}$ is *not* true, remove one of these four nodes.
6. For every latent L with at least two children, $\{A,B\}$, if there is some node C in G such that $\sigma_{AC} = 0$ and $\sigma_{BC} \neq 0$, split L into two latents L_1 and L_2, where L_1 becomes the only parent of all children of L that are correlated with C, and L_2 becomes the only parent of all children of L that are not correlated with C;
7. Remove any cluster with exactly 3 variables $\{X_1,X_2,X_3\}$ such that there is no X_4 where all three tetrads in the covariance matrix $\mathbf{X} = \{X_1,X_2,X_3,X_4\}$ hold, all variables of $\mathbf{X}$ are correlated and no partial correlation of a pair of elements of $\mathbf{X}$ is zero conditioned on some observed variable;
8. While there is a pair of clusters with latents L_i and L_j, such that for all subsets $\{A,B,C,D\}$ of the union of the children of L_i, L_j we have $\sigma_{AB}\sigma_{CD} = \sigma_{AC}\sigma_{BD} = \sigma_{AD}\sigma_{BC}$, and no marginal or conditional independencies (where the condition set is of size 1) are observed in this cluster, set $L_i = L_j$ (i.e., merge the clusters);
9. Again, verify all implied tetrad constraints and remove elements accordingly: iterate Steps 6-7-8 until no changes happen;
10. Remove all latents with less than three children, and their respective measures;
11. if G has at least four observed variables, return G. Otherwise, return an empty model.

Table 6: The complete version of BUILDPURECLUSTERS.

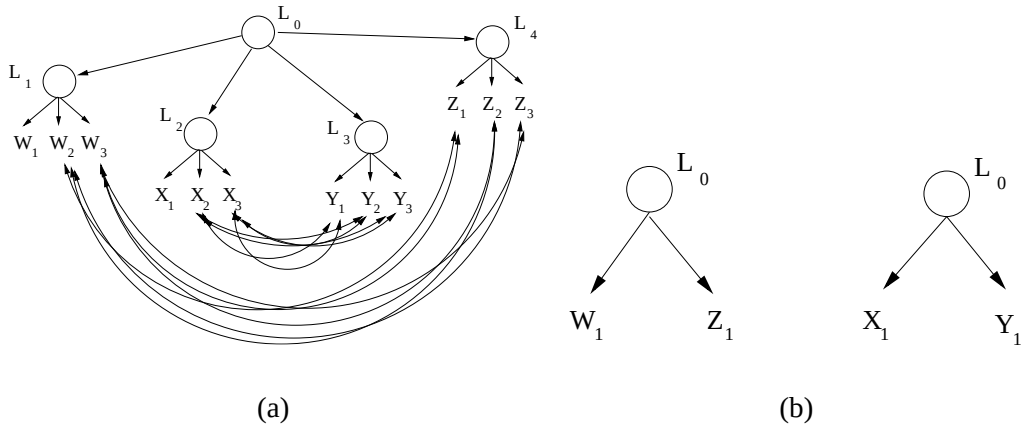


Figure 20: The true graph in (a) will generate at some point a purified measurement pattern as in (b). It is desirable to merge both clusters.

It might be surprising that we merge clusters of variables that we know cannot share a common latent parent in the true graph. However, we are not guaranteed to find a large enough number of pure indicators for each of the original latent parents, and as a consequence only a subset of the true latents will be represented in the measurement pattern. It might be the case that, with respect to the variables present in the output, the observed variables in two different clusters might be directly measuring some ancestor common to all variables in these two clusters. As an illustration, consider the graph in Figure 20(a), where double-directed edges represent independent hidden common causes. Assume any sensible purification procedure will choose to eliminate all elements in $\{W_2, W_3, X_2, X_3, Y_2, Y_3, Z_2, Z_3\}$ because they are directly correlated with a large number of other observed variables (extra edges and nodes not depicted).

Meanwhile, one can verify that all three tetrad constraints hold in the covariance matrix of $\{W_1, X_1, Y_1, Z_1\}$, and therefore there will be no undirected edges connecting pairs of elements in this set in the corresponding measurement pattern. Rule CS1 is able to separate W_1 and X_1 into two different clusters by using $\{W_2, W_3, X_2, X_3\}$ as the support nodes, and analogously the same happens to Y_1 and Z_1, W_1 and Y_1, X_1 and Z_1. However, no test can separate W_1 and Z_1, nor X_1 and Y_1. If we do not merge clusters, we will end up with the graph seen in Figure 20(b) as part of our output pattern. Although this is a valid measurement pattern, and in some situations we might want to output such a model, it is also true that W_1 and Z_1 measure a same latent L_0 (as well as X_1 and Y_1). It would be problematic to learn a structural model with such a measurement model. There is a deterministic relation between the latent measured by W_1 and Z_1, and the latent measured by X_1 and Y_1: they are the same latent! Probability distributions with deterministic relations are not faithful, and that causes problems for learning algorithms.

Finally, we show examples where Steps 6 and 7 of BUILDPURECLUSTERS are necessary. In Figure 21(a) we have a partial view of a latent variable graph, where two of the latents are marginally independent. Suppose that nodes X_4, X_5 and X_6 are correlated to many other measured nodes not in this figure, and therefore are removed by our purification procedure. If we ignore Step 6, the result-

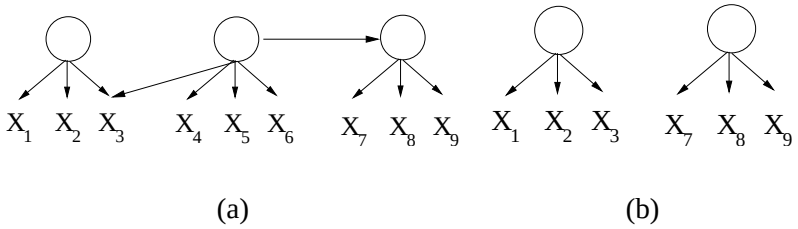


Figure 21: Suppose (a) is our true model. If for some reason we need to remove nodes X_4, X_5 and X_6 from our final pure graph, the result will be as shown in Figure (b), unless we apply Step 6 of BUILDPURECLUSTERS. There are several problems with (b), as explained in the text.

ing pure submodel over $\{X_1, X_2, X_3, X_7, X_8, X_9\}$ will be the one depicted in Figure 21(b) ($\{X_1, X_2\}$ are clustered apart from $\{X_7, X_8, X_9\}$ because of marginal zero correlation, and X_3 is clustered apart from $\{X_7, X_8, X_9\}$ because of CS1 applied to $\{X_3, X_4, X_5\} \times \{X_7, X_8, X_9\}$). However, no linear latent variable model can be parameterized by this graph: if we let the two latents to be correlated, this will imply X_1 and X_7 being correlated. If we make the two latents uncorrelated, X_3 and X_7 will be uncorrelated.

Step 7 exists to avoid rare situations where three observed variables are clustered together and are *pairwise* part of some foursome entailing all three tetrad constraints with no vanishing marginal and partial correlation, but still should be removed because they are not *simultaneously* in such a foursome. They might not be detected by Step 4 if, e.g., all three of them are uncorrelated with all other remaining observed variables.

In the rest of this section, we describe a practical implementation of BUILDPURECLUSTERS. The algorithm is described for a given covariance matrix to simplify the exposition. Since typically one has only a sample covariance matrix, we need a statistical decision procedure. Statistical tests for tetrad constraints are described by Spirtes et al. (2000). Although it is known that in practice constraint-based approaches for learning graphical model structure are outperformed on accuracy by score-based algorithms such as GES (Chickering, 2002), we favor a combination of a constraint-based approach and a score-based approach due mostly to computational efficiency. A smart implementation of constraint-satisfaction algorithms can avoid many statistical shortcomings. If the experimental results are any indication of success, we can claim we provide such an implementation.

We also describe in full detail how particular choices in BUILDPURECLUSTERS (e.g., Step 2, where one has to choose a set of latents from the measurement pattern) are solved in our implementation. We stress that the particularities of the implementation bear no implication on the theoretical results given in this paper: the algorithms remain point-wise consistent. The informativeness of the results (i.e., how much of the true structure is discovered) will vary, but in the particular examples given in this paper, results were quite insensitive to variations of the following implementation.

A.1 Robust Purification

We do avoid a constraint-satisfaction approach for purification. At least for a fixed p-value and using false discovery rates to control for multiplicity of tests, purification by testing tetrad constraints often throws away many more nodes than necessary when the number of variables is relative small, and does not eliminate many impurities when the number of variables is too large. We suggest a robust purification approach as follows.

Suppose we are given a clustering of variables (not necessarily disjoint clusters) and a undirect graph indicating which variables might be ancestors of each other, analogous to the undirect edges generated in FINDPATTERN. We purify this clustering not by testing multiple tetrad constraints, but through a greedy search that eliminates nodes from a linear measurement model that entails tetrad constraints. This is iterated till the current model fits the data according to a chi-square test of significance (Bollen, 1989) and a given acceptance level. Details are given in Table 7.

This implementation is used as a subroutine for a more robust implementation of BUILDPURECLUSTERS described in the next section. However, it can be considerably slow. An alternative is using the approximation derived by Kano and Harada (2000) to rapidly calculate the fitness of a factor analysis model when a variable is removed. Another alternative is a greedy search over the initial measurement model, freeing correlations of pairs of measured variables. Once we found which variables are directly connected, we eliminate some of them till no pair is impure. Details of this particular implementation are given by Silva and Scheines (2004). In our experiments with synthetic data, it did not work as well as the iterative removal of variables described in Table 7. However, we do apply this variation in the last experiment described in Section 6, because it is computationally cheaper. If the model search in ROBUSTPURIFY does not fit the data after we eliminate too many variables (i.e., when we cannot statistically test the model) we just return an empty model.

A.2 Finding a Robust Initial Clustering

The main problem of applying FINDPATTERN directly by using statistical tests of tetrad constraints is the number of false positives: accepting a rule (CS1, CS2, or CS3) as true when it does not hold in the population. One can see that might happen relatively often when there are large groups of observed variables that are pure indicators of some latent: for instance, assume there is a latent L_0 with 10 pure indicators. Consider applying CS1 to a group of six pure indicators of L_0. The first two constraints of CS1 hold in the population, and so assume they are correctly identified by the statistical test. The last constraint, $\sigma_{X_1X_2}\sigma_{Y_1Y_2} \neq \sigma_{X_1Y_2}\sigma_{X_2Y_1}$, should not hold in the population, but will not be rejected by the test with some probability. Since there are $10!/(6!4!) = 210$ ways of CS1 being wrongly applied due to a statistical mistake, we *will* get many false positives in all certainty.

We can highly minimize this problem by separating *groups* of variables instead of pairs. Consider the test DISJOINTGROUP$(X_i, X_j, X_k, Y_a, Y_b, Y_c; \Sigma)$:

- DISJOINTGROUP$(X_i, X_j, X_k, Y_a, Y_b, Y_c; \Sigma) =$ *true* if and only if CS1 returns true for all sets $\{X_1, X_2, X_3, Y_1, Y_2, Y_3\}$, where $\{X_1, X_2, X_3\}$ is a permutation of $\{X_i, X_j, X_k\}$ and $\{Y_1, Y_2, Y_3\}$ is a permutation of $\{Y_a, Y_b, Y_c\}$. Also, we test an extra redundant constraint: for every pair $\{X_1, X_2\} \subset \{X_i, X_j, X_k\}$ and every pair $\{Y_1, Y_2\} \subset \{Y_a, Y_b, Y_c\}$ we also require that $\sigma_{X_1Y_1}\sigma_{X_2Y_2} = \sigma_{X_1Y_2}\sigma_{X_2Y_1}$.

Notice it is much harder to obtain a false positive with DISJOINTGROUP than, say, with CS1 applied to a single pair. This test can be implemented in steps: for instance, if for no four foursome

Algorithm	ROBUSTPURIFY
Inputs:	*Clusters*, a set of subsets of some set **O**;
	C, an undirect graph over **O**;
	Σ, a sample covariance matrix of **O**.

1. Remove all nodes that have appear in more than one set in *Clusters*.
2. For all pairs of nodes that belong to two different sets in *Clusters* and are adjacent in C, remove the one from the largest cluster or the one from the smallest cluster if this has less than three elements.
3. Let G be a graph. For each set $S \in$ *Clusters*, add all nodes in S to G and a new latent as the only common parent of all nodes in S. Create an arbitrary full DAG among latents.
4. For each variable V in G, fit a graph $G'(V)$ obtained from G by removing V. Update G by choosing the graph $G'(V)$ with the smallest chi-square score. If some latent ends up with less than two children, remove it. Iterate till a significance level is achieved.
5. Do mergings if that increases the fitness. Iterate 4 and 5 till no improvement can be done.
6. Eliminate all clusters with less than three variables and return G.

Table 7: A score-based purification.

including X_i and Y_a we have that all tetrad constraints hold, then we do not consider X_i and Y_a in DISJOINGGROUP.

Based on DISJOINTGROUP, we propose here a modification to increase the robustness of BUILDPURECLUSTERS, the ROBUSTBUILDPURECLUSTERS algorithm, as given in Table 8. It starts with a first step called FINDINITIALSELECTION (Table 9). The goal of FINDINITIALSELECTION is to find a pure model using only DISJOINTGROUP instead of CS1, CS2 or CS3. This pure model is then used as an starting point for learning a more complete model in the remaining stages of ROBUSTBUILDPURECLUSTERS.

In FINDINITIALSELECTION, if a pair $\{X,Y\}$ cannot be separated into different clusters, but also does not participate in any successful application of DISJOINTGROUP, then this pair will be connected by a GRAY or YELLOW edge: this indicates that these two nodes cannot be in a pure submodel with three indicators per latent. Otherwise, these nodes are "compatible", meaning that they *might* be in such a pure model. This is indicated by a BLUE edge.

In FINDINITIALSELECTION we then find cliques of compatible nodes (Step 8).[17] Each clique is a candidate for a one-factor model (a latent model with one latent only). We purify every clique found to create pure one-factor models (Step 9). This avoids using clusters that are large not because they are all unique children of the same latent, but because there was no way of separating its elements. This adds considerably more computational cost to the whole procedure.

After we find pure one-factor models M_i, we search for a combination of compatible groups. Step 10 first indicates which pairs of one-factor models cannot be part of a pure model with three indicators each: if M_i and M_j are not pairwise a two-factor model with three pure indicators (as tested by DISJOINTGROUP), they cannot be both part of a valid solution.

CHOOSECLUSTERINGCLIQUE is a heuristic designed to find a large set of one-factor models (nodes of H) that can be grouped into a pure model with three indicators per latent (we need a

17. Any algorithm can be used to find maximal cliques. Notice that, by the anytime properties of our approach, one does not need to find all maximal cliques.

Algorithm ROBUSTBUILDPURECLUSTERS
Input: Σ, a sample covariance matrix of a set of variables **O**

1. $(Selection, C, C_0) \leftarrow$FINDINITIALSELECTION$(\Sigma)$.
2. For every pair of nonadjacent nodes $\{N_1, N_2\}$ in C where at least one of them is not in *Selection* and an edge $N_1 - N_2$ exists in C_0, add a RED edge $N_1 - N_2$ to C.
3. For every pair of nodes linked by a RED edge in C, apply successively rules CS1, CS2 and CS3. Remove an edge between every pair corresponding to a rule that applies.
4. Let H be a complete graph where each node corresponds to a maximal clique in C.
5. *FinalClustering* $\leftarrow$ CHOOSECLUSTERINGCLIQUE(H).
6. Return ROBUSTPURIFY$(FinalClustering, C, \Sigma)$.

Table 8: A modified BUILDPURECLUSTERS algorithm.

heuristic since finding a maximum clique in H is NP-hard). First, we define the *size* of a clustering $H_{candidate}$ (a set of nodes from H) as the number of variables that remain according to the following elimination criteria: 1. eliminate all variables that appear in more than one one-factor model inside $H_{candidate}$; 2. for each pair of variables $\{X_1, X_2\}$ such that X_1 and X_2 belong to different one-factor models in $H_{candidate}$, if there is an edge $X_1 - X_2$ in C, then we remove one element $\{X_1, X_2\}$ from $H_{candidate}$ (i.e., guarantee that no pair of variables from different clusters which were not shown to have any common latent parent will exist in $H_{candidate}$). We eliminate the one that belongs to the largest cluster, unless the smallest cluster has less than three elements to avoid extra fragmentation; 3. eliminate clusters that have less than three variables.

The heuristic motivation is that we expected that a model with a large size will have a large number of variables after purification. Our suggested heuristic to be implemented as CHOOSECLUSTERINGCLIQUE is trying to find a good model using a very simple hill-climbing algorithm that starts from an arbitrary node in H and add new clusters to the current candidate according to the one that will increase its size mostly while still forming a maximal clique in H. We stop when we cannot increase the size of the candidate. This is calculated using each node in H as a starting point, and the largest candidate is returned by CHOOSECLUSTERINGCLIQUE.

A.3 Clustering Refinement

The next steps in ROBUSTBUILDPURECLUSTERS are basically the FINDPATTERN algorithm of Table 1 with a final purification. The main difference is that we do not check anymore if pairs of nodes in the initial clustering given by *Selection* should be separated. The intuition explaining the usefulness of this implementation is as follows: if there is a group of latents forming a pure subgraph of the true graph with a large number of pure indicators for each latent, then the initial step should identify such group. The consecutive steps will refine this solution without the risk of splitting the large clusters of variables, which are exactly the ones most likely to produce false positive decisions. ROBUSTBUILDPURECLUSTERS has the power of identifying the latents with large sets of pure indicators and refining this solution with more flexible rules, covering also cases where DISJOINTGROUP fails.

Algorithm FINDINITIALSELECTION
Input: Σ, a sample covariance matrix of a set of variables $\mathbf{O}$

1. Start with a complete graph C over $\mathbf{O}$.
2. Remove edges of pairs that are marginally uncorrelated or uncorrelated conditioned on a third variable.
3. $C_0 \leftarrow C$.
4. Color every edge of C as BLUE.
5. For all edges $N_1 - N_2$ in C, if there is no other pair $\{N_3, N_4\}$ such that all three tetrads constraints hold in the covariance matrix of $\{N_1, N_2, N_3, N_4\}$, change the color of the edge $N_1 - N_2$ to GRAY.
6. For all pairs of variables $\{N_1, N_2\}$ linked by a BLUE edge in C

   If there exists a pair $\{N_3, N_4\}$ that forms a BLUE clique with N_1 in C, and a pair $\{N_5, N_6\}$ that forms a BLUE clique with N_2 in C, all six nodes form a clique in C_0 and DISJOINTGROUP$(N_1, N_3, N_4, N_2, N_5, N_6; \Sigma) = true$, then remove all edges linking elements in $\{N_1, N_3, N_4\}$ to $\{N_2, N_5, N_6\}$.

   Otherwise, if there is no node N_3 that forms a BLUE clique with $\{N_1, N_2\}$ in C, and no BLUE clique in $\{N_4, N_5, N_6\}$ such that all six nodes form a clique in C_0 and DISJOINTGROUP$(N_1, N_2, N_3, N_4, N_5, N_6; \Sigma) = true$, then change the color of the edge $N_1 - N_2$ to YELLOW.
7. Remove all GRAY and YELLOW edges from C.
8. $List_C \leftarrow$FINDMAXIMALCLIQUES(C).
9. Let H be a graph where each node corresponds to an element of $List_C$ and with no edges. Let M_i denote both a node in H and the respective set of nodes in $List_C$. Let $M_i \leftarrow$ ROBUSTPURIFY(M_i, C, Σ);
10. Add an edge $M_1 - M_2$ to H only if there exists $\{N_1, N_2, N_3\} \subseteq M_1$ and $\{N_4, N_5, N_6\} \subseteq M_2$ such that DISJOINTGROUP$(N_1, N_2, N_3, N_4, N_5, N_6; \Sigma) = true$.
11. $H_{choice} \leftarrow$CHOOSECLUSTERINGCLIQUE(H).
12. Let $H_{clusters}$ be the corresponding set of clusters, i.e., the set of sets of observed variables, where each set in $H_{clusters}$ correspond to some M_i in H_{choice}.
13. $Selection \leftarrow$ROBUSTPURIFY$(H_{clusters}, C, \Sigma)$.
14. Return $(Selection, C, C_0)$.

Table 9: Selects an initial pure model.

Notice that the order by which tests are applied might influence the outcome of the algorithms, since if we remove an edge $X - Y$ in C at some point, then we are excluding the possibility of using some tests where X and Y are required. Imposing such restriction reduces the overall computational cost and statistical mistakes. To minimize the ordering effect, an option is to run the algorithm multiple times and select the output with the highest number of nodes.

Appendix B. Proofs

Before we present the proofs of our results, we need a few more definitions:

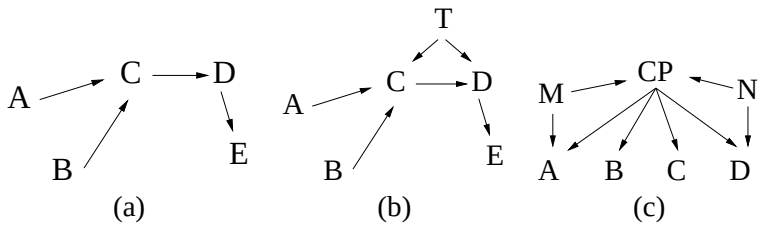


Figure 22: In (a), C is a choke point for sets $\{A,B\} \times \{D,E\}$, since it lies on all treks connecting nodes in $\{A,B\}$ to nodes in $\{D,E\}$ and lies also on the $\{D,E\}$ side of all of such treks. For instance, C is on the $\{D,E\}$ side of $A \rightarrow C \rightarrow D$, where A is the source of such a trek. Notice also that this choke point d-separates nodes in $\{A,B\}$ from nodes in $\{D,E\}$. Analogously, D is also a choke point for $\{A,B\} \times \{D,E\}$ (there is nothing on the definition of a choke point $\mathbf{I} \times \mathbf{J}$ that forbids it of belonging $\mathbf{I} \cup \mathbf{J}$). In Figure (b), C is a choke point for sets $\{A,B\} \times \{D,E\}$ that does not d-separate such elements. In Figure (c), CP is a node that lies on all treks connecting $\{A,C\}$ and $\{B,D\}$ but it is not a choke point, since it does not lie on the $\{A,C\}$ side of trek $A \leftarrow M \rightarrow CP \rightarrow B$ and neither lies on the $\{B,D\}$ side of $D \leftarrow N \rightarrow CP \rightarrow A$. The same node, however, is a $\{A,D\} \times \{B,C\}$ choke point.

- a *path* in a graph G is a sequence of nodes $\{X_1,\ldots,X_n\}$ such that X_i and X_{i+1} are adjacent in G, $1 \leq i < n$. Paths are assumed to be *simple* by definition, i.e., no node appears more than once. Notice there is an unique set of edges associated with each given path. A path is *into* X_1 (or X_n) if the arrow of the edge $\{X_1,X_2\}$ is into X_1 ($\{X_{n-1},X_n\}$ into X_n);

- a *collider* on a path $\{X_1,\ldots,X_n\}$ is a node X_i, $1 < i < n$, such that X_{i-1} and X_{i+1} are parents of X_i;

- a *trek* is a path that does not contain any collider;

- the *source* of a trek is the unique node in a trek to which no arrows are directed;

- the *I side* of a trek between nodes I and J with source X is the subpath directed from X to I. It is possible that $X = I$;

- a *choke point CP* between two sets of nodes $\mathbf{I}$ and $\mathbf{J}$ is a node that lies on every trek between any element of $\mathbf{I}$ and any element of $\mathbf{J}$ such that CP is either (i) on the $\mathbf{I}$ side of every such trek [18] or (ii) on the $\mathbf{J}$ side or every such trek.

With the exception of choke points, all other concepts are well known in the literature of graphical models (Spirtes et al., 2000; Pearl, 1988, 2000). What is interesting in a choke point is that, by definition, such a node is in all treks linking elements in two sets of nodes. Being in all treks connecting a node X_i and a node X_j is a necessary condition for a node to d-separate X_i and X_j, although this is not a sufficient condition.

18. That is, for every $\{I,J\} \in \mathbf{I} \times \mathbf{J}$, CP is on the I side of every trek $T = \{I,\ldots,X,\ldots,J\}$, X being the source of T.

Consider Figure 22, which illustrates several different choke points. In some cases, the choke point will d-separate a few nodes. The relevant fact is that even when the choke point is a latent variable, this has an implication on the observed marginal distribution, as stated by the *Tetrad Representation Theorem*:

Theorem 21 (The Tetrad Representation Theorem) *Let G be a linear latent variable model, and let I_1, I_2, J_1, J_2 be four variables in G. Then $\sigma_{I_1J_1}\sigma_{I_2J_2} = \sigma_{I_1J_2}\sigma_{I_2J_1}$ if and only if there is a choke point between $\{I_1, I_2\}$ and $\{J_1, J_2\}$.*

Proof: The original proof was given by Spirtes et al. (2000). Shafer et al. (1993) provide an alternative and simplied proof. □

Shafer et al. (1993) also provide more details on the definitions and several examples.

Therefore, unlike a partial correlation constraint obtained by conditioning on a given set of variables, where such a set should be observable, *some d-separations due to latent variables can be inferred using tetrad constraints*. We will use the Tetrad Representation Theorem to prove most of our results. The challenge lies on choosing the right combination of tetrad constraints that allows us to identify latents and d-separations due to latents, since the Tetrad Representation Theorem is far from providing such results directly.

In the following proofs, we will frequently use the symbol $G(\mathbf{O})$ to represent a linear latent variable model with a set of observed nodes $\mathbf{O}$. A choke point between sets $\mathbf{I}$ and $\mathbf{J}$ will be denoted as $\mathbf{I} \times \mathbf{J}$. We will first introduce a lemma that is going to be useful to prove several other results.

Lemma 9 *Let $G(\mathbf{O})$ be a linear latent variable model, and let $\{X_1, X_2, X_3, X_4\} \subset \mathbf{O}$ be such that $\sigma_{X_1X_2}\sigma_{X_3X_4} = \sigma_{X_1X_3}\sigma_{X_2X_4} = \sigma_{X_1X_4}\sigma_{X_2X_3}$. If $\rho_{AB} \neq 0$ for all $\{A, B\} \subset \{X_1, X_2, X_3, X_4\}$, then an unique node P entails all the given tetrad constraints, and P d-separates all elements in $\{X_1, X_2, X_3, X_4\}$.*

Proof: Let P be a choke point for pairs $\{X_1, X_2\} \times \{X_3, X_4\}$. Let Q be a choke point for pairs $\{X_1, X_3\} \times \{X_2, X_4\}$. We will show that $P = Q$ by contradiction.

Assume $P \neq Q$. Because there is a trek that links X_1 and X_4 through P (since $\rho_{X_1X_4} \neq 0$), we have that Q should also be on that trek. Suppose T is a trek connecting X_1 to X_4 through P and Q, and without loss of generality assume this trek follows an order that defines three subtreks: T_0, from X_1 to P; T_1, from P to Q; and T_2, from Q to X_4, as illustrated by Figure 23(a). In principle, T_0 and T_2 might be empty, i.e., we are not excluding the possibility that $X_1 = P$ or $X_4 = Q$.

There must be at least one trek T_{Q2} connecting X_2 and Q, since Q is on every trek between X_1 and X_2 and there is at least one such trek (since $\rho_{X_1X_2} \neq 0$). We have the following cases:

Case 1: T_{Q2} includes P. T_{Q2} has to be into P, and $P \neq X_1$, or otherwise there will be a trek connecting X_2 to X_1 through a (possibly empty) trek T_0 that does not include Q, contrary to our hypothesis. For the same reason, T_0 has to be into P. This will imply that T_1 is a directed path from P to Q, and T_2 is a directed path from Q to X_4 (Figure 23(b)).

Because there is at least one trek connecting X_1 and X_2 (since $\rho_{X_1X_2} \neq 0$), and because Q is on every such trek, Q has to be an ancestor of at least one member of $\{X_1, X_2\}$. Without loss of generality, assume Q is an ancestor of X_1. No directed path from Q to X_1 can include P, since P is an ancestor of Q and the graph is acyclic. Therefore, there is a trek connecting X_1 and X_4 with Q as the

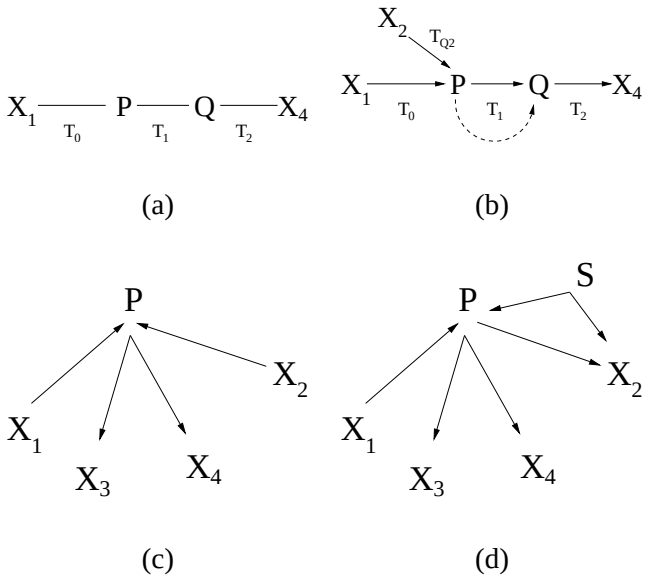


Figure 23: In (a), a depiction of a trek T linking X_1 and X_4 through P and Q, creating three subtreks labeled as T_0, T_1 and T_2. Directions in such treks are left unspecified. In (b), the existence of a trek T_{Q2} linking X_2 and Q through P will compel the directions depicted as a consequence of the given tetrad and correlation constraints (the dotted path represents any possible continuation of T_{Q2} that does not coincide with T). The configuration in (c) cannot happen if P is a choke point entailing all three tetrads among marginally dependent nodes $\{X_1, X_2, X_3, X_4\}$. The configuration in (d) cannot happen if P is a choke point for $\{X_1, X_3\} \times \{X_2, X_4\}$, since there is a trek $X_1 - P - X_2$ such that P is not on the $\{X_1, X_3\}$ side of it, and another trek $X_2 - S - P - X_3$ such that P is not on the $\{X_2, X_4\}$ side of it.

source that does not include P, contrary to our hypothesis.

Case 2: T_{Q2} does not include P. This case is similar to Case 1. T_{Q2} has to be into Q, and $Q \neq X_4$, or otherwise there will be a trek connecting X_2 to X_4 through a (possible empty) trek T_2 that does not include P, contrary to our hypothesis. For the same reason, T_2 has to be into Q. This will imply that T_1 is a directed path from Q to P, and T_0 is a directed path from P to X_1. An argument analogous to Case 1 will follow.

We will now show by that P d-separates all nodes in $\{X_1, X_2, X_3, X_4\}$. From the $P = Q$ result, we know that P lies on every trek between any pair of elements in $\{X_1, X_2, X_3, X_4\}$. First consider the case where at most one element of $\{X_1, X_2, X_3, X_4\}$ is linked to P through a trek that is into P. By the Tetrad Representation Theorem, any trek connecting two elements of $\{X_1, X_2, X_3, X_4\}$ goes through P. Since P cannot be a collider on any trek, then P d-separates these two elements.

To finish the proof, we only have to show that P cannot be a collider in a path connecting any two elements of $\{X_1, X_2, X_3, X_4\}$. We will prove that by contradiction. That is, assume without loss of generality that there is a trek connecting X_1 and P that is into P, and a trek connecting X_2 and P that is into P. We will show this either entails that $\rho_{X_1X_2} = 0$ or that P is not a choke point for $\{X_1, X_3\} \times \{X_2, X_4\}$.

Case 3: *there is no trek connecting X_1 and P that is out of P neither any trek connecting X_2 and P that is out of P.* This implies there is no trek connecting X_1 and X_2, since P is on every trek connecting these two elements according to the Tetrad Representation Theorem. But this implies $\rho_{X_1X_2} = 0$, a contradiction, as illustrated by Figure 23(c).

Case 4 (this case will be similar to the example given in Figure 22(c)): *assume without loss of generality that there is also a trek out of P and into X_2.* Then there is a trek connecting X_1 to X_2 through P that is not on the $\{X_1, X_3\}$ side of pair $\{X_1, X_3\} \times \{X_2, X_4\}$ to which P is a choke point. Therefore, P should be on the $\{X_2, X_4\}$ of every trek connecting elements pairs in $\{X_1, X_3\} \times \{X_2, X_4\}$. Without loss of generality, assume there is a trek out of P and into X_3 (because if there is no such trek for either X_3 and X_4, we fall in the previous case by symmetry). Let S be the source of a trek into P and X_2, which should exist since X_2 is not an ancestor of P. Then there is a trek of source S connecting X_3 and X_2 such that P is not on the $\{X_2, X_4\}$ side of it as shown in Figure 23(d). Therefore P cannot be a choke point for $\{X_1, X_3\} \times \{X_2, X_4\}$. Contradiction. □

Lemma 13 *Let $G(\mathbf{O})$ be a linear latent variable model. If for some set $\mathbf{O}' = \{X_1, X_2, X_3, X_4\} \subseteq \mathbf{O}$, $\sigma_{X_1X_2}\sigma_{X_3X_4} = \sigma_{X_1X_3}\sigma_{X_2X_4} = \sigma_{X_1X_4}\sigma_{X_2X_3}$ and for all triplets $\{A, B, C\}$, $\{A, B\} \subset \mathbf{O}', C \in \mathbf{O}$, we have $\rho_{AB.C} \neq 0$ and $\rho_{AB} \neq 0$, then no element $A \in \mathbf{O}'$ is a descendant of an element of $\mathbf{O}' \setminus \{A\}$ in G.*

Proof: Without loss of generality, assume for the sake of contradiction that X_1 is an ancestor of X_2. From the given tetrad and correlation constraints and Lemma 9, there is a node P that lies on every trek between X_1 and X_2 and d-separates these two nodes. Since P lies on the directed path from X_1 to X_2, P is a descendant of X_1, and therefore an observed node. However, this implies $\rho_{X_1X_2.P} = 0$, contrary to our hypothesis. □

Lemma 10 *Let $G(\mathbf{O})$ be a linear latent variable model. Assume $\mathbf{O}' = \{X_1, X_2, X_3, Y_1, Y_2, Y_3\} \subseteq \mathbf{O}$. If constraints $\{\tau_{X_1Y_1X_2X_3}, \tau_{X_1Y_1X_3X_2}, \tau_{Y_1X_1Y_2Y_3}, \tau_{Y_1X_1Y_3Y_2}, \neg\tau_{X_1X_2Y_2Y_1}\}$ all hold, and that for all triplets $\{A, B, C\}, \{A, B\} \subset \mathbf{O}'$, $C \in \mathbf{O}$, we have $\rho_{AB} \neq 0, \rho_{AB.C} \neq 0$, then X_1 and Y_1 do not have a common parent in G.*

Proof: We will prove this result by contradiction, by assuming that X_1 and Y_1 have a common parent L in G and showing this entails $\tau_{X_1X_2Y_2Y_1}$, contrary to the hypothesis.

Initially, we will show by contradiction that L is a choke point for $\{X_1, Y_1\} \times \{X_2, X_3\}$. Suppose L is not a choke point for $\{X_1, X_2\} \times \{Y_1, X_3\}$ corresponding to one of the tetrad constraints given by hypothesis. Because of the trek $X_1 \leftarrow L \rightarrow Y_1$, then either X_1 or Y_1 is a choke point. Without loss of generality, assume X_1 is a choke point in this case. By Lemma 9 and the given constraints, X_1 d-separates any two elements in $\{X_2, X_3, Y_1\}$ contrary to the hypothesis that $\rho_{X_2X_3.X_1} \neq 0$. By

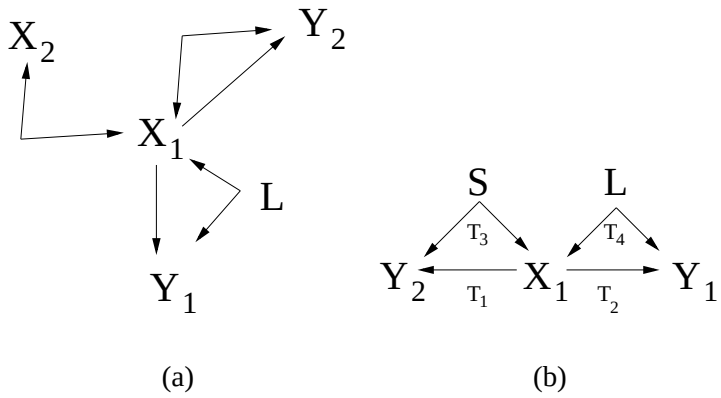


Figure 24: Figure (a) illustrates necessary treks among elements of $\{X_1, X_2, Y_1, Y_2, L\}$ according to the assumptions of Lemma 11 if we further assume that X_1 is a choke point for pairs $\{X_1, X_2\} \times \{Y_1, Y_2\}$ (other treks might exist). Figure (b) rearranges (a) by emphasizing that Y_1 and Y_2 cannot be d-separated by a single node.

symmetry, Y_1 cannot be a choke point. Therefore, L is a choke point for $\{X_1, Y_1\} \times \{X_2, X_3\}$ and by Lemma 9, it also lies on every trek for any pair in $\mathbf{S_1} = \{X_1, X_2, X_3, Y_1\}$.

Analogously, L is on every trek connecting any pair from the set $\mathbf{S_2} = \{X_1, Y_1, Y_2, Y_3\}$. It follows that L is on every trek connecting any pairs in the product $\{X_1, Y_1\} \times \{X_2, Y_2\}$, and it is on the $\{X_1, Y_1\}$ side of $\{X_1, Y_1\} \times \{X_2, Y_2\}$, i.e., L is a choke point that implies $\tau_{X_1 X_2 Y_2 Y_1}$. Contradiction. □

Remember that predicate $Factor(X, Y, G)$ is true if and only if there exist two nodes W and Z in G such that τ_{WXYZ} and τ_{WXZY} are both entailed, all nodes in $\{W, X, Y, Z\}$ are correlated, and there is no observed C in G such that $\rho_{AB.C} = 0$ for $\{A, B\} \subset \{W, X, Y, Z\}$.

Lemma 11 *Let $G(\mathbf{O})$ be a linear latent variable model. Assume $\mathbf{O}' = \{X_1, X_2, X_3, Y_1, Y_2, Y_3\} \subseteq \mathbf{O}$, such that $Factor(X_1, X_2, G)$ and $Factor(Y_1, Y_2, G)$ hold, Y_1 is not an ancestor of Y_3 and X_1 is not an ancestor of X_3. If constraints $\{\tau_{X_1 Y_1 Y_2 X_2}, \tau_{X_2 Y_1 Y_3 Y_2}, \tau_{X_1 X_2 Y_2 X_3}, \neg\tau_{X_1 X_2 Y_2 Y_1}\}$ all hold, and that for all triplets $\{A, B, C\}, \{A, B\} \subset \mathbf{O}', C \in \mathbf{O}$, we have $\rho_{AB} \neq 0, \rho_{AB.C} \neq 0$, then X_1 and Y_1 do not have a common parent in G.*

Proof: We will prove this result by contradiction. Assume X_1 and Y_1 have a common parent L. Because of the tetrad constraints given by hypothesis and the existence of the trek $X_1 \leftarrow L \rightarrow Y_1$, one node in $\{X_1, L, Y_1\}$ should be a choke point for the pair $\{X_1, X_2\} \times \{Y_1, Y_2\}$. We will first show that L has to be such a choke point, and therefore lies on every trek connecting X_1 and Y_2, as well as X_2 and Y_1. We then show that L lies on every trek connecting Y_1 and Y_2, as well as X_1 and X_2. Finally, we show that L is a choke point for $\{X_1, Y_1\} \times \{X_2, Y_2\}$, contrary to our hypothesis.

Step 1: If there is a common parent L to X_1 and Y_1, then L is a $\{X_1, X_2\} \times \{Y_1, Y_2\}$ choke point. For the sake of contradiction, assume X_1 is a choke point in this case. By Lemma 13 and assumption

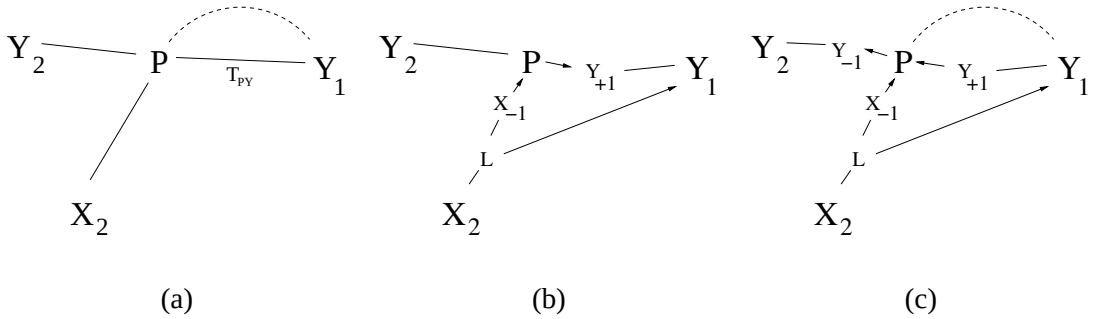


Figure 25: In (a), a depiction of T_Y and T_X, where edges represent treks (T_X can be seen more generally as the combination of the solid edge between X_2 and P concatenated with a dashed edge between P and Y_1 representing the possibility that T_Y and T_X might intersect multiple times in T_{PY}, but in principle do not need to coincide in T_{PY} if P is not a choke point.) In (b), a possible configurations of edges $< X_{-1}, P >$ and $< P, Y_{+1} >$ that do not collide in P, and P is a choke point (and $Y_{+1} \neq Y$). In (c), the edge $< Y_{-1}, P >$ is compelled to be directed away from P because of the collider with the other two neighbors of P.

$Factor(X_1, X_2, G)$, we have that X_1 is not an ancestor of X_2, and therefore all treks connecting X_1 and X_2 should be into X_1. Since $\rho_{X_2 Y_2} \neq 0$ by assumption and X_1 is on all treks connecting X_2 and Y_2, there must be a directed path out of X_1 and into Y_2. Since $\rho_{X_2 Y_2 . X_1} \neq 0$ by assumption and X_1 is on all treks connecting X_2 and Y_2, there must be a trek into X_1 and Y_2. Because $\rho_{X_2 Y_1} \neq 0$, there must be a trek out of X_1 and into Y_1. Figure 24(a) illustrates the configuration.

Since $Factor(Y_1, Y_2, G)$ is true, by Lemma 9 there must be a node d-separating Y_1 and Y_2 (neither Y_1 nor Y_2 can be the choke point in $Factor(Y_1, Y_2, G)$ because this choke point has to be latent, according to the partial correlation conditions of *Factor*). However, by Figure 24(b), treks $T_2 - T_3$ and $T_1 - T_4$ cannot both be blocked by a single node. Contradiction. Therefore X_1 cannot be a choke point for $\{X_1, X_2\} \times \{Y_1, Y_2\}$ and, by symmetry, neither can Y_1.

Step 2: L is on every trek connecting Y_1 and Y_2 and on every trek connecting X_1 and X_2. Let L be the choke point for pairs $\{X_1, X_2\} \times \{Y_1, Y_2\}$. As a consequence, all treks between Y_2 and X_1 go through L. All treks between X_2 and Y_1 go through L. All treks between X_2 and Y_2 go through L. Such treks exist, since no respective correlation vanishes.

Consider the given hypothesis $\sigma_{X_2 Y_1} \sigma_{Y_2 Y_3} = \sigma_{X_2 Y_3} \sigma_{Y_2 Y_1}$, corresponding to a choke point $\{X_2, Y_2\} \times \{Y_1, Y_3\}$. From the previous paragraph, we know there is a trek linking Y_2 and L. L is a parent of Y_1 by construction. That means Y_2 and Y_1 are connected by a trek through L.

We will show by contradiction that L is on every trek connecting Y_1 and Y_2. Assume there is a trek T_Y connecting Y_2 and Y_1 that does not contain L. Let P be the first point of intersection of T_Y and a trek T_X connecting X_2 to Y_1, starting from X_2. If T_Y exists, such point should exist, since T_Y should contain a choke point $\{X_2, Y_2\} \times \{Y_1, Y_3\}$, and all treks connecting X_2 and Y_1 (including T_X) contain the same choke point.

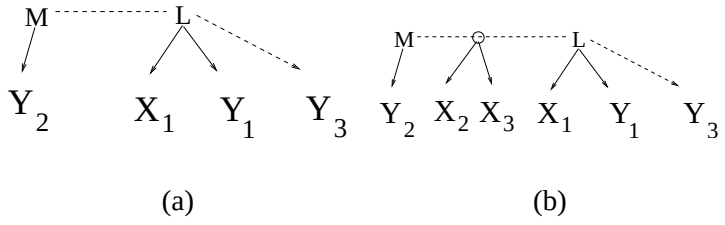


Figure 26: In (a), Y_2 and X_1 cannot share a parent, and because of the given tetrad constraints, L should d-separate M and Y_3. Y_3 is not a child of L either, but there will be a trek linking L and Y_3. In (b), an (invalid) configuration for X_2 and X_3, where they share an ancestor between M and L.

Let T_{PY} be the subtrek of T_Y starting on P and ending one node before Y_1. Any choke point $\{X_2,Y_2\}\times\{Y_1,Y_3\}$ should lie on T_{PY} (Figure 25(a)). (Y_1 cannot be such a choke point, since all treks connecting Y_1 and Y_2 are into Y_1, and by hypothesis all treks connecting Y_1 and Y_3 are into Y_1. Since all treks connecting Y_2 and Y_3 would need to go through Y_1 by definition, then there would be no such trek, implying $\rho_{Y_2Y_3}=0$, contrary to our hypothesis.)

Assume first that $X_2\neq P$ and $Y_2\neq P$. Let X_{-1} be the node before P in T_X starting from X_2. Let Y_{-1} be the node before P in T_Y starting from Y_2. Let Y_{+1} be the node after P in T_Y starting from Y_2 (notice that it is possible that $Y_{+1}=Y_1$). If X_{-1} and Y_{+1} do not collide on P (i.e., there is no structure $X_{-1}\rightarrow P\leftarrow Y_{+1}$), then there will be a trek connecting X_2 to Y_1 through T_{PY} after P. Since L is not in T_{PY}, L should be before P in T_X. But then there will be a trek connecting X_2 and Y_1 that does not intersect T_{PY}, which is a contradiction (Figure 25(b)). If the collider does exist, we have the edge $P\leftarrow Y_{+1}$. Since no collider $Y_{-1}\rightarrow P\leftarrow Y_{+1}$ can exist because T_Y is a trek, the edge between Y_{-1} and P is out of P. But that forms a trek connecting X_2 and Y_2 (Figure 25(c)), and since L is in every trek between X_2 and Y_2 and T_Y does not contain L, then T_X should contain L before P, which again creates a trek between X_2 and Y_1 that does not intersect T_{PY}.

If $X_2=P$, then T_{PY} has to contain L, because every trek between X_2 and Y_1 contains L. Therefore, $X_2\neq P$. If $Y_2=P$, then because every trek between X_2 and Y_2 should contain L, we again have that L lies in T_X before P, which creates a trek between X_2 and Y_1 that does not intersect T_{PY}. Therefore, we showed by contradiction that L lies on every trek between Y_2 and Y_1.

Consider now the given hypothesis $\sigma_{X_1X_2}\sigma_{X_3Y_2}=\sigma_{X_1Y_2}\sigma_{X_3X_2}$, corresponding to a choke point $\{X_2,Y_2\}\times\{X_1,X_3\}$. By symmetry with the previous case, all treks between X_1 and X_2 go through L.

Step 3: If L exists, so does a choke point $\{X_1,Y_1\}\times\{X_2,Y_2\}$. By the previous steps, L intermediates all treks between elements of the pair $\{X_1,Y_1\}\times\{X_2,Y_2\}$. Because L is a common parent of $\{X_1,Y_1\}$, it lies on the $\{X_1,Y_1\}$ side of every trek connecting pairs of elements in $\{X_1,Y_1\}\times\{X_2,Y_2\}$. L is a choke point for this pair. This implies $\tau_{X_1X_2Y_2Y_1}$. Contradiction. □

Lemma 12 *Let* $G(\mathbf{O})$ *be a linear latent variable model. Let* $\mathbf{O}'=\{X_1,X_2,X_3,Y_1,Y_2,Y_3\}\subseteq\mathbf{O}$. *If constraints* $\{\tau_{X_1Y_1Y_2Y_3},\tau_{X_1Y_1Y_3Y_2},\tau_{X_1Y_2X_2X_3},\tau_{X_1Y_2X_3X_2},\tau_{X_1Y_3X_2X_3},\tau_{X_1Y_3X_3X_2},\neg\tau_{X_1X_2Y_2Y_3}\}$ *all hold, and that for all triplets* $\{A,B,C\},\{A,B\}\subset\mathbf{O}',C\in\mathbf{O}$, *we have* $\rho_{AB}\neq 0,\rho_{AB.C}\neq$

0, *then* X_1 *and* Y_1 *do not have a common parent in* G.

Proof: We will prove this result by contradiction. Suppose X_1 and Y_1 have a common parent L in G. Since all three tetrads hold in the covariance matrix of $\{X_1, Y_1, Y_2, Y_3\}$, by Lemma 9 the choke point that entails these constraints d-separates the elements of $\{X_1, Y_1, Y_2, Y_3\}$. The choke point should be in the trek $X_1 \leftarrow L \rightarrow Y_1$, and since it cannot be an observed node because by hypothesis no d-separation conditioned on a single node holds among elements of $\{X_1, Y_1, Y_2, Y_3\}$, L has to be a latent choke point for all pairs of pairs in $\{X_1, Y_1, Y_2, Y_3\}$.

It is also given that $\{\tau_{X_1Y_2X_2X_3}, \tau_{X_1Y_2X_3X_2}, \tau_{X_1Y_1Y_2Y_3}, \tau_{X_1Y_1Y_3Y_2}\}$ holds. Since it is the case that $\neg\tau_{X_1X_2Y_2Y_3}$, by Lemma 10 X_1 and Y_2 cannot share a parent. Let T_{ML} be a trek connecting some parent M of Y_2 and L. Such a trek exists because $\rho_{X_1Y_2} \neq 0$.

We will show by contradiction that there is no node in $T_{ML} \backslash L$ that is connected to Y_3 by a trek that does not go through L. Suppose there is such a node, and call it V. If the trek connecting V and Y_3 is into V, and since V is not a collider in T_{ML}, then V is either an ancestor of M or an ancestor of L. If V is an ancestor of M, then there will be a trek connecting Y_2 and Y_3 that is not through L, which is a contradiction. If V is an ancestor of L but not M, then both Y_2 and Y_3 are d-connected to a node V is a collider at the intersection of such d-connecting treks. However, V is an ancestor of L, which means L cannot d-separate Y_2 and Y_3, a contradiction. Finally, if the trek connecting V and Y_3 is out of V, then Y_2 and Y_3 will be connected by a trek that does not include L, which again is not allowed. We therefore showed there is no node with the properties of V. This configuration is illustrated by Figure 26(a).

Since all three tetrads hold among elements of $\{X_1, X_2, X_3, Y_2\}$, then by Lemma 9, there is a single choke point P that entails such tetrads and d-separates elements of this set. Since T_{ML} is a trek connecting Y_2 to X_1 through L, then there are three possible locations for P in G:

Case 1: $P = M$. We have all treks between X_3 and X_2 go through M but not through L, and some trek from X_1 to Y_3 goes through L but not through M. No choke point can exist for pairs $\{X_1, X_3\} \times \{X_2, Y_3\}$, which by the Tetrad Representation Theorem means that the tetrad $\sigma_{X_1Y_3}\sigma_{X_2X_3} = \sigma_{X_1X_2}\sigma_{Y_3X_3}$ cannot hold, contrary to our hypothesis.

Case 2: P lies between M and L in T_{ML}. This configuration is illustrated by Figure 26(b). As before, no choke point exists for pairs $\{X_1, X_3\} \times \{X_2, Y_3\}$, contrary to our hypothesis.

Case 3: $P = L$. Because all three tetrads hold in $\{X_1, X_2, X_3, Y_3\}$ and L d-separates all pairs in $\{X_1, X_2, X_3\}$, one can verify that L d-separates all pairs in $\{X_1, X_2, X_3, Y_3\}$. This will imply a $\{X_1, Y_3\} \times \{X_2, Y_2\}$ choke point, contrary to our hypothesis. □

Theorem 14 *The output of* FINDPATTERN *is a measurement pattern with respect to the tetrad and vanishing partial correlation constraints of* Σ

Proof: Two nodes will not share a common latent parent in a measurement pattern if and only if they are not linked by an edge in graph C constructed by algorithm FINDPATTERN and that happens if and only if some partial correlation vanishes or if any of rules CS1, CS2 or CS3 applies. But then by Lemmas 10, 11, 12 and the equivalence of vanishing partial correlations and conditional independence in linearly faithful distributions (Spirtes et al., 2000) the claim is proved. The claim

about undirected edges follows from Lemma 13. □

Theorem 15 *Given a covariance matrix* Σ *assumed to be generated from a linear latent variable model* $G(\mathbf{O})$ *with latent variables* $\mathbf{L}$*, let* G_{out} *be the output of* BUILDPURECLUSTERS(Σ) *with observed variables* $\mathbf{O_{out}} \subseteq \mathbf{O}$ *and latent variables* $\mathbf{L_{out}}$*. Then* G_{out} *is a measurement pattern, and there is an injective mapping* $M : \mathbf{L_{out}} \rightarrow \mathbf{L}$ *with the following properties:*

1. *Let* $L_{out} \in \mathbf{L_{out}}$*. Let* $\mathbf{X}$ *be the children of* L_{out} *in* G_{out}*. Then* $M(L_{out})$ *d-separates any element* $X \in \mathbf{X}$ *from* $\mathbf{O_{out}} \backslash X$ *in* G*;*

2. $M(L_{out})$ *d-separates* X *from every latent in* G *for which* $M^{-1}(.)$ *exists;*

3. *Let* $\mathbf{O'} \subseteq \mathbf{O_{out}}$ *be such that each pair in* $\mathbf{O'}$ *is correlated. At most one element in* $\mathbf{O'}$ *with latent parent* L_{out} *in* G_{out} *is not a descendant of* $M(L_{out})$ *in* G*, or has a hidden common cause with it;*

Proof: We will start by showing that for each cluster Cl_i in G_{out}, there exists an unique latent L_i in G that d-separates all elements of Cl_i. This shows the existance of an unique function from latents in G_{out} to latents in G. We then proceed to prove the three claims given in the theorem, and finish by proving that the given function is injective.

Let Cl_i be a cluster in a non-empty G_{out}. Cl_i has three elements X, Y and Z, and there is at least some W in G_{out} such that all three tetrad constraints hold in the covariance matrix of $\{W, X, Y, Z\}$, where no pair of elements in $\{X, Y, Z\}$ is marginally d-separated or d-separated by an observable variable. By Lemma 9, it follows that there is an unique latent L_i d-separating X, Y and Z. If Cl_i has more than three elements, it follows that since no node other than L_i can d-separate all three elements in $\{X, Y, Z\}$, and any choke point for $\{W', X, Y, Z\}$, $W' \in Cl_i$, will d-separate all elements in $\{W', X, Y, Z\}$, then there is an unique latent L_i d-separating all elements in Cl_i. An analogous argument concerns the d-separation of any element of Cl_i and observed nodes in other clusters.

Now we will show that each L_i d-separates each X in Cl_i from all other mapped latents. As a byproduct, we will also show the validity of the third claim of the theorem. Consider $\{Y, Z\}$, two other elements of Cl_i besides X, and $\{A, B, C\}$, three elements of Cl_j. Since L_i and L_j each d-separate all pairs in $\{X, Y\} \times \{A, B\}$, and no pair in $\{X, Y\} \times \{A, B\}$ has both of its elements connected to L_i (L_j) through a trek that is into L_i (L_j) (since L_i, or L_j, d-separates then), then both L_i and L_j are choke points for $\{X, Y\} \times \{A, B\}$. According to Lemma 2.5 given by Shafer et al. (1993), any trek connecting an element from $\{X, Y\}$ to an element in $\{A, B\}$ passes through both choke points in the same order. Without loss of generality, assume the order is first L_i, then L_j.

If there is no trek connecting X to L_i that is into L_i, then L_i d-separates X and L_j. The same holds for L_j and A with respect to L_i. If there is a trek T connecting X and L_i that is into L_i, and since all three tetrad constraints hold in the covariance matrix of $\{X, Y, Z, A\}$ by construction, then there is no trek connecting A and L_i that is into L_i (Lemma 9). Since there are treks connecting L_i and L_j, they should be all out of L_i and into L_j. This means that L_i d-separates X and L_j. But this also creates a trek connecting X and L_j that is into L_j. Since all three tetrad constraints hold in the covariance matrix of $\{X, A, B, C\}$ by construction, then there is no trek connecting A and L_j that is into L_j (by the d-separation implied by Lemma 9). This means that L_j d-separates A from L_i. This also means that the existance of such a trek T out of X and into L_i forbids the existance of any trek connecting a variable correlated to X that is into L_i (since all treks connecting L_i and some L_j are out of L_i), which proves the third claim of the theorem.

We will conclude by showing that given two clusters Cl_i and Cl_j with respective latents L_i and L_j, where each cluster is of size at least three, if they are not merged, then $L_i \neq L_j$. That is, the mapping from latents in G_{out} to latents in G, as defined at the beginning of the proof, is injective.

Assume $L_i = L_j$. We will show that these clusters will be merged by the algorithm, proving the counterpositive argument. Let X and Y be elements of Cl_i and W, Z elements of Cl_j. It immediately follows that L_i is a choke point for all pairs in $\{W,X,Y,Z\}$, since L_i d-separates any pair of elements of $\{W,X,Y,Z\}$, which means all three tetrads will hold in the covariance matrix of any subset of size four from $Cl_i \cup Cl_j$. These two clusters will then be merged by BUILDPURECLUSTERS. □

Theorem 16 *Given a covariance matrix Σ assumed to be generated from a linear latent variable model $G(\mathbf{O})$ with latent variables $\mathbf{L}$, let G_{out} be the output of* BUILDPURECLUSTERS(Σ) *with observed variables $\mathbf{O_{out}} \subseteq \mathbf{O}$ and latent variables $\mathbf{L_{out}}$. Let $M(\mathbf{L_{out}}) \subseteq \mathbf{L}$ be the set of latents in G obtained by the mapping function $M()$. Let $\Sigma_{\mathbf{O_{out}}}$ be the population covariance matrix of $\mathbf{O_{out}}$, i.e., the corresponding marginal of Σ. Let the DAG G_{out}^{aug} be G_{out} augmented by connecting the elements of $\mathbf{L_{out}}$ such that the structural model of G_{out}^{aug} is an I-map of the distribution of $M(\mathbf{L_{out}})$. Then there exists a linear latent variable model using G_{out}^{aug} as the graphical structure such that the implied covariance matrix of $\mathbf{O_{out}}$ equals $\Sigma_{\mathbf{O_{out}}}$.*

Proof: If a linear model is an I-map DAG of the true distribution of its variables, then there is a well-known natural instantiation of the parameters of this model that will represent the true covariance matrix (Spirtes et al., 2000). We will assume such parametrization for the structural model, and denote as $\Sigma_L(\Theta)$ the parameterized latent covariance matrix. Instead of showing that G_{out}^{aug} is an I-map of the respective set of latents and observed variables and using the same argument, we will show a valid instantion of its parameters directly.

Assume without loss of generality that all variables have zero mean. To each observed node X with latent ancestor L_X in G such that $M^{-1}(L_X)$ is a parent of X in G_{out}, the linear model representation is:

$$X = \lambda_X L_X + \varepsilon_X$$

For this equation, we have two associated parameters, λ_X and $\sigma^2_{\varepsilon_X}$, where $\sigma^2_{\varepsilon_X}$ is the variance of ε_X. We instantiate them by the linear regression values, i.e., $\lambda_X = \sigma_{XL_X}/\sigma^2_{L_X}$, and $\sigma^2_{\varepsilon_X}$ is the respective residual variance. The set $\{\lambda_X\} \cup \{\sigma^2_{\varepsilon_X}\}$ of all λ_X and $\sigma^2_{\varepsilon_X}$, along with the parameters used in $\Sigma_L(\Theta)$, is our full set of parameters Θ.

Our definition of linear latent variable model requires $\sigma_{\varepsilon_X\varepsilon_Y} = 0$, $\sigma_{\varepsilon_X L_X} = 0$ and $\sigma_{\varepsilon_X L_Y} = 0$, for all $X \neq Y$. This corresponds to a covariance matrix $\Sigma(\Theta)$ of the observed variables with entries defined as:

$$E[X^2](\Theta) = \sigma^2_X(\Theta) = \lambda^2_X \sigma^2_{L_X} + \sigma^2_{\varepsilon_X}$$

$$E[XY](\Theta) = \sigma_{XY}(\Theta) = \lambda_X \lambda_T \sigma_{L_X L_Y}$$

To prove the theorem, we have to show that $\Sigma_{\mathbf{O_{out}}} = \Sigma(\Theta)$ by showing that correlations between different residuals, and residuals and latent variables, are actually zero.

The relation $\sigma_{\varepsilon_X L_X} = 0$ follows directly from the fact that λ_X is defined by the regression coefficient of X on L_X. Notice that if X and L_X do not have a common ancestor, λ_X is the direct effect

of L_X in X with respect to G_{out}. As we know, by Theorem 15, at most one variable in any set of correlated variables will not fulfill this condition.

We have to show also that $\sigma_{XY} = \sigma_{XY}(\Theta)$ for any pair X,Y in G_{out}. Residuals ε_X and ε_Y are uncorrelated due to the fact that X and Y are independent given their latent ancestors in G_{out}, and therefore $\sigma_{\varepsilon_X\varepsilon_Y} = 0$. To verify that $\sigma_{\varepsilon_X L_Y} = 0$ is less straightforward, but one can appeal to the graphical formulation of the problem. In a linear model, the residual ε_X is a function only of the variables that are not independent of X given L_X. None of this variables can be nodes in G_{out}, since L_X d-separates X from all such variables. Therefore, given L_X none of the variables that define ε_X can be dependent on L_Y, implying $\sigma_{\varepsilon_X L_Y} = 0$. □

Theorem 17 *Problem $\mathcal{MP}^3$ is NP-complete.*

Proof: Direct reduction from the 3-SAT problem: let S be a 3-CNF formula from which we want to decide if there is an assignment for its variables that makes the expression true. Define G as a latent variable graph with a latent node L_i for each clause C_i in M, with an arbitrary fully connected structural model. For each latent in G, add five pure children. Choose three arbitrary children of each latent L_i, naming them $\{C_i^1, C_i^2, C_i^3\}$. Add a bi-directed edge $C_i^p \leftrightarrow C_j^q$ for each pair $C_i^p, C_j^q, i \neq j$, if and only that they represent literals over the same variable but of opposite values. As in the maximum clique problem, one can verify that there is a pure submodel of G with at least three indicators per latent if and only if S is satisfiable. □

The next corollary suggests that even an invalid measurement pattern could be used in BUILDPURECLUSTERS instead of the output of FINDPATTERN. However, an arbitrary (invalid) measurement pattern is unlikely to be informative at all after being purified. In constrast, FINDPATTERN can be highly informative.

Corollary 18 *The output of* BUILDPURECLUSTERS *retains its guarantees even when rules CS1, CS2 and CS3 are applied an arbitrary number of times in* FINDPATTERN *for any arbitrary subset of nodes and an arbitrary number of maximal cliques is found.*

Proof: Independently of the choice made on Step 2 of BUILDPURECLUSTERS and which nodes are not separated into different cliques in FINDPATTERN, the exhaustive verification of tetrad constraints by BUILDPURECLUSTERS provides all the necessary conditions for the proof of Theorem 15. □

Corollary 20 *Given a covariance matrix Σ assumed to be generated from a linear latent variable model G, and G_{out} the output of* BUILDPURECLUSTERS *given Σ, the output of* PC-MIMBUILD *or* FCI-MIMBUILD *given (Σ, G_{out}) returns the correct Markov equivalence class of the latents in G corresponding to latents in G_{out} according to the mapping implicit in* BUILDPURECLUSTERS

Proof: By Theorem 15, each observed variable is d-separated from all other variables in G_{out} given its latent parent. By Theorem 16, one can parameterize G_{out} as a linear model such that the observed covariance matrix as a function of the parameterized G_{out} equals its corresponding marginal of Σ. By Theorem 19, the rank test using the measurement model of G_{out} is therefore a consistent independence test of latent variables. The rest follows immediately from the consistency property

of PC and FCI given a valid oracle for conditional independencies. □

References

H. Attias. Independent factor analysis. *Graphical Models: foundations of neural computation*, pages 207–257, 1999.

F. Bach and M. Jordan. Beyond independent components: trees and clusters. *Journal of Machine Learning Research*, 4:1205–1233, 2003.

D. Bartholomew and M. Knott. *Latent Variable Models and Factor Analysis*. Arnold Publishers, 1999.

D. Bartholomew, F. Steele, I. Moustaki, and J. Galbraith. *The Analysis and Interpretation of Multivariate Data for Social Scientists*. Arnold Publishers, 2002.

K. Bollen. *Structural Equation Models with Latent Variables*. John Wiley & Sons, 1989.

K. Bollen. Outlier screening and a distribution-free test for vanishing tetrads. *Sociological Methods and Research*, 19:80–92, 1990.

D. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

G. Elidan, N. Lotner, N. Friedman, and D. Koller. Discovering hidden variables: a structure-based approach. *Neural Information Processing Systems*, 13:479–485, 2000.

N. Friedman. The Bayesian structural EM algorithm. *Proceedings of 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

D. Geiger and C. Meek. Quantifier elimination for statistical problems. *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*, 1999.

C. Glymour. Social statistics and genuine inquiry: reflections on *the bell curve. Intelligence, Genes and Sucess: Scientists Respond to The Bell Curve*, 1997.

C. Glymour. *The Mind's Arrow: Bayes Nets and Graphical Causal Models in Psychology*. MIT Press, 2002.

C. Glymour, Richard Scheines, Peter Spirtes, and Kevin Kelly. *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling*. Academic Press, 1987.

Y. Kano and A. Harada. Stepwise variable selection in factor analysis. *Psychometrika*, 65:7–22, 2000.

J. Loehlin. *Latent Variable Models: An Introduction to Factor, Path and Structural Equation Analysis*. Lawrence Erlbaum, 2004.

C. Meek. *Graphical Models: Selecting Causal and Statistical Models*. PhD Thesis, Carnegie Mellon University, 1997.

J. Pearl. *Probabilistic Reasoning in Expert Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2000.

T. Richardson. A discovery algorithm for directed cyclic graphs. *Proceedings of 12th Conference on Uncertainty in Artificial Intelligence*, 1996.

G. Shafer, A. Kogan, and P.Spirtes. Generalization of the tetrad representation theorem. *DIMACS Technical Report*, 1993.

R. Silva. Automatic discovery of latent variable models. *PhD Thesis, Carnegie Mellon University, http://www.cs.cmu/edu/˜ rbas*, 2005.

R. Silva and R. Scheines. Generalized measurement models. *Technical Report CMU-CALD-04-101, Carnegie Mellon University*, 2004.

R. Silva and R. Scheines. New d-separation identification results for learning continuous latent variable models. *Proceedings of the 22nd Interational Conference in Machine Learning*, 2005.

R. Silva, R. Scheines, C. Glymour, and P. Spirtes. Learning measurement models for unobserved variables. *Proceedings of 19th Conference on Uncertainty in Artificial Intelligence*, pages 543–550, 2003.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Cambridge University Press, 2000.

J. Wishart. Sampling errors in the theory of two factors. *British Journal of Psychology*, 19:180–187, 1928.

N. Zhang. Hierarchical latent class models for cluster analysis. *Journal of Machine Learning Research*, 5:697–723, 2004.

Journal of Machine Learning Research 7 (2006) 247–282 Submitted 3/05; Revised 10/05; Published 2/06

In Search of Non-Gaussian Components of a High-Dimensional Distribution

Gilles Blanchard BLANCHAR@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany
and
CNRS, Université Paris-Sud
Orsay, France

Motoaki Kawanabe NABE@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Masashi Sugiyama SUGI@CS.TITECH.AC.JP
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany
and
Department of Computer Science
Tokyo Institute of Technology
2-12-1, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan

Vladimir Spokoiny SPOKOINY@WIAS-BERLIN.DE
Weierstrass Institute and Humboldt University
Mohrenstrasse 39
10117 Berlin, Germany

Klaus-Robert Müller KLAUS@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany
and
Department of Computer Science
University of Potsdam
August-Bebel-Strasse 89, Haus 4
14482 Potsdam, Germany

Editor: Sam Roweis

Abstract

Finding non-Gaussian components of high-dimensional data is an important preprocessing step for efficient information processing. This article proposes a new *linear* method to identify the "non-Gaussian subspace" within a very general semi-parametric framework. Our proposed method, called NGCA (non-Gaussian component analysis), is based on a linear operator which, to any arbitrary nonlinear (smooth) function, associates a vector belonging to the low dimensional non-Gaussian target subspace, up to an estimation error. By applying this operator to a family of different nonlinear functions, one obtains a family of different vectors lying in a vicinity of the target space. As a final step, the target space itself is estimated by applying PCA to this family of vectors. We show that this procedure is consistent in the sense that the estimaton error tends to zero at a parametric rate, uniformly over the family, Numerical examples demonstrate the usefulness of our method.

1. Introduction

Suppose $\{X_i\}_{i=1}^n$ are i.i.d. samples in a high dimensional space $\mathbb{R}^d$ drawn from an unknown distribution with density $p(x)$. A general multivariate distribution is typically too complex to analyze directly from the data, thus dimensionality reduction is useful to decrease the complexity of the model (see Cox and Cox, 1994; Schölkopf et al., 1998; Roweis and Saul, 2000; Tenenbaum et al., 2000; Belkin and Niyogi, 2003). Here, our point of departure is the following assumption: the high dimensional data includes low dimensional non-Gaussian components, and the other components are Gaussian. This assumption follows the rationale that in most real-world applications, the 'signal' or 'information' contained in the high-dimensional data is essentially non-Gaussian, while the 'rest' can be interpreted as high dimensional Gaussian noise.

1.1 Setting and General Principle

We want to emphasize from the beginning that we do *not* assume the Gaussian components to be of *smaller* order of magnitude than the signal components; all components are instead typically of *the same amplitude*. This setting therefore excludes the use of dimensionality reduction methods based on the assumption that the data lies, say, on a lower dimensional manifold, up to some small noise. In fact, this type of methods addresses a different kind of problem altogether.

Under our modeling assumption, therefore, the task is to recover the relevant *non-Gaussian* components. Once such components are identified and extracted, various tasks can be applied in the data analysis process, say, data visualization, clustering, denoising or classification.

If the number of Gaussian components is *at most one* and all the non-Gaussian components are mutually independent, *independent component analysis (ICA)* techniques (see, e.g., Comon, 1994; Hyvärinen et al., 2001) are relevant to identify the non-Gaussian subspace. Unfortunately, however, this is often a too strict assumption on the data.

The framework we consider is on the other hand very close to that of *projection pursuit* (denoted PP in short in the sequel) algorithms (Friedman and Tukey, 1974; Huber, 1985; Hyvärinen et al., 2001). The goal of projection pursuit methods is to extract non-Gaussian components in a general setting, i.e., the number of Gaussian components can be more than one and the non-Gaussian components can be dependent.

Projection pursuit methods typically proceed by fixing a *single* index which measures the non-Gaussianity (or 'interessingness') of a projection direction. This index is then optimized over all

possible directions of projection; the procedure can be repeated iteratively (over directions orthogonal to the first ones already found) to find a higher dimensional projection of the data as needed.

However, it is known that some projection indices are suitable for finding super-Gaussian components (heavy-tailed distribution) while others are suited for identifying sub-Gaussian components (light-tailed distribution) (Hyvärinen et al., 2001). Therefore, traditional PP algorithms may not work effectively if the data contains, say, both super- and sub-Gaussian components.

To summarize: existing methods for the setting we consider typically proceed by defining an appropriate interestingness index, and then compute a projection that maximizes this index (projection pursuit methods, and some ICA methods). The philosophy that we would like to promote in this paper is in a sense different: in fact, we do not specify what we are interested in, but we rather define what is *not interesting* (see also Jones and Sibson , 1987). Clearly, a multi-dimensional Gaussian subspace is a reasonable candidate for an undesired component (our idea could be generalized by defining, say, a Laplacian subspace to be uninformative). Having defined this uninteresting subspace, its (orthogonal) complement is by contrast interesting: this therefore precisely defines our target space.

1.2 Presentation of the Method

Technically, our new approach to identifying the non-Gaussian subspace uses a very general semi-parametric framework. The proposed method, called *non-Gaussian component analysis (NGCA)*, is essentially based on a central property stating that there exists a linear mapping $h \mapsto \beta(h) \in \mathbb{R}^d$ which, to any *arbitrary* (smooth) nonlinear function $h : \mathbb{R}^d \to \mathbb{R}$, associates a vector β lying in the non-Gaussian target subspace. In practice, the vector $\beta(h)$ has to be estimated from the data, giving rise to an estimation error. However, our main consistency result shows that this estimation error vanishes at a rate $\sqrt{\log(n)/n}$ with the sample size n. Using a whole family of different nonlinear functions h then yields a family of different vectors $\widehat{\beta}(h)$ which all approximately lie in, and span, the non-Gaussian subspace. We finally perform PCA on this family of vectors to extract the principal directions and estimate the target space.

In practice, we consider functions of the particular form $h_{\omega,a}(x) = f_a(\langle \omega, x \rangle)$, where f is a function class parameterized, say, by a parameter a, and $\|\omega\| = 1$. Even for a fixed a, it is infeasible to compute values of $\beta(h_{\omega,a})$ for all possible values of ω (say, on a discretized net of the unit sphere), because of the cardinality involved. In order to choose a relevant value for ω (still for fixed a), we then opt to use as a heuristic a well-known PP algorithm, FastICA (Hyvärinen, 1999). This was suggested by the surprising observation that the mapping $\omega \to \beta(h_{\omega,a})$ is then *equivalent* to a *single* iteration of FastICA (although this algorithm was built using different theoretical considerations); hence, in this special case, FastICA is exactly the same as iterating our mapping. In short, we use a PP method as a proxy to select the most relevant direction ω for a fixed a. This results in a particular choice of ω_a, to which we apply the mapping once more, thus yielding $\beta_a = \beta(h_{\omega_a,a})$. Finally, we aggregate the different vectors β_a obtained when varying a by applying PCA as indicated previously, in order to recover the target space.

Thus, apart from the conceptual point, defining uninterestingness as the point of departure instead of interestingness, another way to look at our method is to say that it allows the combination of information coming from different indices: here the above function f_a (for fixed a) plays a role similar to that of a non-Gaussianity index in PP, but we do combine a rich family of such functions (by varying a and even by considering several function classes at the same time). The important

point here is that, while traditional projection pursuit does not provide a well-founded justification for combining directions from using different indices, our framework allows to do precisely this – thus implicitly selecting, in a given family of indices, the ones which are the most informative for the data at hand.

In the following section we will outline the theoretical cornerstone of the method, a novel semi-parametric theory for *linear* dimension reduction. Section 3 discusses the algorithmic procedure and is conluded with theoretical results establishing statistical consistency of the method. In Section 4, we study on simulated and real data examples the behavior of the algorithm. A brief conclusion is given in Section 5.

2. Theoretical Framework

In this section, we give a theoretical basis for the non-Gaussian component search within a *semi-parametric* framework. We present a population analysis, where expectations can in principle be calculated exactly, in order to emphasize the main idea and show how the algorithm is built. A more rigorous statistical study of the estimation error will be exposed later in section 3.5.

2.1 Motivation

Before introducing the semi-parametric density model which will be used as a foundation for developing our method, we motivate it by starting from elementary considerations. Suppose we are given a set of observations $X_i \in \mathbb{R}^d,\ (i = 1,\ldots,n)$ obtained as a sum of a signal S and an independent Gaussian noise component N :

$$X = S + N, \tag{1}$$

where $N \sim \mathcal{N}(0,\Gamma)$. Note that no particular structural assumption is made about the noise covariance matrix Γ.

Assume the signal S is contained in a lower-dimensional linear subspace E of dimension $m < d$. Loosely speaking, we would like to project X linearly so as to eliminate as much of the noise as possible while preserving the signal information. An important issue for the analysis of the model (1) is a suitable representation of the density of X which reflects the low dimensional structure of the non-Gaussian signal. The next lemma presents a generic representation of the density p for the model (1).

Lemma 1 *The density $p(x)$ for the model (1) with the m-dimensional signal S and an independent Gaussian noise N can be represented as*

$$p(x) = g(Tx)\phi_\Gamma(x)$$

where T is a linear operator from $\mathbb{R}^d$ to $\mathbb{R}^m$, $g(\cdot)$ is some function on $\mathbb{R}^m$ and $\phi_\Gamma(x)$ is the density of the Gaussian component.

The formal proof of this lemma is given in the Appendix. Note that the above density representation is not unique, as the parameters g, T, Γ are not identifable from the density p. However, the null suspace (kernel) $\mathfrak{K}(T)$ of the linear operator T is an identifiable parameter. In particular, is useful to notice that if the noise N is standard normal, then the operator T can be taken equal to the projector on the signal space E. Therefore, in this case, $\mathfrak{K}(T)$ coincides with $E^\perp$, the orthogonal

complementary subspace to E. In the general situation with "colored" Gaussian noise, the signal space E does not coincide with the orthogonal complementary of the kernel $I = \mathfrak{K}(T)^{\perp}$ of the operator T. However, the density representation of Lemma 1 shows that the the subspace $\mathfrak{K}(T)$ is non-informative and contains only noise. The original data can then be projected orthogonally onto I, which we call the *non-Gaussian subspace*, without loss of information. This way, we are preserving the totality of the signal information. This definition implements the general point of view outlined in the introduction, namely: we define what is considered *uninteresting*; the target space is then defined indirectly as the orthogonal of the uninteresting component.

2.2 Relation to ICA

An equivalent view of the same model is to decompose the noise N appearing in Eq.(1) into a component N_1 belonging to the signal space E and an *independent* component N_2 ; it can then be shown that N_2 belongs to the subspace $\mathfrak{K}(T)$ defined above. In this view, the space I is orthogonal to the independent noise component, and projecting the data onto I amounts to cancelling this independent noise component by an orthogonal projection.

In the present paper, we assume that we wish to project the data *orthogonally*, i.e., that the Euclidean geometry of the input space is meaningful for the data at hand, and that we want to respect it while projecting. An alternative point of view would be to disregard the input space geometry altogether, and to first map the data linearly to a reference space where it has covariance identity ("whitening" transform), which would be closer to a traditional ICA analysis. This would have on the one hand the advantage of resulting in an affine invariant procedure, but, on the other hand, the disadvantage of losing the information of the original space geometry. It is relatively straightforward to adapt the procedure to fit into this framework. For simplicity, we will stick to our original goal of orthogonal projection in the original space.

2.3 Main Model

Based on the above motivation, we assume to be dealing with an unknown probability density function $p(x)$ on $\mathbb{R}^d$ which can put under the form

$$p(x) = g(Tx)\phi_{\Gamma}(x), \tag{2}$$

where T is an unknown linear mapping from $\mathbb{R}^d$ to $\mathbb{R}^m$ with $m \leq d$, g is an unknown function on $\mathbb{R}^m$, and ϕ_{Γ} is a centered[1] Gaussian density with covariance matrix Γ.

Note that the *semi-parametric* model (2) includes as particular cases both the pure parametric ($m = 0$) and purely non-parametric ($m = d$) models. For practical purposes, however, we are effectively interested in an intermediate case where d is large and m is relatively small. In what follows, we denote by I the m-dimensional *linear* subspace in $\mathbb{R}^d$ generated by the adjoint operator T^*:

$$I = \mathfrak{K}(T)^{\perp} = \mathfrak{I}(T^*),$$

where $\mathfrak{I}(\cdot)$ denotes the range of an operator. We call I the *non-Gaussian subspace*.

The proposed goal is therefore to estimate I by some subspace $\widehat{I}$ computed from an i.i.d. sample $\{X_i\}_{i=1}^n$ following the distribution with density $p(x)$. In this paper, we assume the effective

1. It is possible to handle a more general situation where the Gaussian part has an unknown mean parameter θ in addition to the unknown covariance Γ. For simplicity of exposition, we consider here only the case $\theta = 0$.

dimension m to be known or fixed *a priori* by the user. Note that we do *not* estimate Γ nor g when estimating I. We measure the closeness of the two subspaces $\widehat{I}$ and I by the following error function:

$$\mathcal{E}(\widehat{I}, I) = (2m)^{-1} \left\| \Pi_I - \Pi_{\widehat{I}} \right\|_{Frob}^2 = m^{-1} \sum_{i=1}^{m} \| (I_d - \Pi_{\widehat{I}}) v_i \|^2, \tag{3}$$

where Π_I denotes the orthogonal projection on I, $\|\cdot\|_{Frob}$ is the Frobenius norm, $\{v_i\}_{i=1}^m$ is an orthonormal basis of I and I_d is the identity matrix.

2.4 Key Result

The main idea underlying our approach is summed up in the following Proposition (the proof is given in Appendix A.2). Whenever variable X has covariance[2] matrix identity, this result allows, from an *arbitrary* smooth real function h on $\mathbb{R}^d$, to find a vector $\beta(h) \in I$.

Proposition 2 *Let X be a random variable whose density function $p(x)$ satisfies Eq.*(2) *and suppose that $h(x)$ is a smooth real function on $\mathbb{R}^d$. Assume furthermore that $\Sigma = \mathbb{E}\left[XX^\top\right] = I_d$. Then, under mild regularity conditions on h, the following vector $\beta(h)$ belongs to the target space I:*

$$\beta(h) = \mathbb{E}\left[Xh(X) - \nabla h(X)\right]. \tag{4}$$

In the general case where the covariance matrix Σ is different from identity, provided it is non-degenerated, we can apply a whitening operation (also known as Mahalanobis transform). Namely, let us put $Y = \Sigma^{-\frac{1}{2}} X$ the "whitened" data; the covariance matrix of Y is then identity. Note that if the density function of X is of the form

$$p(x) = g(Tx)\phi_\Gamma(x),$$

then by change of variable the density function of $Z = AX$ is given by

$$q(z) = c_A g(TA^{-1}z)\phi_{A\Gamma A^\top}(z),$$

where c_A is a normalization constant depending on A.

This identity applied to $A = \Sigma^{-\frac{1}{2}}$ and the previous proposition allow to conclude that

$$\beta_Y(h) = \mathbb{E}\left[\nabla h(y) - y h(y)\right] \in \mathcal{J} = \Im(\Sigma^{\frac{1}{2}} T^*)$$

and therefore that

$$\gamma(h) = \Sigma^{-\frac{1}{2}} \beta_Y(h) \in I = \Im(T^*),$$

where I is the non-Gaussian index space for the initial variable X, and $\mathcal{J} = \Sigma^{\frac{1}{2}} I$ the transformed non-Gaussian space for the whitened variable Y.

2. Here and in the sequel, with some abuse we call $\Sigma = \mathbb{E}\left[XX^\top\right]$ the *covariance matrix*, even though we do not assume the non-Gaussian part of the data to be centered.

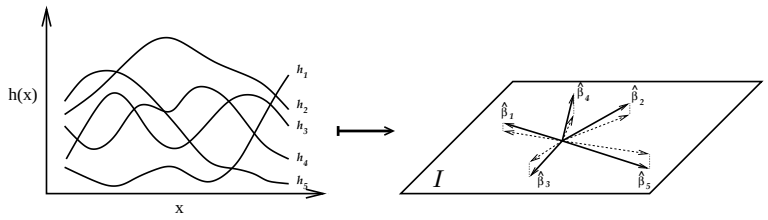


Figure 1: The NGCA main idea: from a varied family of real functions, compute a family of vectors belonging to the target space up to small estimation error.

3. Procedure

We now use the key proposition established in the previous section to design a practical algorithm in order to identify the non-Gaussian subspace. The first step is to apply the whitening transform to the data (where the true covariance matrix Σ is estimated by the empirical covariance $\widehat{\Sigma}$). We then estimate the "whitened" non-Gaussian space $\mathcal{J}$ by some $\widehat{\mathcal{J}}$ (this will be described next); this space is then finally pulled back in the original space by application of $\widehat{\Sigma}^{-\frac{1}{2}}$. To simplify the exposition, in this section we will forget about the whitening/dewhitening steps and always implicitly assume that we are dealing directly with the whitened data: every time we refer to the non-Gaussian space it is therefore to be understood that we refer to $\mathcal{J} = \Sigma^{\frac{1}{2}} I$, corresponding to the whitened data Y.

3.1 Principle of the Method

In the previous section, we have proved that for an arbitrary function h satisfying mild smoothness conditions, it is possible to construct a vector $\beta(h)$ which lies in the non-Gaussian subspace. However, since the unknown density $p(x)$ is used (via the expectation operator) to define β by Eq.(2), one cannot directly use this formula in practice: it is then natural to approximate it by replacing the true expectation by the empirical expectation. This gives rise to the estimated vector

$$\widehat{\beta}(h) = \frac{1}{n}\sum_{i=1}^{n} Y_i h(Y_i) - \nabla h(Y_i)\,, \tag{5}$$

which we expect to be close to the non-Gaussian subspace up to some estimation error. At this point, the natural next step is to consider a whole family of functions $\{h_i\}_{i=1}^{n}$, giving rise to an associated vector family of $\{\widehat{\beta}_i\}_{i=1}^{L}$, all lying close to the target subspace, where $\widehat{\beta}_i := \widehat{\beta}(h_i)$. The final step is to recover the non-Gaussian subspace from this set. For this purpose, we suggest to use the principal directions of this family, i.e. to apply PCA (although other algorithmic options are certainly avalaible for this task). This general idea is illustrated on Figure 1.

3.2 Normalization of the Vectors

When extracting information on the target subspace from the set of vectors $\{\widehat{\beta}_i\}_{i=1}^{L}$, attention should be paid to how the functions $\{h_i\}_{i=1}^{L}$ are normalized. As can be seen from its definition, the operator which maps a function h to $\beta(h)$ (and also its empirical counterpart $\widehat{\beta}(h)$) is linear. Therefore, if, for example, one of the functions $\{h_i\}_{i=1}^{L}$ is multiplied by an arbitrarily large scalar, the associ-

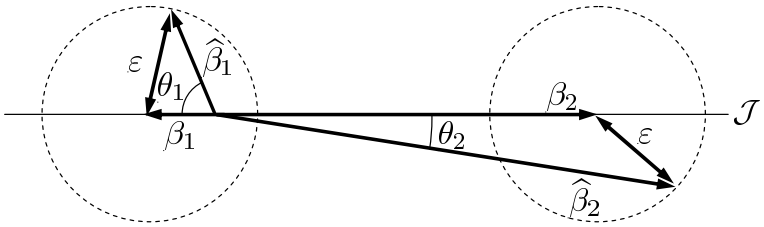


Figure 2: For the same estimation error represented as a confidence ball of radius ε, estimated vectors with higher norm give a more precise information about the true target space.

ated $\widehat{\beta}(h)$ could have an arbitrarily large norm: this is likely to influence heavily the procedure of principal direction extraction applied to the whole family.

To prevent this problem, the functions $\{h_i\}_{i=1}^L$ should be normalized in some way or other. Several possibilities can come to mind, like using the supremum or L_2 norm of h or of ∇h. We argue here that a sensible way to normalize functions is such that the average squared deviation (estimation error) of $\widehat{\beta}(h)$ to its mean is of the same order for all functions h considered. This has a first direct intuitive interpretation in terms of making the length of each estimated vector proportional to its associated signal-to-noise ratio. We argue in more detail that the norm of $\widehat{\beta}(h)$ after normalization is directly linked to the amount of information brought by this vector about the target subspace.

Namely, if we measure the information that is brought by a certain vector $\widehat{\beta}(h)$ about the target space $\mathcal{J}$ through the angle $\theta(\widehat{\beta}(h))$ between the vector and the space, we have

$$\|\widehat{\beta}(h) - \beta(h)\| \geq dist(\widehat{\beta}(h), \mathcal{J}) = \sin(\theta(\widehat{\beta}(h)))\|\widehat{\beta}(h)\| . \tag{6}$$

Suppose we have ensured by renormalization that $\sigma(h)^2 = \mathbb{E}\left[\|\widehat{\beta}(h) - \beta(h)\|^2\right]$ is constant and independent of h, and assume that this results in $\|\widehat{\beta}(h) - \beta(h)\|^2$ being bounded by some constant with high probability. It entails that $\sin(\theta(\widehat{\beta}(h)))\|\widehat{\beta}(h)\|$ is bounded independently of h. We expect, in this situation, that the bigger $\|\widehat{\beta}\|$, the smaller is $\sin(\theta)$, and therefore the more reliable the information about $\mathcal{J}$. This intuition is illustrated in Figure 2, where the estimation error is represented by a confidence ball of equal size for all vectors.[3]

Therefore, at least at an intuitive level, it appears appropriate to use $\sigma(h)$ as a renormalization. Note that this is just the square root of the trace of the covariance matrix of $\widehat{\beta}(h)$, and therefore easy to estimate in practice from its empirical counterpart. In section 3.5, we give actual theoretical confidence bounds for $\|\beta - \widehat{\beta}\|$ which justify this intuition in a more rigorous manner.

Finally, to confirm this idea on actual data, we plot in the top row Figure 3 the distribution of $\widehat{\beta}$ on an illustrative data set using the normalization scheme just described. In order to investigate

3. Of course, the situation depicted in Figure 2 is idealized: we actually expect (from the Central Limit Theorem) that $\beta - \widehat{\beta}$ has approximately a Gaussian distribution with some non-spherical variance, giving rise to a confidence ellipsoid rather than a confidence ball. To obtain a spherical error ball, we would have to apply a (linear) error whitening transform separately to each $\widehat{\beta}(h)$. However, in that case the error whitening transform would be different for each h, and the information of the vector family about the target subspace would then be lost. To preserve this information, only a scalar normalization is adequate, which is why we recommend the normalization scheme explained here.

the relation between the norm of the (normalized) $\widehat{\beta}$ and the amount of information on the non-Gaussian subspace brought by $\widehat{\beta}$, we plot in the right part of Figure 3 the relation between $\|\widehat{\beta}\|$ and $\|\Pi_{\mathcal{J}}\widehat{\beta}\|/\|\widehat{\beta}\| = \cos(\theta(\widehat{\beta}))$. As expected, the vectors $\widehat{\beta}$ with highest norm are indeed much closer to the non-Gaussian subspace in general. Furthermore, vectors $\widehat{\beta}$ with norm close to zero appear to bear almost no information about the non-Gaussian space, which is consistent with the setting depicted in Figure 2: whenever an estimated vector $\widehat{\beta}$ has norm smaller than the estimation error ε, its confidence ball contains the origin, which means that it brings no useable information about the direction of the non-Gaussian subspace.

These findings motivate two important points for the algorithm:

1. It should be beneficial to *actively search* for functions h which yield an estimated $\widehat{\beta}(h)$ with higher norm, since these are more informative about the target space $\mathcal{J}$;
2. The vectors $\widehat{\beta}$ with norm below a certain threshold ε can be discarded as they are non-informative. So far, the theoretical bounds presented below in section 3.5 are not precise enough to give a canonical value for this threshold: we therefore recommend that it be determined by a preliminary calibration procedure. For this, we consider independent Gaussian data: in this case, $\beta = 0$ for any h and thus $\|\widehat{\beta}\|$ represents pure estimation noise. A reasonable choice for the threshold is therefore the 95th percentile (say) of this distribution, which we expect to reject a large majority of the noninformative vectors.

3.3 Using FastICA as Preprocessing to Find Promising Functions

When considering a parametrized family of functions $\{h_\omega\}$, it is a desirable goal to search the parameter space to find indices ω such that $\widehat{\beta}(h_\omega)$ has a high norm, as proposed in the last section. From now on we will restrict our attention to functions of the form

$$h_\omega(x) = f(\langle \omega, x \rangle), \tag{7}$$

where $\omega \in \mathbb{R}^d$, $\|\omega\| = 1$, and f is a smooth real function of a real variable. Clearly, it is not feasible to sample the entire parameter space for ω as soon as it has more than a few dimensions, and it is not obvious *a priori* to find parameters ω such that $\widehat{\beta}(h_\omega)$ has a high norm. Remember however that we do not need to find an *exact maximum* of this norm over the parameter space. We merely want to find parameters such that the associated norm is preferably high, because they bring more information; this may also involve heuristics. Naturally, good heuristics should be able to find parameters giving rise to vectors with higher norm, bringing more information on the subspace and ultimately better practical results; nevertheless, the underlying theoretical motivation stays unchanged regardless of the way the functions are picked.

A particularly relevant heuristic for choosing ω comes naturally with a closer look at Eq.(5) when we plug in functions of the specific form given by Eq.(7):

$$\widehat{\beta}(h_\omega) = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i f(\langle \omega, Y_i \rangle) - f'(\langle \omega, Y_i \rangle)\omega\right). \tag{8}$$

It is interesting to notice that this equation precisely coincides with *one* iteration of a well-known projection pursuit algorithm, FastICA (Hyvärinen, 1999). More precisely, FastICA consists in iter-

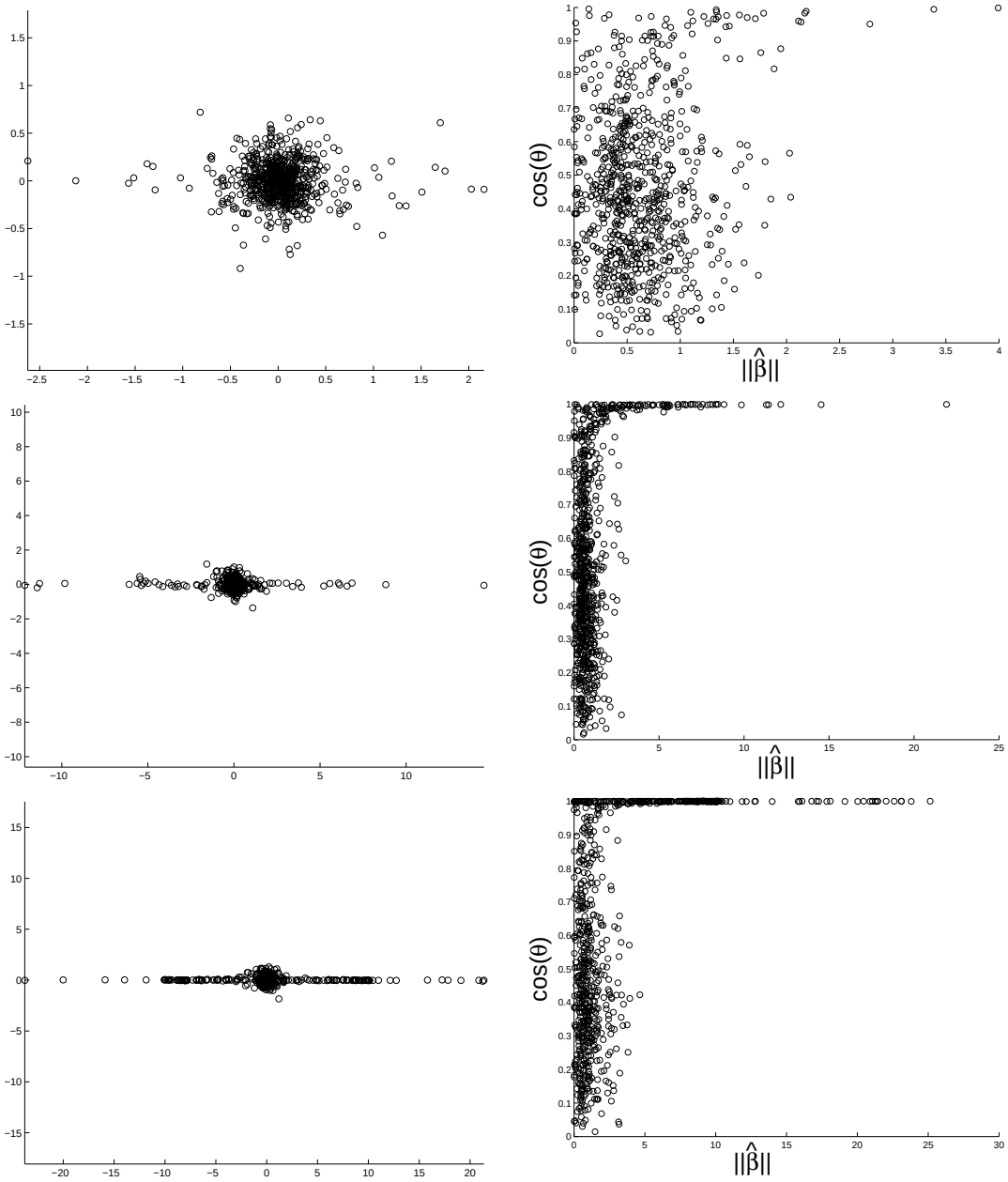


Figure 3: Illustrative plots of the method, applied to toy data of type **(A)** (See section 4.1). Left column: Distribution of $\widehat{\beta}$ projected on a direction belonging to the target space $\mathcal{J}$ (abscissa) and a direction orthogonal to it (ordinate). Right column: $\|\widehat{\beta}\|$ (after normalization) vs. $\cos(\theta(\widehat{\beta},\mathcal{J}))$. From top to bottom rows: random draw of functions, after 1-step, and after 4-step of FastICA preprocessing. $\widehat{\beta}$'s are normalized as described in section 3.2.

ating the following update rule to form a sequence $\omega_1, \ldots, \omega_T$:

$$\omega_{t+1} \propto \frac{1}{n} \sum_{i=1}^{n} \left(Y_i f(\langle \omega_t, Y_i \rangle) - f'(\langle \omega_t, Y_i \rangle) \omega_t \right) \tag{9}$$

where the sign $\propto$ indicates that vector ω_{t+1} is renormalized to be of unit norm.

Note that the FastICA procedure is derived from quite a different theoretical setting of what we considered here (see, e.g., Hyvärinen et al., 2001); its goal is in principle to optimize a non-Gaussianity measure $\mathbb{E}[F(\langle \omega, x \rangle)]$ (where F is such that F' formally coincides with our f above) and the solution is reached by an approximate Newton method giving rise to the update rule of Eq.(9), repeated until convergence.

This formal identity leads us to adopt the FastICA methodology as a heuristic for our method. Since finding an actual optimum point is not needed, convergence is not an issue, so that we only iterate the update rule of Eq.(9) for a fixed number of iterations T to find a relevant direction ω_T. Finally we apply Eq.(8) one more time to this choice of parameter, so that the procedure finally outputs $\widehat{\beta}(h_{\omega_T})$. On Figure 3, we plot the effect of a few iterations of this preprocessing for the method, applied on toy data and see that it leads to a significant improvement.

Paradoxically, if the convergence of this FastICA preprocessing is too good, there is in principle a risk that all vectors $\widehat{\beta}$ end up in the vicinity of one single "best" direction instead of spanning the whole target space: the preprocessing would then have the opposite effect of what is wished, namely impoverishing the vector family. One possible remedy against this is to apply so-called batch FastICA, which consists in iterating equation (9) on a m-dimensional system of vectors, which is orthonormalized anew before each new iteration. In our practical experiments we did not observe any significant change in the results when using this refinement, so we mention this possibility only as a matter of precaution. We suspect two mitigating factors against this possible unwished behavior are that (1) it is known that FastICA does not converge to a global maximum, so that we probably find vectors in the vicinity of different local optima and (2) the "optimal" directions depend on the function f used and we combine a large number of such functions.

In the next section, we will describe the full algorithm, which consists in applying the procedure just described to different choices of the function f. Since we are using projection pursuit as a heuristic to find suitable parameters ω for a fixed f, the theoretical setting proposed here can therefore also be seen as a suitable framework for combining projection pursuit results when using different index functions f.

3.4 Full Procedure

The previous sections have been devoted to detailing some key points of the procedure. We now gather these points and describe the full algorithm. We previously considered the case of a basis function family $h_\omega(y) = f(\langle \omega, y \rangle)$. We now consider a finite family of possible choices $\{f_k\}_{k=1}^{L}$ which are then combined.

In the implementation tested, we have used the following forms of the functions f_k:

$$f_\sigma^{(1)}(z) = z^3 \exp\left(-\frac{z^2}{2\sigma^2}\right), \qquad \text{(Gauss-Pow3)}$$

$$f_b^{(2)}(z) = \tanh(bz), \qquad \text{(Hyperbolic Tangent)}$$

$$f_a^{(3)}(z) = \exp(iaz), \qquad \text{(Fourier}^4\text{)}$$

More precisely, we consider discretized ranges for $\sigma \in [\sigma_{\min}, \sigma_{\max}]$, $b \in [0, B]$, and $a \in [0, A]$, giving rise to a finite collection $\{f_k\}$ (which therefore includes *simultaneously* functions of the three different above families). Note that using z^3 and Hyperbolic Tangent functions is inspired by the classical PP algorithms (including FastICA) where these indices are used. We multiplied z^3 by a Gaussian factor in order to satisfy the boundedness assumption needed to control the estimation error (see Theorem 3 and 4 below). Furthermore, the introduction of the parameter σ^2 allows for a richer family. Finally, the Fourier functions were introduced as they constitute a rich and important family. A pseudocode for the NGCA algorithm is described in Figure 4.

3.5 Theoretical Bounds on the Statistical Estimation Error

In this section we tackle the question of controlling the estimation error when approximating the vectors $\beta(h)$ by their empirical estimations $\widehat{\beta}(h)$ from a rigorous theoretical point of view. These results were derived with the following goals in mind:

- A cornerstone of the algorithm is that we consider a whole family $h_1, \ldots, h_L$ of functions and pick selected members from it. In order to justify this from a statistical point of view, we therefore need to control the estimation error not for a single function h and the associated $\widehat{\beta}(h)$, but instead uniformly over the function family. For this, a simple control of, e.g., the averaged squared deviation $\mathbb{E}\left[\|\beta - \widehat{\beta}\|^2\right]$ for each individual h is not sufficient: we need a stronger result, namely an exponential control of the deviation probability. This allows, by the union bound, to obtain a uniform control over the whole family with a mild (logarithmic) dependence on the cardinality of the family.

- We aim at making the covariance trace $\widehat{\sigma}^2$ directly appear into the main bounding terms of our error control. This provides a more solid justification to the renormalization scheme developed in section 3.2, where we have used arguments based on a non rigorous intuition. The choice to involve directly the *empirical* covariance in the bound instead of the population one was made to emphasize that estimation error for the covariance itself is also taken into account for the bound.

- While the control of the deviation of an empirical average of the form given in Eq.(5) is a very classical problem, we want to explicitly take into account the effect of the empirical whitening/dewhitening using the empirical covariance matrix $\widehat{\Sigma}$. This complicates matters noticeably since this whitening is itself data-dependent.

- Our goal was *not* to obtain tight confidence intervals or even exact asymptotical behavior. There is a number of ways in which our results could be substantially refined, for example obtaining uniform bounds over continuous (instead of finite) families of functions using covering number arguments; showing asymptotical uniform central limit properties for a precise study of the typical deviations, etc. Here, we tried to obtain simple, while still mathematically rigorous, results, covering essential statistical foundations of our method: consistency and order of the convergence rate.

In the sequel, for a matrix A, we denote $\|A\|$ its operator norm.

4. In practice, separated into real and complex parts (sine and cosine).

Input: Data points $(X_i) \in \mathbb{R}^d$, dimension m of target subspace.
Parameters: Number T of FastICA iterations; threshold ε.

Whitening.
The data X_i is recentered by subtracting the empirical mean.
Let $\widehat{\Sigma}$ denote the empirical covariance matrix of the data sample (X_i) ;
put $\widehat{Y}_i = \widehat{\Sigma}^{-\frac{1}{2}} X_i$ the empirically whitened data.
Main Procedure.
Loop on $k = 1, \ldots, L$:
Draw ω_0 at random on the unit sphere of $\mathbb{R}^d$.
Loop on $t = 1, \ldots, T$: *[FastICA loop]*
Put $\widehat{\beta}_t \leftarrow \frac{1}{n} \sum_{i=1}^{n} \left(\widehat{Y}_i f_k(\langle \omega_{t-1}, \widehat{Y}_i \rangle) - f_k'(\langle \omega_{t-1}, \widehat{Y}_i \rangle) \omega_{t-1} \right)$.
Put $\omega_t \leftarrow \widehat{\beta}_t / \|\widehat{\beta}_t\|$.
End Loop on t
Let N_k be the trace of the empirical covarariance matrix of $\widehat{\beta}_T$:
$N_k = \frac{1}{n} \sum_{i=1}^{n} \left\| \widehat{Y}_i f_k(\langle \omega_{T-1}, \widehat{Y}_i \rangle) - f_k'(\langle \omega_{T-1}, \widehat{Y}_i \rangle) \omega_{T-1} \right\|^2 - \left\| \widehat{\beta}_T \right\|^2$.
Store $v^{(k)} \leftarrow \widehat{\beta}_T * \sqrt{n/N_k}$. *[Normalization]*
End Loop on k
Thresholding.
From the family $v^{(k)}$, throw away vectors having norm smaller than threshold ε.
PCA step.
Perform PCA on the set of remaining $v^{(k)}$.
Let $\widehat{\mathcal{J}}$ be the space spanned by the first m principal directions.
Pull back in original space.
$\widehat{I} = \widehat{\Sigma}^{-\frac{1}{2}} \widehat{\mathcal{J}}$.

Output: $\widehat{I}$.

Figure 4: Pseudocode of the NGCA algorithm.

Analysis of the estimation error with exact whitening. We start by considering an idealized case where whitening is done using the true covariance matrix Σ: $Y = \Sigma^{-\frac{1}{2}} X$.

In this case we have the following control of the estimation error:

Theorem 3 *Let $\{h_k\}_{k=1}^L$ be a family of smooth functions from $\mathbb{R}^d$ to $\mathbb{R}$. Assume that $\sup_{y,k} \max(\|\nabla h_k(y)\|, \|h_k(y)\|) < B$ and that X has covariance matrix Σ with $\left\|\Sigma^{-1}\right\| \leq K^2$, and is such that for some $\lambda_0 > 0$ the following inequality holds:*

$$\mathbb{E}\left[\exp\left(\lambda_0 \|X\|\right)\right] \leq a_0 < \infty. \tag{10}$$

Denote $\widetilde{h}(y) = yh(y) - \nabla h(y)$. Suppose $X_1, \ldots, X_n$ are i.i.d. copies of X and let $Y_i = \Sigma^{-\frac{1}{2}} X_i$. If we define

$$\widehat{\beta}_Y(h) = \frac{1}{n} \sum_{i=1}^n \widetilde{h}(Y_i) = \frac{1}{n} \sum_{i=1}^n Y_i h(Y_i) - \nabla h(Y_i), \tag{11}$$

and

$$\widehat{\sigma}_Y^2(h) = \frac{1}{n} \sum_{i=1}^n \left\| \widetilde{h}(Y_i) - \widehat{\beta}_Y(h) \right\|^2, \tag{12}$$

then for any $\delta < \frac{1}{4}$, with probability at least $1 - 4\delta$ the following bounds hold simultaneously for all $k \in \{1, \ldots, L\}$:

$$dist\left(\widehat{\beta}_Y(h_k), \mathcal{J}\right) \leq 2\sqrt{\widehat{\sigma}_Y^2(h_k) \frac{\log(L\delta^{-1}) + \log d}{n}} + C\left(\frac{\log(nL\delta^{-1}) \log(L\delta^{-1})}{n^{\frac{3}{4}}}\right),$$

and

$$dist\left(\Sigma^{-\frac{1}{2}} \widehat{\beta}_Y(h_k), I\right) \leq 2K\sqrt{\widehat{\sigma}_Y^2(h_k) \frac{\log(L\delta^{-1}) + \log d}{n}} + C'\left(\frac{\log(nL\delta^{-1}) \log(L\delta^{-1})}{n^{\frac{3}{4}}}\right),$$

where $dist(\gamma, I)$ denotes the distance between a vector γ and the subspace I, and C, C' are constants depending only on the parameters $(d, \lambda_0, a_0, B, K)$.

Comments.

1. The above inequality tells us that the rate of convergence of the estimated vectors to the target space is in this case of order $n^{-1/2}$ (classical "parametric" rate). Furthermore, the theorem gives us an estimation of the relative size of the estimation error for different functions h through the empirical factor $\widehat{\sigma}_Y(h)$ in the principal term of the bound. As announced in our initial goals, this therefore gives a rigorous foundation to the intuition exposed in section 3.2 for vector renormalization.
2. Also following our goals, we obtained a uniform control of the estimation error over a finite family with a logarithmic dependence in the cardinality. This does not correspond exactly to the continuous families we use in practice but comes close enough if we consider adequate parameter discretization. We will comment on this in more detail after the next theorem.

Whitening using empirical covariance. When Σ is unknown (which is in general the case), we use instead the empirical covariance matrix $\widehat{\Sigma}$. Here, we will show that, under a somewhat stronger assumption on the distribution of X and on the functions h, we are still able to obtain a convergence rate of order at most $\sqrt{\log(n)/n}$ towards the index space I.

Let us denote $\widehat{Y}_i = \widehat{\Sigma}^{-\frac{1}{2}} X_i$ the empirically whitened datapoints, $\widetilde{h}(y) = yh(y) - \nabla h(y)$ as previously, and

$$\widehat{\beta}_{\widehat{Y}}(h) = \frac{1}{n}\sum_{i=1}^{n} \widetilde{h}(\widehat{Y}_i) = \frac{1}{n}\sum_{i=1}^{n} \widehat{Y}_i h(\widehat{Y}_i) - \nabla h(\widehat{Y}_i); \tag{13}$$

finally, let us denote

$$\widehat{\gamma}(h) = \widehat{\Sigma}^{-\frac{1}{2}} \widehat{\beta}_{\widehat{Y}}(h), \qquad \text{and} \qquad \widehat{\sigma}^2_{\widehat{Y}}(h) = \frac{1}{n}\sum_{i=1}^{n} \left\| \widetilde{h}(\widehat{Y}_i) - \widehat{\beta}_{\widehat{Y}}(h) \right\|^2.$$

We then have the following theorem:

Theorem 4 *Let us assume the following :*
(i) There exists $\lambda_0 > 0, a_0 > 0$ *such that*

$$\mathbb{E}\left[\exp\left(\lambda_0 \|X\|^2\right)\right] = a_0 < \infty;$$

(ii) The covariance matrix Σ *of* X *is such that* $\left\|\Sigma^{-1}\right\| \leq K^2$;
(iii) $\sup_{k,y} \max\left(\|\nabla h_k(y)\|, \|h_k(y)\|\right) < B$;
(iv) The functions $\widetilde{h}_k(y) = \nabla h_k(y) - y h_k(y)$ *are all Lipschitz with constant* M.

Then for big enough n, *with probability at least* $1 - \frac{4}{n} - 4\delta$ *the following bounds hold true simultaneously for all* $k \in \{1, \ldots, L\}$ *:*

$$dist(\widehat{\beta}_{\widehat{Y}}(h_k), \mathcal{J}) \leq C_1 \sqrt{\frac{d \log n}{n}} + 2\sqrt{\widehat{\sigma}^2_{\widehat{Y}}(h_k) \frac{\log(L\delta^{-1}) + \log d}{n}} + C_2 \frac{\log(nL\delta^{-1}) \log(L\delta^{-1})}{n^{\frac{3}{4}}},$$

and

$$dist(\widehat{\gamma}(h_k), I) \leq C_1' \sqrt{\frac{d \log n}{n}} + 2K\sqrt{\widehat{\sigma}^2_{\widehat{Y}}(h_k) \frac{\log(L\delta^{-1}) + \log d}{n}} + C_2' \frac{\log(nL\delta^{-1}) \log(L\delta^{-1})}{n^{\frac{3}{4}}},$$

where C_1, C_1' *are constants depending on parameters* $(\lambda_0, a_0, B, K, M)$ *only and* C_2, C_2' *on* $(d, \lambda_0, a_0, B, K, M)$.

Comments.

1. Theorem 4 implies that the vectors $\widehat{\gamma}(h_k)$ obtained from any $h(x)$ converge to the unknown non-Gaussian subspace I *uniformly* at a rate of order $\sqrt{\log(n)/n}$.
2. The condition (i) is a restrictive assumption as it excludes some densities with heavy tails. We are considering weakening this assumption in future developments.

3. In the actual algorithm, we consider a family of functions of the form $h_\omega(x) = f(\langle \omega, x \rangle)$, with ω on the unit sphere of $\mathbb{R}^d$. Suppose we approximate ω by its nearest neighbor $\widetilde{\omega}$ on a regular grid of scale ε. Then we only have to apply the bound to a discretized family of size $L = O(\varepsilon^{1-d})$, giving rise only to an additional factor in the bound of order $\sqrt{d \log \varepsilon^{-1}}$. Taking for example $\varepsilon = 1/n$ (the fact that the function family depends on n is not a problem since the bounds are valid for any fixed n), this ensures convergence of the discretized functions to the initial continuous family while introducing only in an additional factor $\sqrt{d \log n}$ in the bound: this does not change fundamentally the order of the bound since there is already another $\sqrt{d \log n}$ term present.

4. For both Theorems 3 and 4, we have given bounds for estimation of both I and $\mathcal{J}$, that is, in terms of the initial data and of the "whitened" data. The result in terms of the initial data ensures the overall consistency of the approach, but the convergence in the whitened space is equally interesting since we use it as the main working space for the algorithm and the bound itself is more precise.

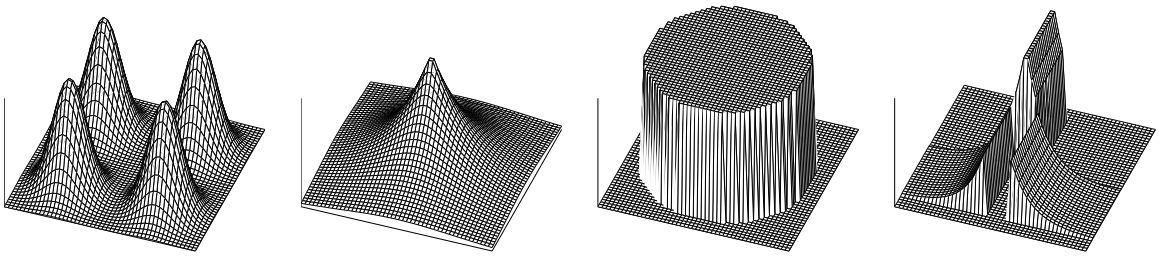
5. Comparing to Theorem 3 obtained for exact whitening, we see in the present case that there is an additional term of principal order in n coming from the estimation error of Σ, with a multiplicative factor which unfortunately is not known accurately. This means that the renormalization scheme is not completely justified in this case, although we feel the idealized situation of Theorem 3 already provides some strong argument in this direction. However, the present result suggests that the accuracy of the normalization could probably be further improved.

4. Numerical Results

We now turn to numerical evaluations of the NGCA method: first on simulated data, where the generating distribution is precisely known, then on exemplary, realistic data. *All* of the experiments presented below, without exception, where obtained with exactly the *same* set of parameters: $a \in [0,4]$ for the Fourier functions; $b \in [0,5]$ for the Hyperbolic Tangent functions; $\sigma^2 \in [0.5,5]$ for the Gauss-pow3 functions. Each of these ranges was divided into 1000 equispaced values, thus yielding a family $\{f_k\}$ of size 4000 (Fourier functions count twice because of the sine and cosine parts). The preliminary calibration procedure described in the end of section 3.2 suggested to take $\varepsilon = 1.5$ as the threshold under which vectors are not informative (strictly speaking, the threshold should be calibrated separately for each function f but we opted here for a single threshold for simplicity). Finally we fixed the number of FastICA iterations $T = 10$. With this choice of parameters and 1000 data points in the sample, the computation time is typically of the order of less than 10 seconds on a modern PC under our Matlab implementation.

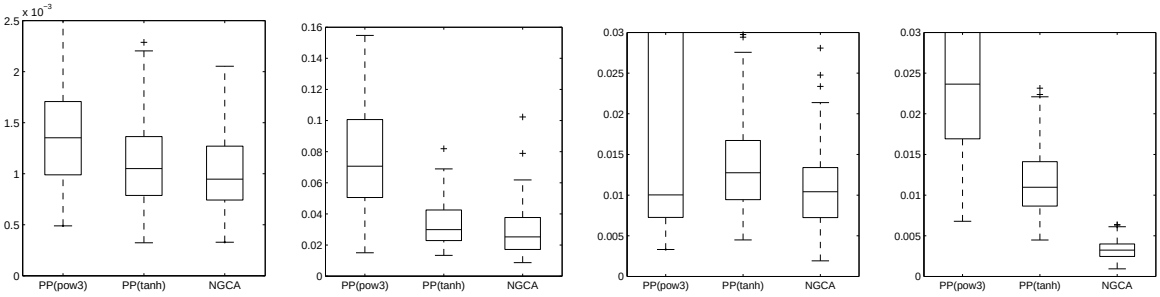
4.1 Tests in a Controlled Setting

For testing our algorithm and comparing it with PP, we performed numerical experiments using various synthetic data. Here, we report exemplary results using the following 4 data sets. Each data set includes 1000 samples in 10 dimensions. The generating distribution consists in 8 independent standard Gaussian components and 2 non-Gaussian components generated as follows:



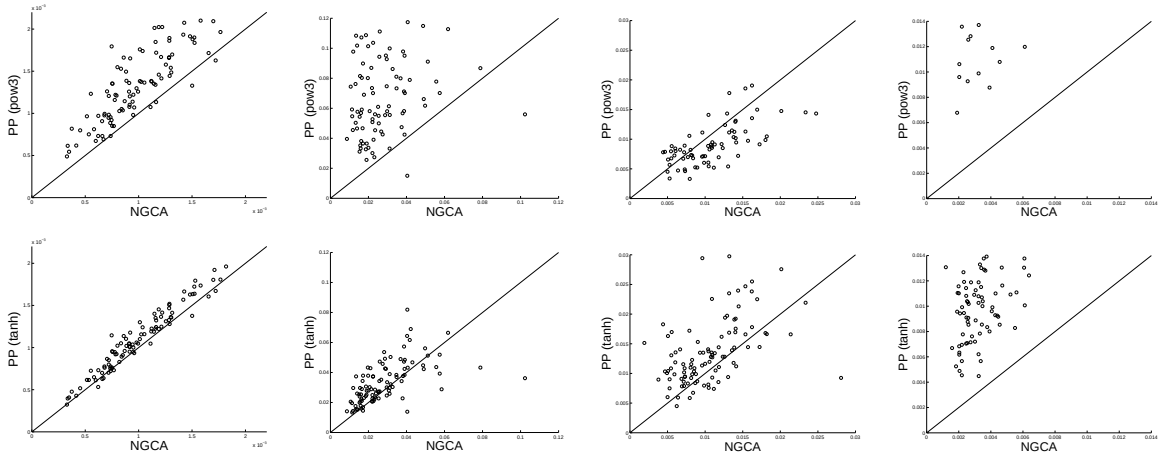
(A) (B) (C) (D)

Figure 5: Densities of non-Gaussian components. The data sets are: (a) 2D independent Gaussian mixtures, (b) 2D isotropic super-Gaussian, (c) 2D isotropic uniform and (d) dependent 1D Laplacian + 1D uniform.



(A) (B) (C) (D)

Figure 6: Boxplots of the error criterion $\mathcal{E}(\widehat{I}, I)$.



(A) (B) (C) (D)

Figure 7: Performance comparison plots (for error criterion $\mathcal{E}(\widehat{I}, I)$) of NGCA versus FastICA; top: versus pow3 index; bottom: versus tanh index.

(A) Simple Gaussian Mixture: 2-dimensional independent Gaussian mixtures, with the density of each component given by

$$\frac{1}{2}\phi_{-3,1}(x)+\frac{1}{2}\phi_{3,1}(x)\,. \tag{14}$$

(B) Dependent super-Gaussian: 2-dimensional isotropic distribution with density proportional to $\exp(-\|x\|)$.

(C) Dependent sub-Gaussian: 2-dimensional isotropic uniform with constant positive density for $\|x\| \leq 1$ and 0 otherwise.

(D) Dependent super- and sub-Gaussian: 1-dimensional Laplacian with density proportional to $\exp(-|x_{Lap}|)$ and 1-dimensional dependent uniform $U(c,c+1)$, where $c=0$ for $|x_{Lap}| \leq \log 2$ and $c=-1$ otherwise.

For each of these situations, the non-Gaussian components are additionally rescaled coordinatewise by a fixed factor so that each coordinate has unit variance. The profiles of the density functions of the non-Gaussian components in the above data sets are described in Figure 5.

We compare the following three methods in the experiments: PP with 'pow3' or 'tanh' index[5] (denoted by PP(pow3) and PP(tanh), respectively), and the proposed NGCA.

Figure 6 shows boxplots of the error criterion $\mathcal{E}(\widehat{I},I)$ defined in Eq.(3) obtained from 100 runs. Figure 7 shows comparison of the errors obtained by different methods for each individual trial. Because PP tends to get trapped into local optima of the index function it optimizes, we restarted it 10 times with random starting points and took the subspace obtaining the best index value. However, even when it is restarted 10 times, PP (especially with the 'pow3' index) still gets caught in local optima in a small percentage of cases (we also tried up to 500 restarts but it led to negligible improvement).

For the simplest data set (A), NGCA is comparable or slightly better than PP methods. It is known that PP(tanh) is suitable for finding super-Gaussian components (heavy-tailed distribution) while PP(pow3) is suitable for finding sub-Gaussian components (light-tailed distribution) (Hyvärinen et al., 2001). This can be observed in the data sets (B) and (C): PP(tanh) works well for the data set (B) and PP(pow3) works well for the data set (C), although the upper-quantile is very large for the data set (C) (because of PP getting trapped in local minima). The sample-wise plots of Figure 7 confirm that NGCA is on average on par with, or slightly better than, PP with the 'correct' non-Gaussianity index, without having to prefix such a non-Gaussianity index. For the data set (C), NGCA appears to be marginally worse than PP(pow3) (excluding those cases where PP fails due to local minima: the corresponding points are outside the range of the figure), but the difference appears hardly significant. The superiority of the index adaptation feature of NGCA can be clearly observed in the data set (D), which includes both sub- and super-Gaussian components. Because of this composition, there is no single best non-Gaussianity index for this data set, and the proposed NGCA gives significantly lower error than that of either PP method.

5. We used the deflation mode of the FastICA algorithm (Hyvärinen et al., 2001) as an implementation of PP. The 'pow3' flavor is equivalent to a kurtosis based index: in other words, in this case, FastICA iteratively maximizes the kurtosis. On the other hand, the 'tanh' flavor uses a robust index which is appropriate in particular for heavy-tailed data.

Failure modes. We now try to explore the limits of the method and the conditions under which estimation of the target space will fail. First, we study the behaviour of NGCA again compared with PP as the total dimension of the data increases. We use the same synthetic data sets with 2-dimensional non-Gaussian components, while the number of Gaussian components increases. The averaged errors over 100 experiments are depicted in Figure 8. In all cases, we seem to observe a sharp phase transition between a good behaviour regime and a failure mode where the procedure is unable to estimate the correct subspace. In 3 out of 4 cases, however, we observe that the phase transition to the failure mode occurs for a higher dimension for NGCA than for the PP methods, which indicates better robustness of NGCA.

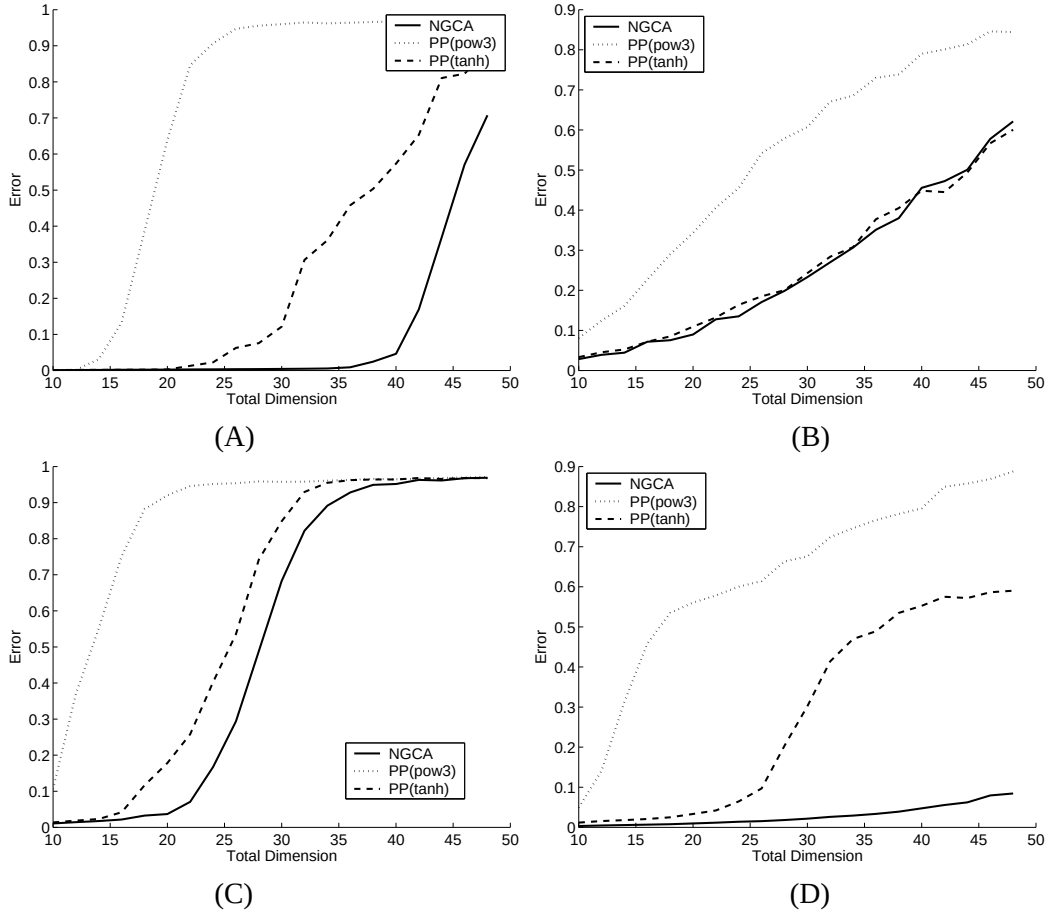


Figure 8: Results when the total dimension of the data increases.

In the synthetic data sets used so far, the data was always generated with a covariance matrix equal to identity. Another interesting setting to study is the robustness with respect to bad conditioning of the covariance matrix. We consider again a fixed-dimension setting, with 2 non-Gaussian and 8 gaussian dimensions.

While the non-Gaussian coordinates always have variance unity, the standard deviation of the 8 Gaussian dimensions now follows the geometrical progression $10^{-r}, 10^{-r+2r/7}, \ldots, 10^{r}$. Thus, the higher r, the worse conditioned is the total covariance matrix.

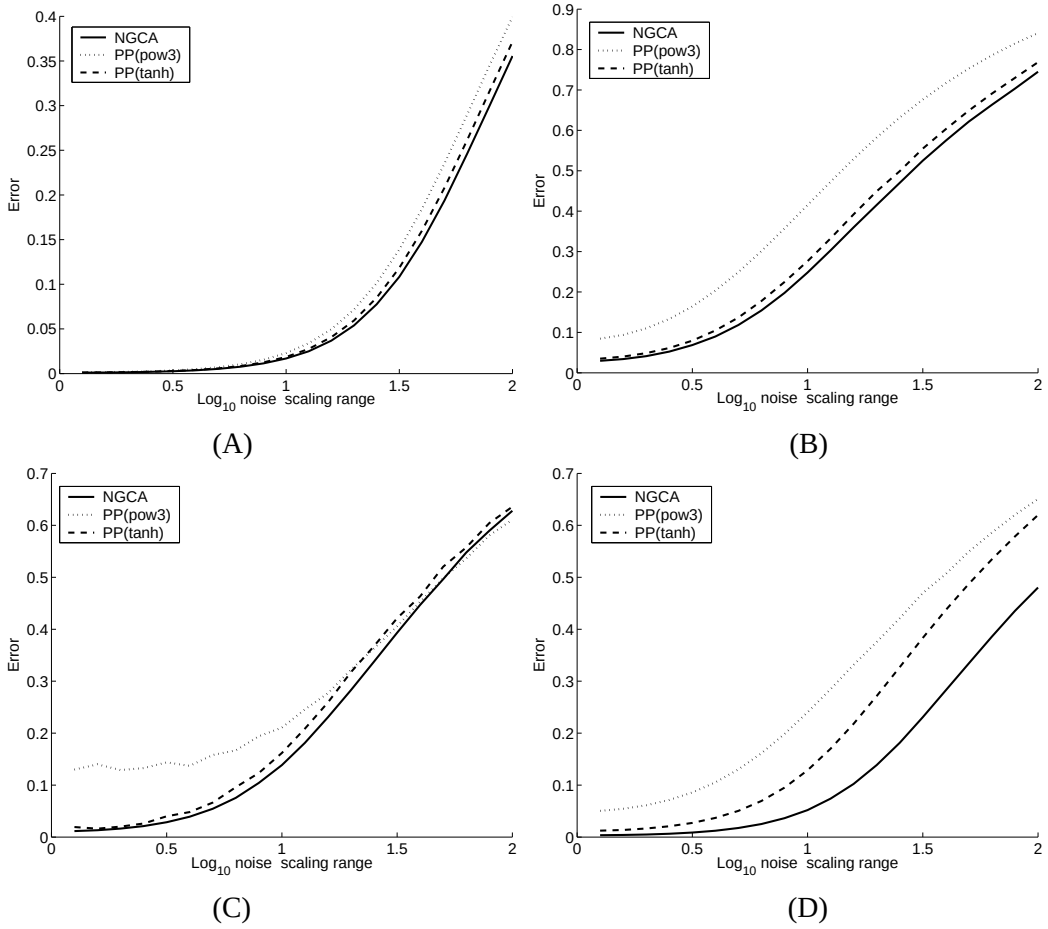


Figure 9: Results when the Gaussian (noise) components have different scales (the standard deviations follow a geometrical progression on $[10^{-r}, 10^{r}]$, where r is the parameter on the abscissa).

The results are depicted in Figure 9, where we observe again a transition to a failure mode when the covariance matrix is too badly conditioned. Although NGCA still appears as the best method, we observe that, on 3 out of 4 data sets, the transition to failure mode seems to happen roughly at the same point as for PP methods. This suggests that there is no or only little added robustness of NGCA with respect to PP in this regard. However, this result is not entirely surprising, as we expect this type of failure mode to be caused by a too large estimation error in the covariance matrix and therefore in the whitening/dewhitening steps. Since these steps are common to NGCA and the PP algorithms, it seems logical to expect a parallel evolution of their errors.

4.2 Example of Application for Realistic Data: Visualization and Clustering

We now give an example of application of our methodology to visualization and clustering of realistic data. We consider here "oil flow" data, which has been obtained by numerical simulation of a complex physical model. This data was already used before for testing techniques of dimension reduction (Bishop et al., 1998). The data is 12-dimensional and it is not known a priori if some dimensions are more relevant. Here our goal is to visualize the data and possibly exhibit a clustered structure. Furthermore, it is known that the data is divided into 3 classes. We show classes with different marker types but the class information is not used in finding the directions (i.e., the process is unsupervised).

We compare the NGCA methodology described in the previous section, projection pursuit ("vanilla" FastICA) using the tanh or the pow3 index, and Isomap (non-linear projection method, see Tenenbaum et al., 2000). The results are shown on Figure 10. A 3D projection of the data was computed using these methods, which was in turn projected in 2D to draw the figure; this last projection was chosen manually so as to make the cluster structure as visible as possible in each case.

We see that the NGCA methodology gives a much more relevant projection than PP using either tanh or pow3 alone: we can distinguish 10-11 clusters versus at most 5 for the PP methods and 7-8 for Isomap. Furthermore, the classes are clearly separated only on the NGCA projection; on the other ones, they are partially confounded in one single cluster. Finally, we confirm, by applying the projection found to held-out test data (i.e., data not used to determine the projection), that the cluster structure is relevant and not due to some overfitting artifact. This, in passing, shows one advantage of a linear projection method, namely that it can be extended to new data in a straightforward way.

Presumably, an important difference between the NGCA projection and the others comes from the Fourier functions, since they are not present in either of the PP methods. It can be confirmed by looking at the vector norms that Fourier functions are more relevant for this data set; they gave rise to estimated vectors with generally higher norms and had consequently a sizable influence of the choice of the projection. One could object that we have been merely lucky for this specific data because Fourier functions happened to be more relevant, and neither PP method uses this index. A possible suggestion for a fair comparison is to use the PP algorithm with a Fourier index. However, beside the fact that this index is not generally used in classical PP methods, the results would be highly dependent of the specific frequency parameter chosen, so we did not make experiments in that direction (by contrast, the NGCA methodology allows to combine vectors obtained from different frequencies). On the other hand, another route to investigate the relevance of this objection is to look at the results obtained by the NGCA method if Fourier functions are *not* used – thus only considering Gauss-pow3 and tanh. In this case, we still expect an improvement over PP because

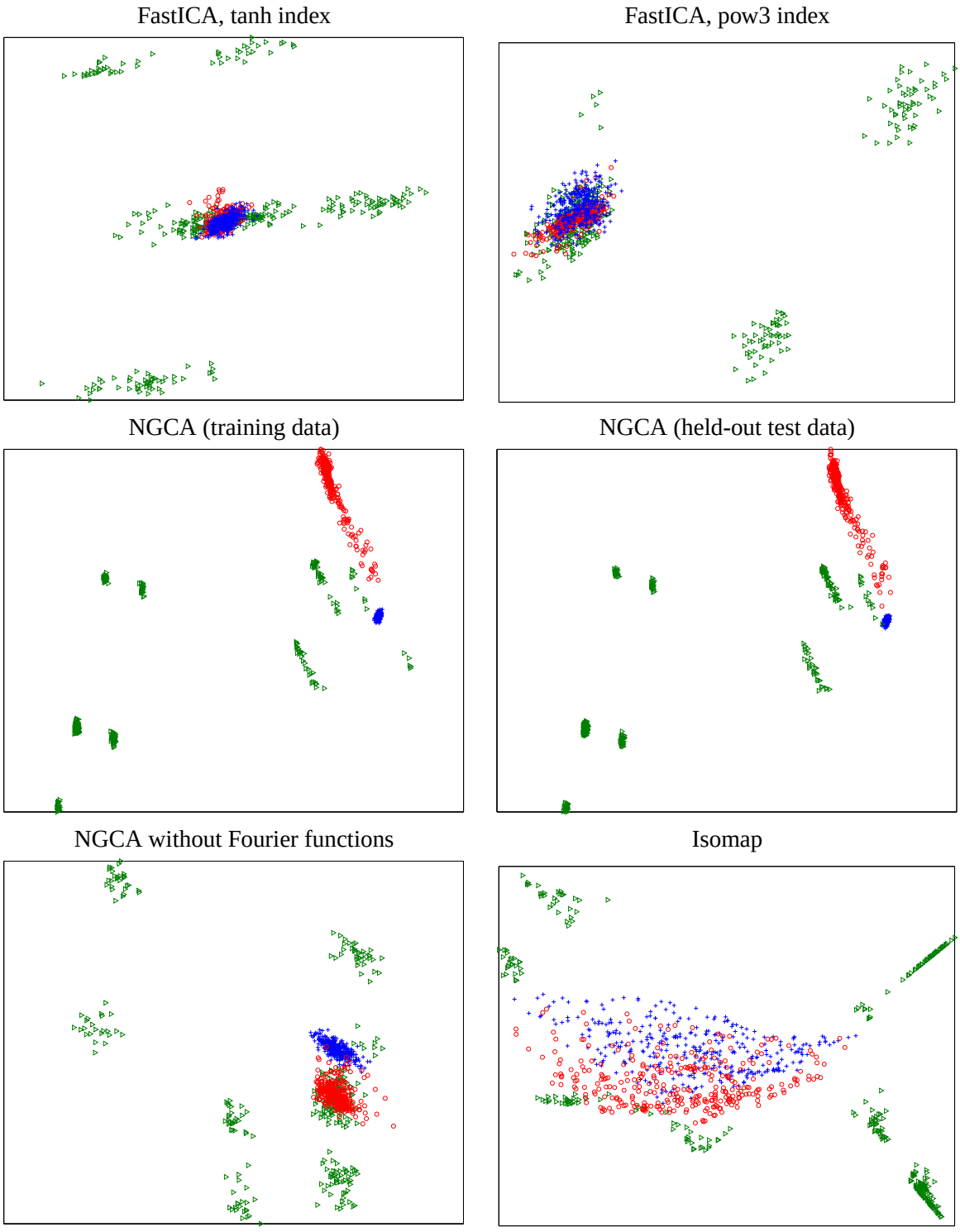


Figure 10: 2D projection of the "oil flow" data obtained by different algorithms. Different marker types/colors indicate the different classes (this information was not used to find the projections). For the middle right panel, the 2D projection found from the middle left panel was used to visualize additional held out test data.

NGCA is combining indices (as well as combining over the parameters ranges σ^2 and b). This is confirmed in Figure 10: even without the relevant Fourier functions, NGCA yields a projection where 8 clusters can be distinguished, and the classes are much more clearly separated than with PP methods. Finally, a visual comparison with the results obtained by Bishop et al. (1998) demonstrated that the projection found by our algorithm exhibits a clearer clustered structure; moreover, ours is a purely *linear* projection whereas the latter reference was a nonlinear data representation

Further analysis on clustering performance with additional data sets are given in the Appendix and underline the usefulness of our method.

5. Conclusion

We proposed a new semi-parametric framework for constructing a linear projection to separate an uninteresting multivariate Gaussian 'noise' subspace of possibly large amplitude from the 'signal-of-interest' subspace. Our theory provides generic consistency results on how well the non-Gaussian directions can be identified (Theorem 4). To estimate the non-Gaussian subspace from the set of vectors obtained, PCA is finally performed after suitable renormalization and elimination of uninformative vectors. The key ingredient of our NGCA method is to make use of the *gradient* computed for the nonlinear basis function $h(x)$ in Eq.(11) after data whitening. Once the low-dimensional 'signal' part is extracted, we can use it for a variety of applications such as data visualization, clustering, denoising or classification.

Numerically, we found comparable or superior performance to, e.g., FastICA in deflation mode as a generic representative of the family of PP algorithms. Note that, in general, PP methods need to pre-specify a projection index used to search for non-Gaussian components. By contrast, an important advantage of our method is that we are able to simultaneously use several families of nonlinear functions; moreover, inside a same function family, we are able to use an entire range of parameters (such as frequency for Fourier functions). Thus, our new method provides higher flexibility, and less restricting assumptions *a priori* on the data. In a sense, the functional indices that are the most relevant for the data at hand are automatically selected.

Future research will adapt the theory to simultaneously estimate the dimension of the non-Gaussian subspace. Extending the proposed framework to non-linear projection scenarios (Cox and Cox, 1994; Schölkopf et al., 1998; Roweis and Saul, 2000; Tenenbaum et al., 2000; Belkin and Niyogi, 2003; Harmeling et al., 2003) and to finding the most discriminative directions using labels are examples for which the current theory could be taken as a basis.

Acknowledgements: The authors would like to thank Stefan Harmeling for discussions and J.-F. Cardoso for suggesting us the pre-whitening step for increased efficiency and robustness. We would also like to thank anonymous reviewers for many insightful comments, in particular pointing out the ICA interpretation. We acknowledge partial financial support by DFG, BMBF (under Grant FKZ 01GQ0415) and the EU NOE PASCAL (EU # 506778). G.B. and M.S. also thank the Alexander von Humboldt foundation for partial financial support.

Appendix A. Proofs of the Theorems

A.1 Proof of Lemma 1

Suppose first that the noise N is standard normal. Denote by Π_E the projector from $\mathbb{R}^d$ to $\mathbb{R}^m$ which corresponds to the subspace E. Let also $E^\perp$ be the subspace complementary to E and $\Pi_{E^\perp}$ mean the projector on $E^\perp$. The standard normal noise can be decomposed as $N = N_1 \dot{+} N_2$ where $N_1 = \Pi_E N$ and $N_2 = \Pi_{E^\perp} N$ are independent noise components. Similarly, the signal X can be decomposed as

$$X = (\Pi_E S + N_1) \dot{+} N_2$$

where we have used the model assumption that the signal S is concentrated in E and it is independent of N. It is clear that the density of $\Pi_E S + N_1$ in $\mathbb{R}^m$ can be represented as the product $g(x_1)\phi(x_1)$ for some function g and the standard normal density $\phi(x_1)$, $x_1 \in \mathbb{R}^m$. The independence of N_1 and N_2 yields the in the similar way for $x = (x_1, x_2)$ with $x_1 = \Pi_E x$ and $x_2 = \Pi_{E^\perp} x$ that $p(x) = g(x_1)\phi(x_1)\phi(x_2) = g(x_1)\phi(x)$. Note that for the linear mapping $T = \Pi_E$ characterizes the signal subspace E. Namely, E is the image $\Im(T^*)$ of the dual operator T^* while $E^\perp$ is the null subspace (kernel) of T: $E^\perp = \Re(T)$.

Next we drop the assumption of the standard normal noise and assume only that the covariance matrix Γ of the noise is nondegenerated. Multiplying the both sides of the equation (1) by the matrix $\Gamma^{-1/2}$ leads to $\Gamma^{-1/2} X = \Gamma^{-1/2} S + \widetilde{N}$ where $\widetilde{N} = \Gamma^{-1/2} N$ is standard normal. The transformed signal $\widetilde{X} = \Gamma^{-1/2} S$ belongs to the subspace $\widetilde{E} = \Gamma^{-1/2} E$. Therefore, the density of $\widetilde{X}$ can be represented as $p(\widetilde{x}) = \widetilde{g}(\Pi_{\widetilde{E}} \widetilde{x})\phi(\widetilde{x})$ where $\Pi_{\widetilde{E}}$ is the projector corresponding to $\widetilde{E}$. Coming back the variable x yields the density of X in the form $p(x) = g(Tx)\phi(\Gamma^{-1/2} x)$ where $T = \Pi_{\widetilde{E}} \Gamma^{-1/2}$. ■

A.2 Proof of Proposition 2

For any function $\psi(x)$, it holds that

$$\int \psi(x+u) p(x) dx = \int \psi(x) p(x-u) dx,$$

if the integrals exists. Under mild regularity conditions on $p(x)$ and $\psi(x)$ allowing differentiation under the integral sign, differentiating this with respect to u gives

$$\int \nabla\psi(x) p(x) dx = -\int \psi(x) \nabla p(x) dx. \tag{15}$$

Let us take the following function

$$\psi_h(x) := h(x) - x^\top \mathbb{E}[X h(X)],$$

whose gradient is

$$\nabla\psi_h(x) = \nabla h(x) - \mathbb{E}[X h(X)].$$

The vector $\beta(h)$ is the expectation of $-\nabla\psi_h$. From Eq.(15) and using $\nabla p(x) = \nabla \log p(x)\, p(x)$, we have

$$\beta(h) = \int \psi_h(x) \nabla \log p(x)\, p(x) dx.$$

Applying Eq.(2) to the above equation yields

$$\begin{aligned}\beta(h) &= \int \psi_h(x)\nabla \log g(Tx)\; p(x)dx - \int \psi_h(x)\Gamma^{-1}x\, p(x)dx \\ &= T^* \int \psi_h(x)\nabla g(Tx)\phi_{\theta,\Gamma}(x)dx - \Gamma^{-1}\int x\psi_h(x)p(x)dx. \end{aligned} \tag{16}$$

Under the assumption $\mathbb{E}\left[XX^\top\right] = I_d$, we get

$$\mathbb{E}\left[X\psi_h(X)\right] = \mathbb{E}\left[Xh(X)\right] - \mathbb{E}\left[XX^\top\right]\mathbb{E}\left[Xh(X)\right] = 0,$$

that is, the second term of Eq.(16) vanishes. Since the first term of Eq.(16) belongs to I by the definition of I, we finally have $\beta(h) \in I$. ■

A.3 Proof of Theorem 3

For a fixed function h, we will essentially apply Lemma 5 stated below for each coordinate of $\beta_Y(h)$. Denoting the k-th coordinate of a vector v by $v^{(k)}$, and $y = \Sigma^{-\frac{1}{2}}x$, we have

$$\widetilde{h}^{(k)}(x) = \left|\left[\nabla h(y) - yh(y)\right]^{(k)}\right| \leq B(1+\|y\|) \leq B\left(1+K\|x\|\right).$$

It follows that $\widetilde{h}^{(k)}(x)$ is such that

$$\mathbb{E}\left[\exp\left(\frac{\lambda_0}{BK}\widetilde{h}^{(k)}(x)\right)\right] \leq a_0 \exp\left(\frac{\lambda_0}{K}\right),$$

and hence satisfies the assumption of Lemma 5. Denoting by $\widehat{\sigma}_k^2$ the sample variance of $\widetilde{h}^{(k)}$, we apply the lemma with $\delta' = \delta/d$, obtaining by the union bound that with probability at least $1-4\delta$, for all $1 \leq k \leq d$:

$$\left(\left[\beta_Y - \widehat{\beta}_Y\right]^{(k)}\right)^2 \leq 4\widehat{\sigma}_k^2 \frac{\log\left(d\delta^{-1}\right)}{n} + C_1(\lambda_0, a_0, B, d, K)\frac{\log^2(n\delta^{-1})\log^2\delta^{-1}}{n^{\frac{3}{2}}},$$

where we have used the inequality $(a+b)^2 \leq 2(a^2+b^2)$, and C_1 denotes some function depending only on the indicated quantities. Now summing over the coordinates, taking the square root and using $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$ leads to:

$$\left\|\beta_Y - \widehat{\beta}_Y\right\| \leq 2\sqrt{\widehat{\sigma}_Y^2(h)\frac{\log\delta^{-1}+\log d}{n}} + C_2(\lambda_0, a_0, B, d, K)\left(\frac{\log(n\delta^{-1})\log\delta^{-1}}{n^{\frac{3}{4}}}\right), \tag{17}$$

with probability at least $1-4\delta$. To turn this into a uniform bound over the family $\{h_k\}_{k=1}^L$, we simply apply this inequality separately to each function in the family with $\delta'' = \delta/L$. This leads to the first announced inequality of theorem. We obtain the second one by multiplying the first by $\Sigma^{-\frac{1}{2}}$ to the left and using the assumption on $\left\|\Sigma^{-1}\right\|$. ■

Lemma 5 *Let X be a real random variable such that for some $\lambda_0 > 0$:*

$$\mathbb{E}\left[\exp\left(\lambda_0 |X|\right)\right] \leq a_0 < \infty.$$

Let $X_1, \ldots, X_n$ denote an i.i.d. sequence of copies of X. Let $\mu = \mathbb{E}[X]$, $\widehat{\mu} = \frac{1}{n}\sum_{i=1}^n X_i$ and $\widehat{\sigma}^2 = \frac{1}{2n(n-1)}\sum_{i\neq j}(X_i - X_j)^2$ be the sample variance.

Then for any $\delta < \frac{1}{4}$ the following holds with probability at least $1-4\delta$, where c is a universal constant:

$$|\mu - \widehat{\mu}| \leq \sqrt{\frac{2\widehat{\sigma}^2 \log \delta^{-1}}{n}} + c\lambda_0^{-1} \max\left(1, \log\left(n a_0 \delta^{-1}\right)\right)\left(\left(\frac{\log \delta^{-1}}{n}\right)^{\frac{3}{4}} + \frac{\log \delta^{-1}}{n}\right).$$

Proof For $A > 0$ denote $X^A = X\mathbf{1}\{|X| \leq A\}$. We decompose

$$\left|\frac{1}{n}\sum_{i=1}^n X_i - \mu\right| \leq \left|\frac{1}{n}\sum_{i=1}^n \left(X_i - X_i^A\right)\right| + \left|\frac{1}{n}\sum_{i=1}^n X_i^A - \mathbb{E}\left[X^A\right]\right| + \left|\mathbb{E}\left[X - X^A\right]\right|;$$

these three terms will be denoted by T_1, T_2, T_3. By Markov's inequality, it holds that

$$\mathbb{P}\left[|X| > t\right] \leq a_0 \exp\left(-\lambda_0 t\right),$$

Fixing $A = \log\left(n\delta^{-1}a_0\right)/\lambda_0$ for the rest of the proof, it follows by taking $t = A$ in the above inequality that for any $1 \leq i \leq n$:

$$\mathbb{P}\left[X_i^A \neq X_i\right] \leq \frac{\delta}{n}.$$

By the union bound, we then have $X_i^A = X_i$ for all i, and therefore $T_1 = 0$, except for a set Ω_A of probability bounded by δ.

We now deal with the third term: we have

$$\begin{aligned} T_3 = \left|\mathbb{E}\left[X\mathbf{1}\{|X| > A\}\right]\right| \leq \mathbb{E}\left[X\mathbf{1}\{X > A\}\right] &= \int_0^\infty \mathbb{P}\left[X\mathbf{1}\{X > A\} > t\right] dt \\ &\leq A\mathbb{P}\left[X > A\right] + \int_A^\infty a_0 \exp\left(-\lambda_0 t\right) dt \\ &\leq a_0\left(A + \lambda_0^{-1}\right)\exp\left(-\lambda_0 A\right) \\ &= \frac{\delta}{n\lambda_0}\left(1 + \log\left(n\delta^{-1}a_0\right)\right). \end{aligned}$$

Finally, for the second term, since $\left|X^A\right| \leq A = \lambda_0^{-1}\log\left(n\delta^{-1}a_0\right)$, Bernstein's inequality ensures that with probability as least $1-2\delta$ the following holds:

$$\left|\frac{1}{n}\sum_{i=1}^n X_i^A - \mathbb{E}\left[X^A\right]\right| \leq \sqrt{\frac{2\mathrm{Var}\left[X^A\right]\log \delta^{-1}}{n}} + 2\frac{\log\left(n\delta^{-1}a_0\right)\log \delta^{-1}}{\lambda_0 n}.$$

We finally turn to the estimation of $\mathrm{Var}\left[X^A\right]$. The sample variance of X^A is given by

$$(\widehat{\sigma}^A)^2 = \frac{1}{2n(n-1)}\sum_{i\neq j}\left(X_i^A - X_j^A\right)^2.$$

Note that $(\widehat{\sigma}^A)^2$ is an unbiased estimator of $\mathrm{Var}\left[X^A\right]$. Furthermore, replacing X_i^A by $X_i'^A$ in the above expression changes this quantity at most of $4A^2/n$ since X_i^A appears only in $2(n-1)$ terms. Therefore, application of the bounded difference (a.k.a. McDiarmid's) inequality (McDiarmid , 1989) to the random variable $\widehat{\sigma}^A$ yields that with probability $1-\delta$ we have

$$\left|(\widehat{\sigma}^A)^2 - \mathrm{Var}\left[X^A\right]\right| \leq 4A^2\sqrt{\frac{\log \delta^{-1}}{n}};$$

finally, except for samples in the set Ω_A which we have already excluded above, we have $\widehat{\sigma}^A = \widehat{\sigma}$. Gathering these inequalities lead to the conclusion. ■

A.4 Proof of Theorem 4

In this proof we will denote by $C(\cdot)$ a factor depending only on the quantities inside the parentheses, and whose exact value can vary from line to line.

From Lemmas 9 and 10 below, we conclude that for big enough n, the following inequality is satisfied with probability $1-2/n$:

$$\left\|\Sigma^{-\frac{1}{2}} - \widehat{\Sigma}^{-\frac{1}{2}}\right\| \leq C(a_0,\lambda_0,K)\sqrt{\frac{d\log n}{n}}; \tag{18}$$

also, it is a a weaker consequence of Lemmas 7 and 8 that the following inequalities hold with probability at least $1-1/n$ each (again for n big enough):

$$\frac{1}{n}\sum_{i=1}^{n}\|X_i\| \leq C(a_0,\lambda_0), \tag{19}$$

$$\frac{1}{n}\sum_{i=1}^{n}\|X_i\|^2 \leq C(a_0,\lambda_0). \tag{20}$$

Let us denote Ω the set of samples where (18), (19) and (20) are satisfied simultaneously; from the above, we conclude that for large enough n, the set Ω contains the sample with probability at least $1-4/n$. For the remainder of the proof, we suppose that this condition is satisfied.

For any function h, we have

$$\left\|\widehat{\beta}_{\widehat{Y}} - \beta_Y\right\| \leq \left\|\widehat{\beta}_{\widehat{Y}} - \widehat{\beta}_Y\right\| + \left\|\widehat{\beta}_Y - \beta_Y\right\|.$$

Note that (up to some changes in the constants) the assumption on the Laplace transform is stronger than the assumption of Theorem 3; hence equation (17) in the proof of this theorem holds and we have with probability at least $1-4\delta$, for any function in the family $\{h_k\}_{k=1}^L$:

$$\left\|\beta_Y - \widehat{\beta}_Y\right\| \leq 2\sqrt{\widehat{\sigma}_Y^2(h)\frac{\log(L\delta^{-1})+\log d}{n}} + C(\lambda_0,a_0,B,d,K)\left(\frac{\log(nL\delta^{-1})\log\left(L\delta^{-1}\right)}{n^{\frac{3}{4}}}\right). \tag{21}$$

On the other hand, conditions (18) and (19) imply that for any function h in the family,

$$\begin{aligned}\left\|\widehat{\beta}_{\widehat{Y}}-\widehat{\beta}_Y\right\| = \left\|\frac{1}{n}\sum_{i=1}^n\left(\widetilde{h}(\widehat{Y}_i)-\widetilde{h}(Y_i)\right)\right\| &\leq \frac{M}{n}\sum_{i=1}^n\left\|\widehat{Y}_i-Y_i\right\| \\ &\leq \frac{M}{n}\left\|\Sigma^{-\frac{1}{2}}-\widehat{\Sigma}^{-\frac{1}{2}}\right\|\sum_{i=1}^n\|X_i\| \\ &\leq C(a_0,\lambda_0,K)M\sqrt{\frac{d\log n}{n}}\,.\end{aligned}$$

where in the first inequality, we have used the Lispchitz assumption on the function h.

One remaining technicality is to replace the term $\widehat{\sigma}_Y(h)$ (which cannot be evaluated from the data since it depends on the exactly whitened data Y_i) in (21) by $\widehat{\sigma}_{\widehat{Y}}(h)$, which can be evaluated from the data. For this use the following, holding for any function h in the family:

$$\left|\widehat{\sigma}^2_Y(h)-\widehat{\sigma}^2_{\widehat{Y}}(h)\right| = \frac{1}{2n(n-1)}\left|\sum_{i\neq j}\left\|\widetilde{h}(Y_i)-\widetilde{h}(Y_j)\right\|^2-\left\|\widetilde{h}(\widehat{Y}_i)-\widetilde{h}(\widehat{Y}_j)\right\|^2\right|;$$

let us now focus on one term of the above sum:

$$\begin{aligned}&\left\|\widetilde{h}(Y_i)-\widetilde{h}(Y_j)\right\|^2-\left\|\widetilde{h}(\widehat{Y}_i)-\widetilde{h}(\widehat{Y}_j)\right\|^2 \\ &\qquad= \left(\widetilde{h}(Y_i)-\widetilde{h}(\widehat{Y}_i)-\widetilde{h}(Y_j)+\widetilde{h}(\widehat{Y}_j)\right)^\top\left(\widetilde{h}(Y_i)-\widetilde{h}(Y_j)+\widetilde{h}(\widehat{Y}_i)-\widetilde{h}(\widehat{Y}_j)\right) \\ &\qquad\leq M^2\left(\left\|Y_i-\widehat{Y}_i\right\|+\left\|Y_j-\widehat{Y}_j\right\|\right)\left(\|Y_i-Y_j\|+\left\|\widehat{Y}_i-\widehat{Y}_j\right\|\right) \\ &\qquad\leq M^2\left\|\Sigma^{-\frac{1}{2}}-\widehat{\Sigma}^{-\frac{1}{2}}\right\|\left(\left\|\Sigma^{-\frac{1}{2}}\right\|+\left\|\widehat{\Sigma}^{-\frac{1}{2}}\right\|\right)\left(\|X_i\|+\|X_j\|\right)^2 \\ &\qquad\leq M^2C(a_0,\lambda_0,K)\sqrt{\frac{d\log n}{n}}\left(\|X_i\|^2+\|X_j\|^2\right),\end{aligned}$$

where we have used the Cauchy-Schwarz inequality, the triangular inequality and the Lipschitz assumption on $\widetilde{h}$ at the third line. Summing this expression over $i\neq j$, and using condition (20), we obtain

$$\left|\widehat{\sigma}^2_Y(h)-\widehat{\sigma}^2_{\widehat{Y}}(h)\right|\leq M^2C(a_0,\lambda_0,K)\sqrt{\frac{d\log n}{n}},$$

so that we can effectively replace $\widehat{\sigma}_Y$ by $\widehat{\sigma}_{\widehat{Y}}$ in (21) up to additional lower-order terms. This concludes the proof of the first inequality in the theorem.

For the second inequality, we additionally write

$$\begin{aligned}dist(\widehat{\gamma}(h),I) &\leq \left\|\widehat{\Sigma}^{-\frac{1}{2}}\widehat{\beta}_{\widehat{Y}}-\Sigma^{-\frac{1}{2}}\beta_Y\right\| \\ &\leq \left\|\Sigma^{-\frac{1}{2}}-\widehat{\Sigma}^{-\frac{1}{2}}\right\|\|\beta_Y\|+\left\|\Sigma^{-\frac{1}{2}}\right\|\left\|\widehat{\beta}_{\widehat{Y}}-\beta_Y\right\|+\left\|\Sigma^{-\frac{1}{2}}-\widehat{\Sigma}^{-\frac{1}{2}}\right\|\left\|\widehat{\beta}_{\widehat{Y}}-\beta_Y\right\|;\end{aligned}$$

we now conclude using (18), the previous inequalities controlling $\left\|\widehat{\beta}_{\widehat{Y}}-\beta_Y\right\|$, the assumption on $\left\|\Sigma^{-\frac{1}{2}}\right\|$ and the fact that

$$\|\beta_Y\| = \|\mathbb{E}[Xh(X)-\nabla h(X)]\|\leq B(1+\mathbb{E}[\|x\|])\leq C(a_0,\lambda_0,B)\,.$$

■

Appendix B. Additional Proofs and Results

We have used Bernstein's inequality, which we recall here for completeness under the following form:

Theorem 6 (Bernstein's inequality) *Suppose $X_1,\ldots,X_n$ are i.i.d. real random variables such that $|X| \leq b$ and $VarX = \sigma^2$. Then*

$$\mathbb{P}\left[\left|n^{-1}\sum_i X_i - E(X_i)\right| > \sqrt{2\sigma^2\frac{x}{n}} + 2b\frac{x}{n}\right] \leq 2\exp(-x).$$

The following results concern the estimation of $\Sigma^{-\frac{1}{2}}$, needed in the proof of Theorem 4. We divide this into 4 lemmas.

Lemma 7 *Let $\xi_1,\ldots,\xi_n$ be i.i.d. with $\mathbb{E}[\xi_1] = m$ and assume $\log\mathbb{E}[\exp\mu(\xi_1 - m)] \leq c\mu^2/2$ holds for all $\mu \leq \mu_0$, for some positive constants c and μ_0. Then for sufficiently large n*

$$\mathbb{P}\left[n^{-1/2}\sum_{i=1}^n(\xi_i - m) > z\right] \leq e^{-c^{-1}z^2/2}.$$

Proof This is an application of Chernoff's bounding method:

$$\begin{aligned} R_n &:= \log\mathbb{P}\left[n^{-1/2}\sum_{i=1}^n(\xi_i - m) > z\right] \\ &\leq -\mu z\sqrt{n} + \log\mathbb{E}\left[\exp\sum_{i=1}^n \mu(\xi_i - m)\right] \\ &= -\mu z\sqrt{n} + n\log\mathbb{E}\left[\exp\mu(\xi_1 - m)\right], \end{aligned}$$

where the above inequality is Markov's. We select $\mu = zn^{-1/2}c^{-1}$. For n sufficiently large, it holds that $\mu \leq \mu_0$ and by the lemma condition

$$R_n \leq -\mu z\sqrt{n} + nc\mu^2/2 = -z^2c^{-1}/2.$$

■

The goal of the following Lemma is merely to replace the assumption about the Laplace transform (in the previous Lemma) by a simpler assumption (existence of some exponential moment). This allows a simpler statement – as far as we are not really interested in the precise constants involved.

Lemma 8 *Let X be a real random variable such that for some $\mu_0 > 0$:*

$$\mathbb{E}\left[\exp(\mu_0 |X|)\right] = e_0 < \infty.$$

Then there exists $c > 0$ (depending only on μ_0 and e_0) such that

$$\forall \mu \in \mathbb{R} \qquad |\mu| \leq \mu_0/2 \Rightarrow \log \mathbb{E}\left[\exp\left(\mu(X - \mathbb{E}[X])\right)\right] \leq c\mu^2/2.$$

Proof Note that X has finite expectation since $|X| \leq \mu_0^{-1} \exp \mu_0 |X|$. Taylor's expansion gives that

$$\forall x \in \mathbb{R},\ \forall \mu \in \mathbb{R},\ |\mu| < \mu_0/2 \Rightarrow \exp(\mu x) \leq 1 + \mu x + \frac{\mu^2}{2} x^2 \exp(|\mu_0|\,|x|/2). \tag{22}$$

There exists some constant $c > 0$ (depending on μ_0) such that

$$\forall x \in \mathbb{R},\ x^2 \exp(|\mu_0 x|/2) \leq c\left(\exp(|\mu_0 x|)\right).$$

Using this and the assumption, taking expectation in (22) yields that for $c' = \frac{1}{2} c e_0 > 0$

$$\forall \mu \in \mathbb{R},\ |\mu| < \mu_0/2 \Rightarrow \mathbb{E}\left[\exp(\mu X)\right] \leq 1 + \mu \mathbb{E}[X] + c'\mu^2 \leq \exp\left(\mu \mathbb{E}[X] + c'\mu^2\right),$$

implying

$$\mathbb{E}\left[\exp\left(\mu(X - \mathbb{E}[X])\right)\right] \leq \exp\left(c'\mu^2\right);$$

taking logarithms on both sides yields the conclusion. ■

The next two Lemmas, once combined, provide the confidence bound on $\left\|\Sigma^{-\frac{1}{2}} - \widehat{\Sigma}^{-\frac{1}{2}}\right\|$ which we need for the proof of Theorem 4.

Lemma 9 *Let $X_1, \ldots, X_n$ be i.i.d. vectors in $\mathbb{R}^d$. Assume that, for some $\mu_0 > 0$,*

$$\mathbb{E}\left[\exp\left(\mu_0 \|X\|^2\right)\right] = e_0 < \infty; \tag{23}$$

denote $\Sigma = \mathbb{E}\left[XX^\top\right]$ and $\widehat{\Sigma}$ it empirical counterpart. Then for some constant κ depending only on (μ_0, e_0), and for big enough n,

$$R_n^* := \mathbb{P}\left[\left\|\Sigma - \widehat{\Sigma}\right\| > \sqrt{\frac{\kappa d \log n}{n}}\right] \leq \frac{2}{n}.$$

Proof Along this proof C, c will denote constants depending only on μ_0, e_0; their exact value can change from line to line. Note that by definition of Σ and $\widehat{\Sigma}$,

$$\left\|\Sigma - \widehat{\Sigma}\right\| = \sup_{\theta \in \mathcal{B}_d} \frac{1}{n} \sum_{i=1}^n \left((X_i^\top \theta)^2 - \mathbb{E}\left[\left(X^\top \theta\right)^2\right]\right),$$

where $\mathcal{B}_d$ denotes the unit ball of $\mathbb{R}^d$. For $\varepsilon < 1$, let $\mathcal{B}_{d,\varepsilon}$ denote a ε-packing set of $\mathcal{B}_d$, that is, a discrete ε-separated set of points of $\mathcal{B}_d$ of maximum cardinality. By the maximality assumption and the triangle inequality, $\mathcal{B}_{d,\varepsilon}$ is also a 2ε-covering net of $\mathcal{B}_d$. On the other hand, the ε-balls

centered on these points are disjoint, and their union is included in the ball of radius $(1+\varepsilon)$, so that a volume comparison allows us to conclude that $\#(B_{d,\varepsilon})\varepsilon^d \leq (1+\varepsilon)^d \leq 2^d$. This shows that $\mathcal{B}_{d,2\varepsilon}$ is a 4ε-covering set of $\mathcal{B}_d$ of cardinality bounded by ε^{-d}.

Now, if $\theta, \theta' \in \mathcal{B}_d$ are such that $\|\theta - \theta'\| \leq 4\varepsilon$, then we have

$$\begin{aligned}\left|\sum_{i=1}^n (X_i^\top \theta)^2 - \sum_{i=1}^n (X_i^\top \theta')^2\right| &= \left|\sum_{i=1}^n (X_i^\top(\theta-\theta'))(X_i^\top(\theta+\theta'))\right| \\ &\leq 8\varepsilon \sum_{i=1}^n \|X_i\|^2,\end{aligned}$$

where we have applied the Cauchy-Schwarz inequality at the last line.

Now application of Lemmas 7 and 8 yields that for n large enough, with probability at least $1-1/n$,

$$n^{-1}\sum_{i=1}^n \|X_i\|^2 \leq \mathbb{E}\left[\|X\|^2\right] + \sqrt{\frac{c\log n}{n}} \leq C.$$

The above implies that with probability at least $1-1/n$,

$$\sup_{\theta,\theta' \in \mathcal{B}_d : \|\theta-\theta'\| \leq 2\varepsilon} n^{-1/2}\left|\sum_{i=1}^n (X_i^\top \theta)^2 - \sum_{i=1}^n (X_i^\top \theta')^2\right| \leq C\varepsilon\sqrt{n}.$$

We can also show a similar inequality about the corresponding expectation

$$\sup_{\theta,\theta' \in \mathcal{B}_d : \|\theta-\theta'\| \leq 2\varepsilon} n^{-1/2}\left|\mathbb{E}\left[(X^\top \theta)^2\right] - \mathbb{E}\left[(X^\top \theta')^2\right]\right| \leq C\varepsilon\sqrt{n}.$$

We now select $\varepsilon = n^{-\frac{1}{2}}$. Therefore, approximating any $\theta \in \mathcal{B}_d$ by its nearest neighbour in $\mathcal{B}_{d,2\varepsilon}$ and using the above inequalitites, we obtain that

$$\begin{aligned} R_n^* &\leq \frac{1}{n} + \mathbb{P}\left[\sup_{\theta \in \mathcal{B}_{d,2\varepsilon}} n^{-1/2}\sum_{i=1}^n \left((X_i^\top \theta)^2 - \mathbb{E}\left[\left(X^\top \theta\right)^2\right]\right) > \sqrt{\kappa d \log n} - C\right] \\ &\leq \frac{1}{n} + \sum_{\theta \in B_{d,2\varepsilon}} \mathbb{P}\left[n^{-1/2}\sum_{i=1}^n \left((X_i^\top \theta)^2 - \mathbb{E}\left[\left(X^\top \theta\right)^2\right]\right) > \sqrt{(\kappa - C) d \log n}\right] \\ &\leq \frac{1}{n} + \#(B_{d,2\varepsilon}) \exp\{-0.5c^{-1}(\kappa - C) d \log n\} \leq \frac{2}{n} \end{aligned}$$

provided that κ is chosen so that $c^{-1}(\kappa - C)d/2 > d/2 + 1$. Here we have again used Lemmas 7 and 8, noting that for any $\theta \in \mathcal{B}_d$ it holds that $\mathbb{E}\left[\exp \mu_0 \left|\theta^\top X\right|\right] \leq \mathbb{E}\left[\exp \mu_0 \|X\|\right] < \exp(\mu_0) + e_0$ by assumption. ■

Lemma 10 *Let A, B be two real positive definite symmetric matrices satisfying $\|A - B\| \leq \varepsilon$ with $\varepsilon \leq (2\left\|A^{-1}\right\|)^{-1}$. Then there exists a constant C such that*

$$\left\|A^{-\frac{1}{2}} - B^{-\frac{1}{2}}\right\| \leq C\left\|A^{-1}\right\|^{\frac{3}{2}}\varepsilon.$$

Proof

Note that for $\|M\| < 1$, it holds that

$$(I-M)^{-\frac{1}{2}} = \sum_{k\geq 0} \gamma_k M^k,$$

with $(\gamma_k) \geq 0$ the coefficients of the power series development of the function $1/\sqrt{1-x}$.

Denote $\lambda_{max}(M), \lambda_{min}(M)$ the biggest and smallest eigenvalue of a matrix M. Put $K = \|A\| = \lambda_{max}(A)$ and $L = \left\|A^{-1}\right\| = \lambda_{min}(A)^{-1}$. Note that $LK \geq 1$. Put $A' = A/K, B' = B/K$. All eigenvalues of A' belong to $(0,1]$ and therefore

$$\left\|I - A'\right\| = \lambda_{\max}(I - A') = 1 - \lambda_{min}(A') = 1 - (LK)^{-1}.$$

By the assumption that $\varepsilon \leq (2L)^{-1}$, it holds that

$$\lambda_{max}(B') = K^{-1}\|B\| \leq K^{-1}(\|A\| + \varepsilon) \leq 1 + (2LK)^{-1} \leq \frac{3}{2},$$

and that

$$\lambda_{min}(B') \geq K^{-1}(\lambda_{min}(A) - \varepsilon) \geq (2KL)^{-1},$$

from this we deduce that

$$\left\|I - B'\right\| = \max(\lambda_{max}(B') - 1, 1 - \lambda_{min}(B')) \leq \max\left(\frac{1}{2}, 1 - (2LK)^{-1}\right) = 1 - (2LK)^{-1}.$$

Putting $\overline{A} = I - A', \overline{B} = I - B'$, we have ensured that $\left\|\overline{A}\right\| < 1$ and $\left\|\overline{B}\right\| < 1$; we can thus write

$$\begin{aligned} A'^{-\frac{1}{2}} - B'^{-\frac{1}{2}} &= \left(I - \overline{A}\right)^{-\frac{1}{2}} - \left(I - \overline{B}\right)^{-\frac{1}{2}} \\ &= \sum_{k\geq 1} \gamma_k (\overline{A}^k - \overline{B}^k). \end{aligned}$$

Noticing that

$$\left\|\overline{A}^k - \overline{B}^k\right\| = \left\|\sum_{i=0}^{k-1} \overline{A}^i (\overline{A} - \overline{B}) \overline{B}^{k-1-i}\right\| \leq k \max\left(\left\|\overline{A}\right\|, \left\|\overline{B}\right\|\right)^{k-1} \left\|A' - B'\right\|,$$

we obtain

$$\begin{aligned} \left\|A'^{-\frac{1}{2}} - B'^{-\frac{1}{2}}\right\| &\leq \left\|A' - B'\right\| \sum_{k\geq 1} k\gamma_k \left(1 - (2LK)^{-1}\right)^{k-1} \\ &= \frac{\varepsilon}{K} \frac{1}{2} (2LK)^{\frac{3}{2}} = CL^{\frac{3}{2}} K^{\frac{1}{2}} \varepsilon. \end{aligned}$$

From this we deduce that

$$\left\|A^{-\frac{1}{2}} - B^{-\frac{1}{2}}\right\| = K^{-\frac{1}{2}} \left\|A'^{-\frac{1}{2}} - B'^{-\frac{1}{2}}\right\| \leq CL^{\frac{3}{2}} \varepsilon.$$

■

Appendix C. Clustering Results

The goal of NGCA is to discover interesting structure in the data. It is naturally a difficult task to quantify this property precisely. In this appendix we try to make this apparent using clustering techniques. We apply a mean distance linkage clustering algorithm to data projected in lower dimension using various techniques: NGCA, FastICA, PCA, local linear embedding (LLE, Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000).

There is no single well-defined performance measure for the performance of clustering. Here we resort to indirect criteria that should however allow a comparative study. We consider the two following criteria:

(1) Label cross-information. We apply clustering to benchmark data for which label information Y is available. Although this information is not used in determining the clustering, we will use it as a yardstick to measure whether the clustering gives rise to relevant structure discovery. We measure this by the scaled mutual information $I(C,Y)/H(Y)$, where C is the cluster labelling and the normalization ensures that the quantity lies between 0 and 1. Note that there is *a priori* no mathematical reason why clustering should be related to label information, but this is often the case for real data, so this can be a relevant criterion of structure discovery. A higher score indicates a better match between discovered cluster structure and label structure.

(2) Stability. Recent attempts at formalizing criteria for clustering have proposed that clustering stability should be a relevant criterion for data clustering (see, e.g., Meinecke et al. , 2002; Lange et al., 2004). Again, this is only an indirect criterion, as, for example, a trivial clustering algorithm dividing the space without actually looking at the data would be very stable. But with this caveat in mind, it provides a relevant diagnostic tool. Here, we measured stability in the following way: the data is divided randomly into 2 groups of equal size on which we apply clustering. Then, the cluster labels obtained on group 1 are extended to group 2 by the nearest-neighbor rule and vice-versa. This thus gives rise to two different cluster labellings C_1, C_2 of the whole data and we measure their agreement through relative mutual information $I(C_1,C_2)/H(C_1,C_2)$. Again, this score lies in the interval $[0,1]$ and a high score indicates better stability.

Table 1: Description of data sets

Data set	Nb. of Classes	Nb. of samples	Total dimension	Projection Dim.
Oil	3	2000	12	3
Wine	3	178	13	3
Vowel	11	528	10	3
USPS	10	7291	30	10

We consider the "oil flow" data already presented in section 4.2, and additional data sets from the UCI classification repository, for which the features all take continuous values. (When there are features taking only discrete values, NGCA is inappropriate since these will generally be picked up as strongly non-Gaussian). Size and dimension of these data sets are given in Table 1.

The results are depicted in Figure 11. On the Oil data set, NGCA works very well for both criteria (as was expected from the good visualization results of section 4.2). On the Wine data set, the different algorithms appear to be divided in two clear groups, with the performance in the first group (NGCA, Isomap, LLE) noticeably better than in the second (PCA, FastICA). NGCA belongs to the

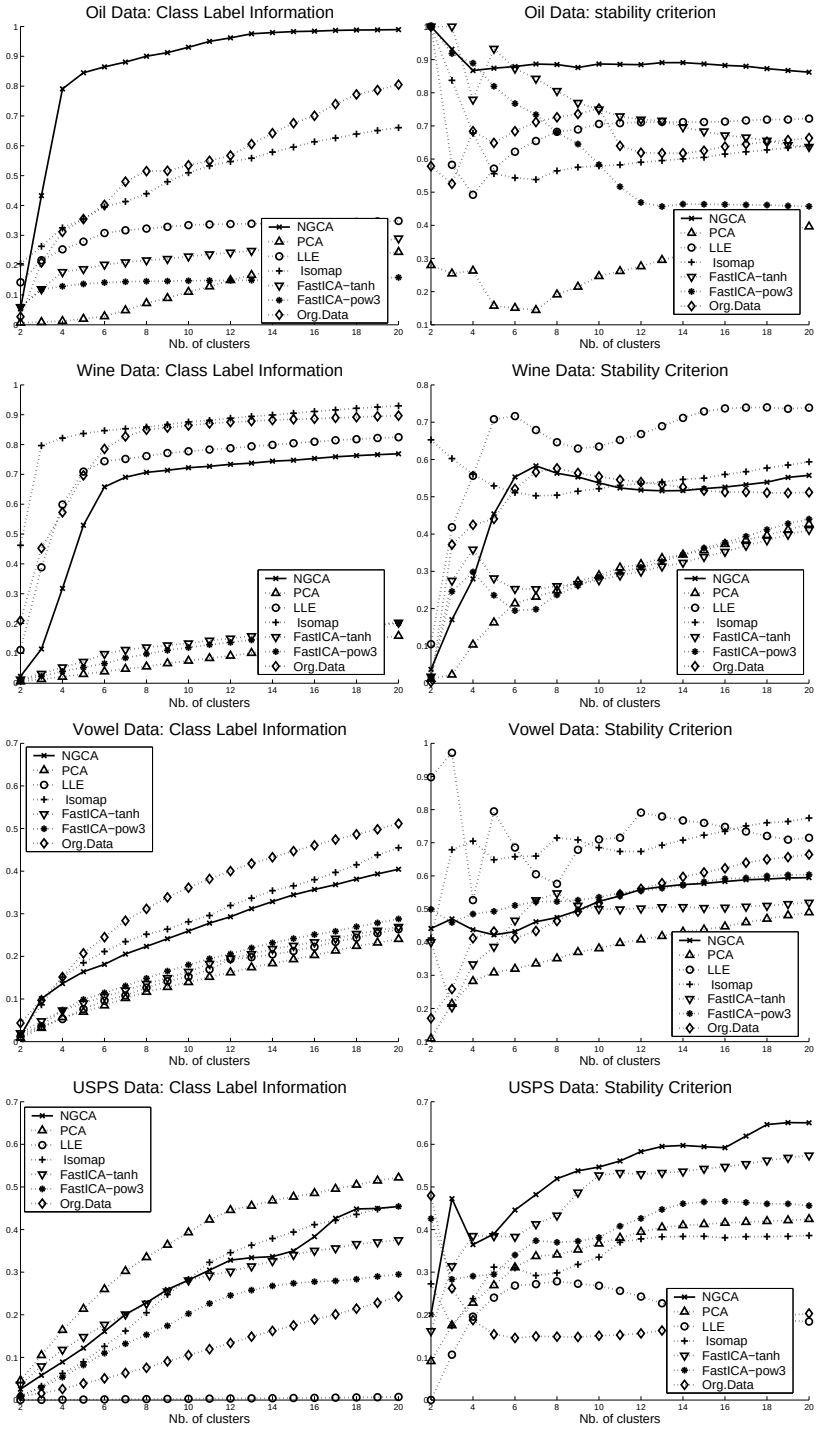


Figure 11: Clustering results

better group although the best methods appear to be the non-linear projections LLE and Isomap. The results of the Vowel data set are probably the most difficult to interpret, as most methods appear to be relatively close. Isomap appears as the winner method in this case, with NGCA quite close in terms of label cross-information and in the middle range for stability. Finally, for the USPS data set we used the 30 first principal components obtained by Kernel-PCA and a polynomial kernel of degree 3. In this case, PCA gives better results in terms of label cross-information with NGCA a close second, while NGCA is the clear winner in terms of stability.

To summarize: NGCA performed very well in 2 of the 4 data sets tried (Oil data and USPS), and was in the best group of methods for the Wine Data and had average performance on the last data set. Even when NGCA is outperformed by nonlinear methods LLE and Isomap, it generally achieves a comparable performance though being a linear method, which has other advantages such as clearer geometrical interpretation, direct extension to additional data if needed, and possible assessment of variable importance in original space.

References

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

C. M. Bishop, M. Svensen and C. K. I. Wiliams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

C. M. Bishop and G. D. James. Analysis of multiphase flow using dual-energy gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research*, A327:580–593, 1993.

P. Comon. Independent component analysis—a new concept? *Signal Processing*, 36:287–314, 1994.

T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 2001.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, Springer, 1996.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1975.

S. Harmeling, A. Ziehe, M. Kawanabe and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15(5):1089–1124, 2003.

P. J. Huber. Projection pursuit. *The Annals of Statistics*, 13:435–475, 1985.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

A. Hyvärinen, J. Karhunen and E. Oja. *Independent Component Analysis*. Wiley, 2001.

M. C. Jones and R. Sibson. What is projection pursuit? *Journal of the Royal Statistical Society, series A*, 150:1–36, 1987.

C. McDiarmid. On the method of bounded differences, *Surveys in Combinatorics*, London Math. Soc. Lecture Notes Series 141:148–188, 1989.

F. Meinecke, A. Ziehe, M. Kawanabe, and K.-R. Müller. A resampling approach to estimate the stability of one-dimensional or multidimensional independent components. *IEEE Transactions on Biomedical Engineering*, 49:1514–1525, 2002.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500): 2323–2326, 2000.

T. Lange, V. Roth, M. L. Braun and J. M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299-1323, 2004.

B. Schölkopf, A. J. Smola and K.–R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

J. B. Tenenbaum, V. de Silva and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Journal of Machine Learning Research 7 (2006) 283–306 Submitted 9/05; Published 2/06

Some Discriminant-Based PAC Algorithms

Paul W. Goldberg PWG@DCS.WARWICK.AC.UK
Department of Computer Science
University of Warwick
Coventry, CV4 7AL, UK

Editor: Dana Ron

Abstract

A classical approach in multi-class pattern classification is the following. Estimate the probability distributions that generated the observations for each label class, and then label new instances by applying the Bayes classifier to the estimated distributions. That approach provides more useful information than just a class label; it also provides estimates of the conditional distribution of class labels, in situations where there is class overlap.

We would like to know whether it is harder to build accurate classifiers via this approach, than by techniques that may process all data with distinct labels together. In this paper we make that question precise by considering it in the context of PAC learnability. We propose two restrictions on the PAC learning framework that are intended to correspond with the above approach, and consider their relationship with standard PAC learning. Our main restriction of interest leads to some interesting algorithms that show that the restriction is not stronger (more restrictive) than various other well-known restrictions on PAC learning. An alternative slightly milder restriction turns out to be almost equivalent to unrestricted PAC learning.

Keywords: computational learning theory, computational complexity, pattern classification

1. Introduction

We present some PAC learning algorithms for various learning problems, within a new restriction of the PAC setting. We begin by explaining the motivation for studying the new restriction, and continue with some general results about it, followed by the algorithms.

1.1 Background and Motivation

A standard approach to classification problems (see for example (Duda and Hart, 1973), page 17) is the following. For each class, find a *discriminant function* that maps elements of the input domain to real values. These functions can be used to label any element x of the domain with the class label whose associated discriminant function takes the largest value on x. The discriminant functions are usually estimates of the probability densities of points in each class, weighted by the class prior (relative frequency of that class), in which case we are using the *Bayes classifier*.

If it is possible to obtain good estimates of the probability distributions that generated the label classes, then (for reasons we explain below) these are often more useful than just an accurate classification rule. However, this raises the question of how much harder it becomes to learn to classify data well, if we actually insist on learning the distributions. This motivates our choice to study this general question in the context of PAC learning, since PAC learning gives a framework for results

giving lower bounds on sample-size or computational requirements. These results allow different models of the learning process to be provably distinguished from each other, in terms of the learning problems that are tractable within each model.

Most recent work on pattern classification (for example work on support vectors (Cristianini and Shawe-Taylor, 2000)) of course does not try to learn label class distributions, but rather to find decision boundaries that optimize some performance guarantee, usually misclassification rate. Performance guarantees are derived from observed classification performance in conjunction with other features of the boundary such as syntactic or combinatorial complexity, or the number of support vectors and margin of separation. The general approach clearly requires examples with different labels to be taken in conjunction with each other when finding a decision boundary. By contrast, discriminant functions are constructed from individual label classes in isolation. It seems clearly "easier" to find a good classifier by considering all data together (so as to apply empirical risk minimisation), than by insisting that each label class must be independently converted into a discriminant function. Noting Vapnik's observation ((Vapnik, 2000), page 30) that one should not try to solve a problem via solving a more general problem, why exactly would we want to estimate the distributions of label classes?

The answer is that when the distributions can be found, the extra information that is obtained is often very useful in practice. In contrast with decision boundaries, we obtain for a domain element x, the values of the probability densities of label classes at x, which provide a conditional distribution over the class label of x. A predicted class label for x can then take into account variable misclassification penalties, or changes in the assumed class priors. There are of course other ways to obtain such distributions, for example using logistic regression, or more generally (for k-class classification) neural networks with k real-valued outputs re-scaled using the softmax activation function (see (Bishop, 1995) for details). However, unsupervised learning for each class—if it can be done successfully—has other advantages over these techniques, such as the following.

1. Label classes can be added without re-training the system. So for example if a new symbol were added to a character set, then given a good estimate of the probability distribution over images of the new symbol, this can be used in conjunction with pre-existing models for how the other symbols are generated.

2. *Outlying* instances are those that lie in regions of the domain where the distributions have low weight. We usually cannot assign a sensible label to such instances, however they may at least be recognised as a result of all class label distributions having very small weight around such an instance.

3. For applications such as handwritten digit recognition, it is arguably more natural—from the perspective of cognitive modeling—to model the data generation process in terms of 10 separate probability distributions, than as a collection of thresholds between different digits. This is because a handwritten zero (say) is nearly always the result of a process that first chooses the label "0" and then creates the image. It is not the result of a process that first generates a character and then assigns it the label "0" based on context, appearance or other criteria.

Another difficulty with decision boundaries arises specifically in the context of multiclass classification. It has been noted (Allwein et al., 2000) that multiclass classifiers are often constructed

using multiple 2-class classifiers. How to combine them is a challenging topic that has itself received much recent attention, see for example (Guruswami and Sahai, 1999; Allwein et al., 2000). In practical studies such as (Platt et al., 2000) that build a multi-class classifier from a collection of 2-class classifiers, a distinction is made between separating each class from the union of the others (*1-v-r* classifiers, where 1-v-r stands for one-versus-rest) and pairwise separation (*1-v-1* classifiers). Neither is entirely satisfactory—for example it may be possible to perform linear 1-v-1 separation for all pairs of classes, but not linear 1-v-r separation, while 1-v-1 classification (as studied in (Platt et al., 2000)) raises the problem of combining the collection of pairwise classifiers in a principled way to get an overall classification, for example ensuring that all classes are treated the same way. In (Platt et al., 2000), the first test for any unlabeled input is to apply the separator that distinguishes 0 from 9. Thus 0 and 9 are being treated differently from other digits (which in turn are also treated differently from each other.)

With regard to PAC learning, the approach of applying unsupervised learning to each label class, can treat situations where class overlap occurs (as is usually the case in practice). Standard PAC algorithms do not address this problem (although there have been extensions such as "probabilistic concepts" (Kearns and Schapire, 1994) that do so, and methods using support vectors that also allow decision boundaries that do not necessarily agree with all observed data). It is not hard to verify (see (Palmer and Goldberg, 2005)) that when we have good estimates of the class label distributions (in a sense described below in Section 1.3) then the associated classifier is approximately optimal in the agnostic PAC sense. For large data set sizes, it becomes feasible to find good estimates of these distributions, and obtain this more useful "summary" of the data.

The algorithms described in this paper are given in the context of simple 2-class classification, as opposed to multi-class classification. This is because we aim to explore the problems arising from an insistence upon treating each label class independently. The algorithms would however apply in a multi-class context where each pair of classes is separated by a boundary belonging to the given set of boundaries.

1.2 Formal Definition of the Learning Framework

In PAC learning (see for example (Anthony and Biggs, 1992) for a detailed introduction) there is a source of data consisting of instances generated by a probability distribution D over a domain X, labeled using an unknown "target function" $t : X \longrightarrow \{0,1\}$. The objective is to find a classifier $h : X \longrightarrow \{0,1\}$ which is a good approximation to t with respect to probability measure D. As usual we will let ε denote a misclassification rate (probability that t and h disagree on random x) and δ denote the uncertainty (probability that error rate ε is not attained). We refer to the members of $t^{-1}(1)$ as the "positive examples" and the members of $t^{-1}(0)$ as the "negative examples".

We say that a set $\mathcal{C}$ of functions from X to $\{0,1\}$ (the "concept class") is PAC learnable if there exists an algorithm $\mathcal{A}$ that for any $t \in \mathcal{C}$, with probability at least $1-\delta$ outputs $h : X \longrightarrow \{0,1\}$ having misclassification rate at most ε. $\mathcal{A}$ is required to run in time polynomial in ε^{-1} and δ^{-1} and other parameters (usually, the syntactic description length of t and members of the training data). $\mathcal{A}$ may sample x from D in unit time, and obtain $(x, t(x))$. In this standard definition, we assume that t and D may be worst-case (chosen by an adversary).

Notation. For $\ell \in \{0,1\}$ let D_ℓ denote the restriction of D to $t^{-1}(\ell)$. Let p_ℓ denote the class prior for label ℓ, $p_\ell = \Pr_{x\sim D}(t(x) = \ell)$. We assume throughout that $p_\ell > 0$. Thus

$$\begin{array}{ll} \Pr_{x\sim D_\ell}(x) = \frac{1}{p_\ell}\Pr_{x\sim D}(x) & \text{for } t(x) = \ell \\ \Pr_{x\sim D_\ell}(x) = 0 & \text{for } t(x) = 1-\ell \end{array}$$

For any probability distribution P over X (for example D or D_ℓ), an algorithm *with access to P* may in unit time draw an unlabeled sample from P.

It is shown in (Haussler et al., 1991) that the standard PAC framework is equivalent to a "two-button" version, where an algorithm has access to a "positive example oracle" and a "negative example oracle". (The two-button version conceals the class priors and only gives the algorithm access to the distribution as restricted to each class label. Thus the oracles generate examples from D_1 and D_0 respectively.) We define a restriction of "two-button" learning as follows.

Definition 1 *Suppose algorithm $\mathcal{A}$ has access to a distribution P over X, and the output of $\mathcal{A}$ is a function $f : X \longrightarrow \mathbb{R}$. Execute $\mathcal{A}$ twice, using D_1 (respectively D_0) for P. Let f_1 and f_0 be the functions obtained respectively. For $x \in X$ let*

$$\begin{array}{rll} h(x) = 1 & \text{if} & f_1(x) > f_0(x) \\ h(x) = 0 & \text{if} & f_1(x) < f_0(x) \\ h(x) \text{ undefined} & \text{if} & f_1(x) = f_0(x) \end{array}$$

If $\mathcal{A}$ takes time polynomial in ε^{-1} and δ^{-1}, and h is PAC with respect to ε and δ, then we will say that $\mathcal{A}$ PAC-learns via discriminant functions.

It is clear that if $\mathcal{A}$ can be found such that the resulting h is PAC, then we have PAC learnability in the two-button setting, and hence standard PAC learnability.

In specifying the restriction above, we are keeping it both simple and "severe", in the sense of making it difficult to find algorithms within the restricted framework. This is with a view to getting strong positive results, and also to maximizing the potential for negative results (PAC learnable problems that are hard within the restricted setting). One could devise less severe restrictions to capture the general idea of learning via discriminant functions. Alternatives are discussed at the end of this section.

We also consider the following slightly less severe variant related to POSEX learnability as introduced in (Denis, 1998), in which $\mathcal{A}$ has access to D (in (Denis, 1998) the "EX" oracle), in addition to D_1 (in (Denis, 1998) the "POS" oracle). This is formalized as Definition 2, and it turns out that we can be much more specific about learnability in this sense, namely it is intermediate between PAC learnability with uniform misclassification noise and basic PAC learnability.

Definition 2 *Suppose algorithm $\mathcal{A}$ has access to D in addition to distribution P over X, and the output of $\mathcal{A}$ is a function $f : X \longrightarrow \mathbb{R}$. Execute $\mathcal{A}$ twice, using D_1 (respectively D_0) for P. Let f_1 and f_0 be the functions obtained respectively. For $x \in X$ let*

$$\begin{array}{rll} h(x) = 1 & \text{if} & f_1(x) > f_0(x) \\ h(x) = 0 & \text{if} & f_1(x) < f_0(x) \\ h(x) \text{ undefined} & \text{if} & f_1(x) = f_0(x) \end{array}$$

If $\mathcal{A}$ takes time polynomial in ε^{-1} and δ^{-1}, and h is PAC with respect to ε and δ, then we will say that $\mathcal{A}$ PAC-learns via discriminant functions with access to D.

POSEX learnability requires that f_1 be a $0/1$-valued function that constitutes a PAC hypothesis. We discuss the relationship in more detail in the next section.

Notation. We refer to the two instances of the unsupervised learning algorithm as $\mathcal{A}_1$ and $\mathcal{A}_0$, so that $\mathcal{A}_1$ consists of $\mathcal{A}$ with the (unlabeled) positive data as input, and $\mathcal{A}_0$ is $\mathcal{A}$ with the (unlabeled) negative data as input. (Note however that learning according to Definition 1 or 2 does not allow $\mathcal{A}_\ell$ to use the value of ℓ.) We let S_ℓ denote the sample of unlabeled data drawn by $\mathcal{A}_\ell$.

Let us conclude this section by considering alternative restrictions to PAC learnability that capture the informal notion of learning via discriminant functions. We could for example ask that f_1 and f_0 should be probability distributions. Our reason for not doing so is that, while we believe that the functions produced by our algorithms can be re-scaled on an ad-hoc basis to become probability distributions, there is no guarantee in the distribution-free PAC setting that they are similar to the unknown distributions D_1 and D_0.

Another natural candidate is a variant in which $\mathcal{A}_1$ and $\mathcal{A}_0$ are two distinct agents that know their class labels. Equivalently, we would be seeking two algorithms $\mathcal{A}_1$ and $\mathcal{A}_0$ that respectively have access to the positive and negative data (equivalently, an algorithm $\mathcal{A}_\ell$ that may use the value ℓ). Observe however that this relaxation is not helpful for any concept class $\mathcal{C}$ that is closed under complement (meaning that for all $c \in \mathcal{C}$, we also have $X \setminus c \in \mathcal{C}$). Consequently, any algorithms that require this relaxation would need to exploit additional prior information about data from different classes. There's nothing wrong with that of course, but it departs from our focus on the approach of independently processing each label class.

1.3 Related Work

There has been much work on the comparison of alternative notions of PAC learning with each other, with the criterion for distinguishability being that some learning task (concept class) should be tractable[1] in one variant but not in the other. The early paper of (Haussler et al., 1991) showed that various alternative definitions of PAC learnability are equivalent in this sense. Examples of distinctions that have been found between different restrictions to the framework include learning with a restricted focus of attention (Ben-David and Dichterman, 1998, 1994) which is shown to be more severe that learnability in the presence of uniform misclassification noise. Learnability with Statistical Queries (Kearns, 1998) is also known to be at least as severe as learnability with uniform misclassification noise. Perhaps the most important result of this kind is the equivalence between PAC learnability and "weak" PAC learnability found by (Schapire, 1990) which led to the development of boosting techniques. The paper (Blum, 1994) exhibits a concept class that distinguishes PAC learnability from mistake-bound learning, and that is of interest here since we use the same concept class (in Section 3.3) to show that our restriction of PAC learnability is likewise distinct from mistake-bound learnability.

We noted that learning under the constraint of Definition 2 is related to POSEX learning. Specifically we have:

Observation 1 *If a concept class and its complement are POSEX learnable, then they are learnable under Definition 2.*

Proof (sketch; the following technique is used in more detail in Theorem 4) Let $\mathcal{A}_P$ be a POSEX algorithm for $\mathcal{C}$ and let $\mathcal{A}'_P$ be a POSEX algorithm for $\mathcal{C}'$, the class of sets whose complements are

1. The word "tractable" is usually taken to mean that the computational and sample-size requirements for learning, should be polynomial in the parameters of the task.

members of $\mathcal{C}$. An algorithm $\mathcal{A}$ in the sense of Definition 2 works as follows. $\mathcal{A}$ can sample from D, and it uses samples from D_ℓ as "positive" examples. $\mathcal{A}$ applies $\mathcal{A}_P$ and $\mathcal{A}'_P$ to this data, and at least one of the two hypotheses h, h' is PAC, since the "positive" data really is positive from the perspective of one of $\mathcal{A}_P$ and $\mathcal{A}'_P$.

h and h' are then tested on further data; the chosen hypothesis should contain almost all samples from D_ℓ, and if both h and h' do so, choose the one that contains fewer samples from D. That hypothesis with high probability maps samples from D_ℓ to 1 and samples from $D_{1-\ell}$ to 0. Let $\mathcal{A}_1$ and $\mathcal{A}_0$ be the two runs of $\mathcal{A}$ having access to D_1 and D_0 respectively. The overall hypothesis is obtained from two functions found by $\mathcal{A}_1$ and $\mathcal{A}_0$, and its misclassification rate is at most the sum of their error rates. ■

In Section 2 we show that if a concept class is learnable in the presence of noise, then it and its complement are POSEX learnable, and hence by the above observation, learnable under Definition 2. The result of Section 2 answers a question raised by (Letouzey et al., 2000) (the hierarchies of inclusions given in Equations (4,5) of (Letouzey et al., 2000) can be merged).

There are some interesting algorithms having PAC-like performance guarantees for learning probability distributions; the topic was introduced in (Kearns et al., 1994), see also (Cryan et al., 2001; Freund and Mansour, 1999; Frieze et al., 1996; Dasgupta, 1999). The criterion for learning a distribution D is to obtain a hypothesis distribution which is within ε of D under some measure of similarity. The KL distance and the variation distance are usually considered. We noted above that when distributions are learned under these criteria, the Bayes classifier achieves agnostic PAC-ness. However, the algorithms we describe here differ substantially from these previous ones (as well as from the algorithms in the much more extensive general literature on unsupervised learning). The reason is that our aim is not really to approximate a distribution over inputs. Rather, it is to construct a discriminant function in such a way that we expect it to work well in conjunction with the corresponding discriminant function constructed on data with the opposite class label.

1.4 Summary of Results

The rest of the paper is organized as follows. In Section 2 we consider learning under Definition 2 in which $\mathcal{A}_\ell$ has access to D in addition to D_ℓ. We show that PAC learnability with uniform misclassification noise implies PAC learnability with discriminant functions and access to D. This gives us a good understanding of how learning under Definition 2 fits in with other restrictions.

In Section 3 we move to the trickier issue of learning according to Definition 1. We exhibit algorithms that show that the restriction is not more severe than various other restrictions of PAC learning. In Section 3.1 we show that parity functions are learnable in this setting, which distinguishes it from learnability in the presence of noise (subject to the "noisy parity assumption", which is the widespread assumption that parity functions are hard to learn in the presence of uniform misclassification noise) as well as Statistical Query (SQ) learnability (Kearns, 1998) (since it is known from (Kearns, 1998) that parity functions are not learnable using SQs.) In Section 3.2 we distinguish the setting from learnability from positive data only (or negative data only) by studying the class of unions of intervals on the real line. In Section 3.3 we distinguish the setting from mistake-bound learning, using a concept class from (Blum, 1994).

The two remaining algorithms indicate some limits to our success at finding algorithms in the restricted setting. Section 3.4 shows how to learn linear separators in the plane, using an approach

that we have not been able to extend to three or more dimensions. In Section 3.5 we show how to learn monomials provided that the input distribution D is an unknown product distribution. Learning under this sort of assumption is in a sense intermediate between learning with access to D (Definition 2) and learning via discriminant functions in the sense of Definition 1.

2. Learning via Discriminant Functions with Access to D

In this section we show that given a standard noise-tolerant PAC learning algorithm, we may use it to construct an algorithm for POSEX learning and hence learning in the restriction of Definition 2. We do this in two stages—Proposition 3 shows how this is achieved provided that an estimate of the class prior p_ℓ is provided to $\mathcal{A}_\ell$, and Theorem 4 extends the result to the setting in which the class priors are unknown.

Here is an overview of the proofs in this section. Proposition 3 analyses learning with a given noise rate. This uses a standard definition of noise-tolerant PAC learning in which an algorithm $\mathcal{A}^\eta$ has parameter η. η is the probability that an example is mislabeled; $0 \le \eta < \frac{1}{2}$. $\mathcal{A}^\eta$ should then be polynomial in $(\frac{1}{2}-\eta)^{-1}$ in addition to other parameters. Given the class prior p_ℓ (the probability that a random instance has label ℓ) we can generate random samples from a fixed distribution that is a mixture of D and D_ℓ, such that they have uniform misclassification rate, which allows $\mathcal{A}^\eta$ to be used. In fact, we show that an additive approximation $\overline{p_\ell}$ can be used in place of p_ℓ. This is done by exhibiting a coupling of the two labeled sample distributions (one using p_ℓ and the other using $\overline{p_\ell}$), in which they are very likely to generate the same data.

In Theorem 4 we exploit the fact that an approximation $\overline{p_\ell}$ can be used. The algorithm of Figure 2 tries out a sequence of values for $\overline{p_\ell}$, at least one of which is a good approximation to p_ℓ. The previous algorithm is used for each of these values, which generates a collection of hypotheses, and the empirically best hypothesis is shown to be PAC.

Proposition 3 *Let $\mathcal{A}^\eta$ be an algorithm with parameter η, $0 \le \eta < \frac{1}{2}$, that has access to labeled data, where elements of X are distributed according to D, with a uniform label noise rate of η. Suppose that $\mathcal{A}^\eta$ uses time $p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$ (where p is some polynomial), and with probability at least $1-\delta$ returns a hypothesis having error at most ε (with respect to D).*

For $\ell \in \{0,1\}$ let $p_\ell = \Pr_{x\sim D}(t(x)=\ell)$. Suppose that $|\overline{p_\ell} - p_\ell| \le \Delta / p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$. ($\Delta \in [0,1]$.) Suppose that the algorithm of Figure 1 is executed with input $\overline{p_\ell}$ and access to D_ℓ and D. Then with probability $1-\delta-\Delta$, the algorithm outputs $f_\ell : X \longrightarrow \{0,1\}$ satisfying

$$\Pr_{x\sim\frac{1}{2}(D+D_\ell)}(f_\ell(x) \ne t(x)) \le \varepsilon \quad \text{for} \quad \ell = 1$$
$$\Pr_{x\sim\frac{1}{2}(D+D_\ell)}(f_\ell(x) \ne 1-t(x)) \le \varepsilon \quad \text{for} \quad \ell = 0$$

Comment. The fact that f_ℓ has error at most ε for $x \sim \frac{1}{2}(D+D_\ell)$ implies that f_ℓ has error at most 2ε for $x \sim D$.

Proof We may assume that the concept class $\mathcal{C}$ is closed under complementation, since if $\mathcal{C}$ is learnable with misclassification noise then its closure under complementation is also learnable under misclassification noise.

Since $\mathcal{C}$ is closed under complementation, it suffices by symmetry to show that f_1 satisfies: with probability at least $1-\delta-\Delta$, $\Pr_{x\sim\frac{1}{2}(D+D_1)}(f_1(x) \ne t(x)) \le \varepsilon$.

Input $\overline{p_\ell}$, an estimate of $\mathcal{A}_\ell$'s class prior p_ℓ ($\ell \in \{0,1\}$).
Let $\eta = \frac{\overline{p_\ell}}{2\overline{p_\ell}+1}$; $N = p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$.

1. *Construct a labeled sample S_ℓ as follows. For $m = 1\ldots,N$ do:*
   (a) *Let c_m be a "fair coin" random variable; $c_m = 0$ or 1 with probability $\frac{1}{2}$; let ℓ_m be a $0/1$ random variable, $\ell_m = 1$ with probability $\frac{\overline{p_\ell}}{2\overline{p_\ell}+1}$.*
   (b) *If $c_m = 1$, sample x from D and let $(x, \ell_m) \in S_\ell$.*
   (c) *If $c_m = 0$, sample x from D_ℓ and let $(x,1) \in S_\ell$.*
2. *Input S_ℓ to $\mathcal{A}^\eta$ using $\eta = \frac{\overline{p_\ell}}{2\overline{p_\ell}+1}$ to obtain a hypothesis $h_\ell : X \longrightarrow \{0,1\}$.*
3. *$f_\ell(x) = h_\ell(x)$ for all $x \in X$.*

Figure 1: Learning in the sense of Definition 2 using noise-tolerant PAC algorithm and estimates of class priors

Let (x, j) be the element of S_1 constructed on the m-th iteration.

$$\begin{array}{rcl} \Pr(t(x)=0) & = & \Pr(c_m = 1)\Pr_{x\sim D}(t(x)=0) = \frac{1}{2}(1-p_1) \\ \Pr(t(x)=1) & = & 1-\frac{1}{2}(1-p_1) = \frac{1}{2}(1+p_1) \end{array} \tag{1}$$

Next we give expressions for misclassification rates $\Pr(j=0 \mid t(x)=1)$ and $\Pr(j=1 \mid t(x)=0)$. Consider first the case that $t(x)=1$. Note that

$$\begin{array}{rl} \Pr(j=0 \mid t(x)=1) = & \Pr(j=0 \mid t(x)=1 \wedge c_m = 0)\Pr(c_m = 0 \mid t(x)=1) \\ & +\Pr(j=0 \mid t(x)=1 \wedge c_m = 1)\Pr(c_m = 1 \mid t(x)=1). \end{array}$$

$\Pr(j=0 \mid t(x)=1 \wedge c_m = 0) = 0$, since if $c_m = 0$ then Step (1c) assigns label 1. Hence

$$\Pr(j=0 \mid t(x)=1) = \Pr(j=0 \mid t(x)=1 \wedge c_m = 1)\Pr(c_m = 1 \mid t(x)=1). \tag{2}$$

When $c_m = 1$, we have $j = \ell_m$ where $\ell_m = 1$ with probability $\overline{p_1}/(2\overline{p_1}+1)$, so

$$\Pr(j=0 \mid t(x)=1 \wedge c_m = 1) = 1 - \frac{\overline{p_1}}{2\overline{p_1}+1} = \frac{\overline{p_1}+1}{2\overline{p_1}+1}. \tag{3}$$

$$\Pr(c_m = 1 \mid t(x)=1) = \frac{\Pr(c_m=1)\Pr(t(x)=1 \mid c_m = 1)}{\Pr(t(x)=1)} = \frac{\frac{1}{2}p_1}{\Pr(t(x)=1)}.$$

$\Pr(t(x)=1) = \frac{1}{2}(1+p_1)$ by Equation (1), hence

$$\Pr(c_m = 1 \mid t(x)=1) = \frac{\frac{1}{2}p_1}{\frac{1}{2}(1+p_1)} = \frac{p_1}{1+p_1}. \tag{4}$$

Hence from Equations (2) and (3) and (4),

$$\Pr(j=0 \mid t(x)=1) = \Big(\frac{\overline{p_1}+1}{2\overline{p_1}+1}\Big)\Big(\frac{p_1}{1+p_1}\Big).$$

Now consider the case that $t(x)=0$ (where (x,j) is the labeled exampled constructed on the m-th iteration of $\mathcal{A}_1$); note that

$$\begin{aligned}\Pr(j=1 \mid t(x)=0) = \quad & \Pr(j=1 \mid t(x)=0 \wedge c_m=0)\Pr(c_m=0 \mid t(x)=0) \\ & +\Pr(j=1 \mid t(x)=0 \wedge c_m=1)\Pr(c_m=1 \mid t(x)=0).\end{aligned}$$

If $c_m=0$ then from Step (1c), $t(x)=1$. Hence

$$\begin{aligned}\Pr(c_m=1 \mid t(x)=0) &= 1 \\ \Pr(c_m=0 \mid t(x)=0) &= 0.\end{aligned}$$

Consequently,

$$\Pr(j=1 \mid t(x)=0) = \Pr(j=1 \mid t(x)=0 \wedge c_m=1). \tag{5}$$

When $c_m=1$ we have $j=\ell_m$ where $\ell_m=1$ with probability $\overline{p_1}/(2\overline{p_1}+1)$, so

$$\Pr(j=1 \mid t(x)=0 \wedge c_m=1) = \frac{\overline{p_1}}{2\overline{p_1}+1}. \tag{6}$$

From (5) and (6),

$$\Pr(j=1 \mid t(x)=0) = \frac{\overline{p_1}}{2\overline{p_1}+1}.$$

Based on the above expressions for the misclassification rates $\Pr(j=0 \mid t(x)=1)$ and $\Pr(j=1 \mid t(x)=0)$, and noting that N is defined in Figure 1, Step (1) of the algorithm of Figure 1 is equivalent to the following:

1. For $m=1\ldots N$ do:
   (a) Sample x from the mixture $\frac{1}{2}(D+D_1)$.
   (b) Sample r_m uniformly at random from $[0,1]$.
   (c) If $t(x)=1$, then if $r_m < (\frac{\overline{p_1}+1}{2\overline{p_1}+1})(\frac{p_1}{1+p_1})$ label x incorrectly else label x correctly.
   (d) If $t(x)=0$, then if $r_m < \frac{\overline{p_1}}{2\overline{p_1}+1}$ label x incorrectly else label x correctly.

Let $D_{\mathcal{A}}$ be the above distribution over samples of size N. Let D_η be the following distribution over samples of size N:

1. For $m=1\ldots N$ do:
   (a) Sample x from the mixture $\frac{1}{2}(D+D_1)$.
   (b) Sample r_m uniformly at random from $[0,1]$.
   (c) If $r_m < \frac{\overline{p_1}}{2\overline{p_1}+1}$ label x incorrectly else label x correctly.

Let S_N denote the set of labeled samples of size N. Define a distribution D_{2N} over $S_N \times S_N$ by using the same sequence of x's and r_m and the above procedures for constructing the labeled samples, so that the two marginal distributions over S_N are $D_{\mathcal{A}}$ and D_η. For $(S,S') \sim D_{2N}$,

$$\Pr(S \neq S') \leq N \cdot \Pr_{x \sim \frac{1}{2}(D+D_1)}(t(x)=1) \cdot \left| \left(\frac{\overline{p_1}+1}{2\overline{p_1}+1}\right)\left(\frac{p_1}{1+p_1}\right) - \frac{\overline{p_1}}{2\overline{p_1}+1} \right|.$$

This is because there are N opportunities for S to differ from S', and this occurs when x sampled from $\frac{1}{2}(D+D')$ satisfies $t(x)=1$. In that case, the labels will differ when r_m lies between $\frac{\overline{p_1}}{2\overline{p_1}+1}$ and $(\frac{\overline{p_1}+1}{2\overline{p_1}+1})(\frac{p_1}{1+p_1})$. Consequently,

$$\Pr(S \neq S') \leq N \cdot \frac{|\overline{p_1} - p_1(\frac{1+\overline{p_1}}{1+p_1})|}{2\overline{p_1}+1} \leq N \cdot (2\overline{p_1}+1)^{-1} \cdot |\overline{p_1} - p_1| \leq N|\overline{p_1} - p_1|.$$

By definition of $\mathcal{A}^\eta$, for $S \sim D_\eta$, $N = p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$, $\eta = \frac{\overline{p_1}}{2\overline{p_1}+1}$, with probability $1-\delta$, $\mathcal{A}^\eta$ on input S returns h' having error $\Pr(h'(x) \neq t(x)) \leq \varepsilon$ for $x \sim \frac{1}{2}(D+D_1)$.

Hence for $S \sim D_{\mathcal{A}}$, $N = p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$, with probability $1-\delta-N(|p_1-\overline{p_1}|)$, $\mathcal{A}^\eta$ on input S returns h' having error $\Pr(h'(x) \neq t(x)) \leq \varepsilon$ for $x \sim \frac{1}{2}(D+D_1)$. Given our assumption that $|p_1 - \overline{p_1}| \leq \Delta/N = \Delta/p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$, the result follows. ■

1. *Let* $p(\varepsilon,\delta) = \max_{0 \leq \eta \leq 1/3} p(\varepsilon^{-1}, \delta^{-1}, (\frac{1}{2}-\eta)^{-1})$*; where* $p(\cdot,\cdot,\cdot)$ *is the sample size in terms of error, uncertainty and noise rate used by* $\mathcal{A}^\eta$ *in Figure 1; Let* $\Delta = \delta/32p(\varepsilon,\delta)$*;* $N = (16/\varepsilon)^2 \log(128p(\varepsilon,\delta)/\delta^2)$*;* $\mathcal{H} = \emptyset$.
2. *For all* $\overline{p_\ell} \in [0,1]$ *such that* $\overline{p_\ell} = k\Delta$ *for* $k \in \mathbb{N}$ *do:*
   (a) *Apply the algorithm of Figure 1 with parameters* $\frac{1}{16}\varepsilon$, $\frac{1}{4}\delta$.
   (b) *If* $h : X \longrightarrow \{0,1\}$ *is returned, add h to* $\mathcal{H}$.
3. *Draw an unlabeled sample* S_ℓ *of size N using* D_ℓ.
4. *For each* $h \in \mathcal{H}$*, if* $|\{x \in S_\ell \ : \ h(x)=1\}| < (1-\frac{3}{16}\varepsilon)|S_\ell|$ *then remove h from* $\mathcal{H}$.
5. *Draw a unlabeled sample S of size N using D.*
6. *Let* $h' = \arg\min_{h \in \mathcal{H}} |\{x \in S \ : \ h(x)=1\}|$.
7. $f_\ell(x) = h'(x)$ *for* $x \in X$.

Figure 2: Learning in the sense of Definition 2 using noise-tolerant PAC algorithm and unknown class priors

Theorem 4 *Let $\mathcal{A}^\eta$ be a noise-tolerant algorithm as defined in Proposition 3.*

For $\ell \in \{0,1\}$, the Algorithm of Figure 2 given access to D_ℓ and D, with probability at least $1-\delta$ outputs (in polynomial time) f_ℓ with $\Pr_{x\sim D}(f_\ell(x) \neq t(x)) \leq \varepsilon$ for $\ell = 1$, and $\Pr_{x\sim D}(f_\ell(x) \neq 1-t(x)) \leq \varepsilon$ for $\ell = 0$.

Comment. As a consequence we have learnability in the sense of Definition 2, since when we derive classifier h from f_1 and f_0, the error of h is at most the sum of the errors of f_1 and f_0. (By the error of f_0 we mean the probability that $f_0(x)$ is not equal to $1-t(x)$.)

Proof Let $\widehat{\Pr}_{x\in S}(\pi(x))$ denote the empirical probability that x satisfies property π, with respect to sample S. We show first that the expression for N used by the algorithm of Figure 2 guarantees that with probability at least $1-\frac{1}{2}\delta$,

$$\forall h \in \mathcal{H} \quad \left| \Pr_{x\sim D_\ell}(h(x)=1) - \widehat{\Pr}_{x\in S_\ell}(h(x)=1) \right| \leq \frac{1}{16}\varepsilon \tag{7}$$

$$\forall h \in \mathcal{H} \quad \left| \Pr_{x\sim D}(h(x)=1) - \widehat{\Pr}_{x\in S}(h(x)=1) \right| \leq \frac{1}{16}\varepsilon \tag{8}$$

We are asking that the relative frequencies (over N observations) of a set of at most $2|\mathcal{H}|$ events should be within $\frac{1}{16}\varepsilon$ of their probabilities. Taking a union bound, it is sufficient that N should satisfy: Given any $f : X \longrightarrow \{0,1\}$, with probability at least $1-\delta/(4|\mathcal{H}|)$

$$\left| \Pr_{x\sim D}(f(x)=1) - \widehat{\Pr}_{x\in S}(f(x)=1) \right| \leq \frac{1}{16}\varepsilon.$$

Recall Hoeffding's inequality: Let $Y_1,\ldots,Y_N$ be Bernoulli trials with probability p of success. Let $T = Y_1+\ldots+Y_N$ denote the total number of successes. Then for $\gamma \in [0,1]$, $\Pr(|T-pN| > \gamma N) \leq 2e^{-2N\gamma^2}$.

This means that N is sufficiently large if N satisfies $\delta/(4|\mathcal{H}|) \geq 2e^{-2N(\varepsilon/16)^2}$. Since $|\mathcal{H}| \leq 1/\Delta$, it is sufficient for N to satisfy $\delta\Delta/4 \geq 2e^{-2N(\varepsilon/16)^2}$.

For $\Delta = \delta/32p(\varepsilon,\delta)$,

$$\delta^2/128p(\varepsilon,\delta) \geq 2e^{-2N(\varepsilon/16)^2}$$

The equation is satisfied by putting $N = (16/\varepsilon)^2 \cdot \log(128p(\varepsilon,\delta)/\delta^2)$, polynomial in the parameters.

Assume that $\ell = 1$. We assume as before that the concept class is closed under complementation, so that the proof for $\ell = 0$ is similar but using the complement of t in place of t.

Note that the algorithm of Figure 1 constructs a noise rate η in the range $[0,\frac{1}{3}]$ based on $\overline{p_\ell}$, so each application of Algorithm 1 in Step (2a) uses sample size at most $p(\frac{1}{16}\varepsilon^{-1}, \frac{1}{4}\delta^{-1})$. (Polynomial $p(\cdot,\cdot)$ is defined in Figure 2.)

One of the values of $\overline{p_\ell}$ used in Step (2a) as input to Algorithm 1 satisfies $|\overline{p_\ell} - p_\ell| < \Delta$. As a result, applying Proposition 3 we have that with probability $1-\frac{1}{2}\delta$, there exists $h^* \in \mathcal{H}$ satisfying

$$\Pr_{x\sim\frac{1}{2}(D+D_1)}(h^*(x) \neq t(x)) \leq \frac{1}{16}\varepsilon.$$

We may deduce that

$$\begin{aligned} &\Pr_{x\sim D_1}(h^*(x) \neq t(x)) \leq \tfrac{1}{8}\varepsilon \\ &\Pr_{x\sim D}(h^*(x) \neq t(x)) \leq \tfrac{1}{8}\varepsilon. \end{aligned} \tag{9}$$

We show that with probability $1-\frac{1}{2}\delta$

1. for $h \in \mathcal{H}$, if $\Pr_{x\sim D}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\varepsilon$ then h is eliminated in Step 4.
2. h^* is not eliminated in Step 4.
3. For $h \in \mathcal{H}$, if $\Pr_{x\sim D}(h(x) = 1 \wedge t(x) = 0) > \frac{1}{2}\varepsilon$ then h is eliminated in Step 6.

Suppose that $\Pr_{x\sim D}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\varepsilon$. Then $\Pr_{x\sim D_1}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\varepsilon/p_1 \geq \frac{1}{4}\varepsilon$. Hence by (7),

$$\widehat{\Pr}_{x\in S_1}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\varepsilon - \frac{1}{16}\varepsilon = \frac{3}{16}\varepsilon.$$

From (7) and (9),

$$\widehat{\Pr}_{x\in S_1}(h^*(x) = 0 \wedge t(x) = 1) \leq \frac{1}{8}\varepsilon + \frac{1}{16}\varepsilon = \frac{3}{16}\varepsilon.$$

Hence Step 4 does not eliminate h^* but it eliminates all h with $\Pr_{x\sim D}(h(x) = 0 \wedge t(x) = 1) > \frac{1}{4}\varepsilon$.

Now suppose that $\Pr_{x\sim D}(h''(x) = 1 \wedge t(x) = 0) > \frac{1}{2}\varepsilon$ for some $h'' \in \mathcal{H}$ after Step 4. We have just shown that h'' satisfies $\Pr_{x\sim D}(h''(x) = 0 \wedge t(x) = 1) \leq \frac{1}{4}\varepsilon$. Consequently, $\Pr_{x\sim D}(h''(x) = 1) - \Pr_{x\sim D}(t(x) = 1) > \frac{1}{2}\varepsilon - \frac{1}{4}\varepsilon = \frac{1}{4}\varepsilon$. Meanwhile note from (9) that $\Pr_{x\sim D}(h^*(x) = 1) - \Pr_{x\sim D}(t(x) = 1) \leq \frac{1}{8}\varepsilon$. As a result, $\Pr_{x\sim D}(h''(x) = 1) - \Pr_{x\sim D}(h^*(x) = 1) > \frac{1}{4}\varepsilon - \frac{1}{8}\varepsilon = \frac{1}{8}\varepsilon$. From (8),

$$\widehat{\Pr}_{x\in S}(h''(x) = 1) - \widehat{\Pr}_{x\in S}(h^*(x) = 1) > 0.$$

Hence h'' is eliminated at Step 6.

Hence all $h \in \mathcal{H}$ with error at least ε are eliminated with probability $1 - \frac{1}{2}\delta$. With probability at least $1 - \frac{1}{2}\delta$ there exists $h^* \in \mathcal{H}$ with error less that ε. Putting these together, with probability at least $1 - \delta$ we are left with h' having error less than ε. ■

3. Learning via Discriminant Functions without Access to D

We exhibit algorithms that show that learnability in the sense of Definition 1 is distinct from various well-known restrictions of PAC learnability. We also study a special case of the problem of learning monomials (in which D is known to belong to a particular class of distributions), for which we have no algorithm in the distribution-independent setting.

Our algorithms are mostly proven to have the PAC property in a standard way, by arguing that the hypothesis is consistent with the data, and furthermore that it belongs to a class of hypotheses that have description length polynomial in the parameters of the problem, and sub-linear in the sample size. (This is the "Occam algorithm" property (Blumer et al., 1987)). For Sections 3.2 and 3.4 we use the (more generally applicable) Vapnik-Chervonenkis dimension (Blumer et al., 1989; Vapnik, 2000) of the class of hypotheses.

3.1 Parity Functions

The following result distinguishes our learning setting from learnability with uniform misclassification noise, or learnability with a restricted focus of attention.

An instance is an element of $\{0,1\}^n$, representing a sequence of values of n boolean variables. A *parity function* (Helmbold et al., 1992) has an associated subset of the variables, and an associated "target parity" (even or odd), and evaluates to 1 provided that the parity of the number of "true" elements of that subset agrees with the target parity, otherwise the function evaluates to 0.

Theorem 5 *The class of parity functions is PAC learnable via discriminant functions.*

Proof Observe that the class is closed under complementation.

To learn a parity function from positive examples only, in an essentially similar way to the algorithm of (Helmbold et al., 1992), $\mathcal{A}_\ell$ finds the affine subspace of $GF(2)^n$ spanned by its examples, and f_ℓ assigns a value of 1 to elements of that subspace and a value of 0 to all other elements of the domain. (By "span" we mean with respect to $GF(2)^n$, as opposed to $\mathbb{R}^n$. $GF(2)$, the generic field with two elements 0 and 1, has addition modulo 2, so that the sum of two $0/1$ vectors is their bitwise exclusive-or. Generally the positive examples of a parity function would span all of $\mathbb{R}^n$.)

In more detail, let x_i be the i-th entry of bit vector x. The positive examples of a parity function satisfy $\sum_i c_i x_i = b$ (all addition and multiplication modulo 2), where $c_i, b \in GF(2)$. Let x' be an arbitrary positive example; positive examples x satisfy $\sum_i c_i(x_i - x'_i) = 0$, and negative examples do not satisfy this. Hence, the subspace constructed by $\mathcal{A}_1$ will be a subset of $\sum_i c_i(x_i - x'_i) = 0$, and will contain no negative examples. $\mathcal{A}_0$ constructs a subspace that contains all the negative data and no positive examples.

We have that $f_\ell(x) = 0$ for all x with $t(x) = 1 - \ell$, and $f_\ell(x) = 1$ for all $x \in S_\ell$ (the unlabeled sample obtained by $\mathcal{A}_\ell$).

The overall hypothesis h has description length $O(n^2)$ (a spanning set has at most n vectors, each of length n) and h is consistent with the training data; thus we have PAC-ness by the standard Occam-algorithm argument. ■

3.2 Unions of Intervals

Let $X = \mathbb{R}$, and let $t : X \longrightarrow \{0,1\}$ be the indicator function of a union of k intervals in $\mathbb{R}$. We show that the class of all such functions, is learnable by discriminant functions in time polynomial in ε^{-1}, δ^{-1} and k. A union of more than one interval cannot be PAC-learned from just positive or just negative data, simply because it is impossible to guess where the data with the opposite label may lie. Learnability via discriminant functions is thus distinct from learnability from positive examples only, or from negative examples only.

Theorem 6 *The class of unions of k intervals on the real line is learnable via discriminant functions.*

Proof Construct discriminant functions f_0 and f_1 as follows. Given an (unlabeled) sample, and a point $x \in \mathbb{R}$, our discriminant function maps x to the negation of its distance to its nearest neighbor in the sample. (Intuitively, it makes sense that x should get a higher value if it is close to a data point in the sample.) We show furthermore that this rule creates a classifier that is "simple" (a union of k intervals) and consistent with the data.

More precisely, given (unlabeled) sample $S_\ell \subset \mathbb{R}$ of size $O(k\log(\delta^{-1}\varepsilon^{-1})/\varepsilon)$, let $d_{NN}(x, S_\ell) = \min_{x_\ell \in S_\ell}\{|x - x_\ell|\}$. Let $f_\ell(x) = -d_{NN}(x, S_\ell)$. Recall that $h(x) = 1$ if $f_1(x) > f_0(x)$ and $h(x) = 0$ if $f_1(x) < f_0(x)$. We show that h is PAC.

For $x, x' \in S_\ell$, suppose x, x' belong to the same interval of $t^{-1}(\ell)$. Then $[x, x'] \subseteq h^{-1}(\ell)$, since any point between x and x' is closer to at least one of x or x' than to any point x'' for which $t(x'') \neq t(x)$.

Suppose $x_0 \in S_0$, $x_1 \in S_1$, $x_0 < x_1$, and there does not exist $x \in S_0 \cup S_1$ with $x_0 < x < x_1$. For a real number x such that $x_0 < x < x_1$, if $x \in (x_0, \frac{1}{2}(x_0 + x_1))$ then $d_{NN}(x, S_0) < d_{NN}(x, S_1)$, so $f_0(x) > f_1(x)$

and $h(x) = 0$ for h constructed according to Definition 1. Similarly, for $x \in (\frac{1}{2}(x_0 + x_1), x_1)$, $h(x) = 1$. $h(\frac{1}{2}(x_0 + x_1))$ is undefined.

For $S_0 \cup S_1$ sorted in ascending order, there are at most $2k$ pairs of consecutive points x, x' in the sequence where $t(x) \neq t(x')$.

Hence h is undefined on at most $2k$ elements of $\mathbb{R}$, and $h^{-1}(1)$ is a union of at most k intervals, and $h^{-1}(0)$ is a union of at most $k+1$ intervals. The VC dimension of unions of k intervals is $2k$, so using the results of (Blumer et al., 1987), the sample size required for PAC learning is $O(k \log(\delta^{-1}\varepsilon^{-1})/\varepsilon)$. ■

Comment. This nearest-neighbour rule does not work in more than one dimension, given that the input distribution D is closen by an adversary. Suppose we wish to learn a linear threshold in the plane $\mathbb{R}^2$. Suppose D is uniform over two parallel line segments that are very close but on opposite sides of the classification threshold. Then the probability is only about $\frac{1}{2}$ that the nearest neighbour of a data point x will have the same label as x. In Section 3.4 we show how to learn linear thresholds in the plane using a more sophisticated rule.

3.3 Distinguishing the Model from the Mistake-bound Setting

In (Blum, 1994), Blum exhibits a concept class that is PAC learnable, but is not (in polynomial time) learnable using membership and equivalence queries, assuming that one-way functions exist. In this section we show that the concept class is PAC learnable via discriminant functions in the sense of Definition 1. We review the concept class introduced in (Blum, 1994). Let $X = \{0,1\}^n$.

If A is a probabilistic polynomial-time algorithm that computes a function from $\{0,1\}^*$ to $\{0,1\}$, and g is some function from $\{0,1\}^*$ to $\{0,1\}^*$, let $\mathcal{P}_k(A, g(s))$ denote the probability that $A(g(s)) = 1$ for strings s of length k generated uniformly at random.

Let G be a Cryptographically Strong Pseudorandom Bit (CSB) generator with stretch $p(k) = 2k$. For polynomial p a CSB generator is defined as follows.

Definition 7 *A deterministic polynomial-time program G is a CSB generator with stretch p if on input $s \in \{0,1\}^k$ it produces an output in $\{0,1\}^{p(k)}$ and for all probabilistic polynomial-time algorithms A, for all polynomials Q, for sufficiently large k (k depending on A and Q),*

$$|\mathcal{P}_k(A, G(s)) - \mathcal{P}_{p(k)}(A, s)| < \frac{1}{Q(k)}.$$

Thus, no polynomial-time algorithm can distinguish between strings generated uniformly at random from $\{0,1\}^{2k}$, and strings obtained by taking the output of G for a random input string of length k. (Technically, the definition allows A to be a circuit family.)

For strings x and y, let $x \circ y$ denote their concatenation. For a bit string x let $LSB[x]$ denote the rightmost bit of x. Let λ denote the empty string. For bit string s of length k, $G(s)$ is a bit string of length $2k$, and we define the following notation.

1. Let $G_0(s)$ be the leftmost k bits of $G(s)$.
2. Let $G_1(s)$ be the rightmost k bits of $G(s)$.
3. Let $G'_0(s)$ be the rightmost k bits of $G(s)$.

4. Let $G'_1(s) = \lambda$.

5. If $i = i_1 \cdots i_d$ (where the i_j are binary digits), let $G_i(s) = G_{i_d}(G_{i_{d-1}}(\cdots G_{i_1}(s)))$.

The concept class is defined as follows:

Definition 8 *Let $k = \lfloor \sqrt{n} \rfloor - 1$ and let $\mathcal{C} = \{c_s\}_{s \in \{0,1\}^k}$ where c_s is defined as follows.*

- *c_s is the indicator function of $\{x_s^i : i \in \{0,1\}^k \text{ and } \mathrm{LSB}[G_{i_1 \cdots i_k}(s)] = 1\}$*

where for $i = i_1 \cdots i_k$,

- $x_s^i = i \circ G'_{i_1}(s) \circ G'_{i_2}(G_{i_1}(s)) \circ G'_{i_3}(G_{i_1 i_2}(s)) \circ \ldots \circ G'_{i_k}(G_{i_1 \cdots i_{k-1}}(s)) \circ 0^w$

where w is chosen to ensure that $|x_s^i| = n$.

Definition 8 is slightly different from the corresponding definition of (Blum, 1994), where $k = \lfloor \sqrt{n} \rfloor$. We use $k = \lfloor \sqrt{n} \rfloor - 1$ so that the length of x_s^i is always less that n, and we can then "pad it out" to a length of exactly n using the string 0^w on the right-hand side.

Note that for any fixed s, a bit string of length n of the form x_s^i is determined entirely by i, its first k bits. We will let the *index* of a bit string of length n refer to its first k bits, viewed as a binary number (to give the natural ordering on indices). For a string x let $index(x)$ denote this number, regardless of whether x is well-formed according to Definition 8. (If x is not well-formed, x is a negative example of c_s, i.e. $c_s(x) = 0$.) Algorithm Compute-Forward (Figure 3) shows how to take any positive example x_s^i, together with an index $j > i$, and construct the pair $\langle x_s^j, c_s(x_s^j) \rangle$ in polynomial time.

The following notation is used in Algorithm Compute-Forward:

1. Let z_s^i be the correctly labeled example $\langle x_s^i, c_s(x_s^i) \rangle$.

2. Let $\overline{z_s^i}$ be the incorrectly labeled example $\langle x_s^i, 1 - c_s(x_s^i) \rangle$.

3. For $i_1, \ldots, i_d \in \{0,1\}^d$, let $G^{i_1 \cdots i_d}(s) = G'_{i_1}(s) \circ G'_{i_2}(G_{i_1}(s)) \circ \ldots \circ G'_{i_d}(G_{i_1 \cdots i_{d-1}}(s))$.

So $x_s^i = i \circ G^{i_1 \cdots i_k}(s) \circ 0^w$.

From (Blum, 1994) we know that $\mathcal{C}$ is not learnable (in time polynomial in n) in the mistake-bound model. We review the PAC learning algorithm of (Blum, 1994) and show how to adapt it to the constraint of Definition 1.

We noted that Algorithm Compute-Forward, given a positive example x_s^i and $j > i$, produces a correctly-labeled example $\langle x_s^j, c_s(x_s^j) \rangle$. Based on this observation, we assign values to examples as shown in Figure 4.

Theorem 9 *The concept class of (Blum, 1994) is learnable via discriminant functions.*

Proof We use the algorithm of Figure 4 to construct discriminant functions.

Recall that for $x \in \{0,1\}^n$, $index(x)$ denotes the k bit binary number forming a prefix of x, for $k = \lfloor \sqrt{n} \rfloor - 1$. For $\ell \in \{0,1\}$, $\mathcal{A}_\ell$ denotes the instance of the algorithm that is given access to D_ℓ.

As in (Blum, 1994), we will argue that what we have is an "Occam Algorithm" in the sense of (Blumer et al., 1987) which is consistent with the training data. Specifically, $\mathcal{A}_1$ and $\mathcal{A}_0$ memorize

Algorithm Compute-Forward (Blum, 1994)
On input x_s^i and $j > i$,

1. *Say $i = i_1 \cdots i_k$ and $j = j_1 \cdots j_k$. Let r be the least index such that $i_r \neq j_r$. Since $j > i$ we have $i_r = 0$ and $j_r = 1$.*

2. *Extract from x_s^i the portions:*

$$\begin{aligned} u &= G'_{i_1}(s) \circ G'_{i_2}(G_{i_1}(s)) \circ G'_{i_3}(G_{i_1 i_2}(s)) \circ \ldots \circ G'_{i_{r-1}}(G_{i_1 \cdots i_{r-2}}(s)) \\ &= G^{i_1 \cdots i_{r-1}}(s). \\ v &= G'_{i_r}(G_{i_1 \cdots i_{r-1}}(s)) = G_{j_r}(G_{j_1 \cdots j_{r-1}}(s)). \end{aligned}$$

3. *Notice that $G'_{j_r}(G_{j_1 \cdots j_{r-1}}(s)) = \lambda$. Since $v = G_{j_r}(G_{j_1 \cdots j_{r-1}}(s))$, we can use v as an intermediate point in the computation of those parts of z_s^j that differ from z_s^i.*

4. *If $r = k$, output: $\langle j \circ u \circ \lambda, \mathrm{LSB}[v] \rangle$. Otherwise, output: $\langle j \circ u \circ \lambda \circ G^{j_{r+1} \cdots j_k}(v), \mathrm{LSB}[G_{j_k \cdots j_{r+1}}(v)] \rangle$.*

Figure 3: Algorithm from (Blum, 1994)

at most 2 training examples each ($\mathcal{A}_0$ possibly memorizes none) and their combined hypothesis (the h in Definition 1) is consistent with the training data.

In particular, $\mathcal{A}_1$ (and possibly also $\mathcal{A}_0$) just retains x_s^m and x_s^M, since for any unlabeled x, the label assigned to it is computed (in polynomial time) using x_s^m and x_s^M. (In the case of $\mathcal{A}_0$, the sample S_0 may fail the test *Consistency-Check*, in which case no examples are memorized.) Hence the description length of the rule that labels examples, is $O(n)$.

Note that f_1 from $\mathcal{A}_1$ will now give a value of 1 to any positive example whose index is between the largest and smallest indices it has seen so far, and will give value of 0 to all other examples. If an example $x \in X$ is either negative or is ill-formed ("bad" in the terminology of (Blum, 1994)), then Step 4 will ensure $f_1(x) = 0$, even if $index(x)$ is between m and M.

At the same time, we claim that f_0 from $\mathcal{A}_0$ gives a value of $\leq \frac{1}{2}$ to all positive examples. Suppose for a contradiction that $\mathcal{A}_0$ gives a value of 1 to positive example x_s^j. Then $\mathcal{A}_0$ must have in its collection an unlabeled example x_s^M and x_s^j must predict x_s^M as being positive. But that implies that x_s^M must be positive, and since it belongs to S_0 it is negative, a contradiction.

$\mathcal{A}_0$ ensures that $f_0(x) \geq \frac{1}{2}$ for all $x \in S_0$. $\mathcal{A}_1$ ensures that $f_1(x) \geq 1$ for all $x \in S_1$ and $f_1(x) = 0$ for all negative x (including all $x \in S_0$). Hence the combined classifier h is consistent with the training data. ■

Input S_ℓ, a sample of unlabeled elements of $X = \{0,1\}^n$.

1. *Apply algorithm Consistency-Check to S_ℓ; if S_ℓ fails the test, then for all $x \in X$, $f_\ell(x) = \frac{1}{2}$. Else:*
2. *Let m and M be the minimum and maximum indices of elements of S_ℓ. Call these elements x_s^m and x_s^M respectively; since Consistency-Check has been passed, they are unique. For $x \in X$, index$(x) = j$, if $j > M$ or $j < m$ then $f_\ell(x) = 0$.*
3. *If $x \in S_\ell$ then $f_\ell(x) = 1$.*
4. *Otherwise, if $\langle x_s^j, 1\rangle$ =Compute-Forward(x_s^m, j) and furthermore, $\langle x_s^M, 1\rangle$ =Compute-Forward(x_s^j, M) then let $f_\ell(x_s^j) = 1$.*
5. *Otherwise, let $f_\ell(x_s^m) = 0$.*

Algorithm Consistency-Check

1. *If there exist $x_1, x_2 \in S_\ell$ with $x_1 \neq x_2$, but index$(x_1) =$ index(x_2), then fail.*
2. *If there exist $x_1, x_2 \in S_\ell$ such that index$(x_2) >$ index(x_1), yet with Compute-Forward(x_1, index(x_2))$\neq \langle x_2, 1\rangle$, then fail.*

Figure 4: Assigning values to unlabeled data for concept class of (Blum, 1994)

3.4 Linear Separators in the Plane

For $X = \mathbb{R}^2$, suppose each $x \in X$ is labeled 0 or 1 according to whether its coordinates satisfy some linear inequality; that is, a concept is a half-space in $\mathbb{R}^2$. This problem is well-known to be PAC-learnable in the standard setting; generally for $X = \mathbb{R}^n$ it reduces to linear programming.

Given a sample S_ℓ of points in $t^{-1}(\ell)$, note that points within their convex hull[2] ought to receive a "high" value from f_ℓ, since the convex hull must be a subset of $t^{-1}(\ell)$. We need to be able to deal with the case when the convex hull has most or all of S_ℓ at its vertices, as would happen for an input distribution D_ℓ whose domain is the boundary of a circle, for example. Our general approach is to start out by computing the convex hull P and give maximal value to points inside P. Then give an intermediate value to points in a polygon Q containing P, where Q has fewer edges. We argue that the way Q is chosen ensures that most points in Q are indeed given the correct label.

Theorem 10 *Linear separators in the plane are learnable via discriminant functions.*

2. The *convex hull* of a finite set S of points is the smallest convex polygon (more generally, polytope) that contains S. Any vertex of the convex hull of S is a member of S.

1. *Draw a sample S_ℓ of size $N = \Theta(\log(1/\delta\varepsilon)/\varepsilon^2)$.*
2. *Let polygon P_ℓ be the convex hull of S_ℓ.*
3. *Let Q_ℓ be a polygon having at most $\lceil\sqrt{N}\rceil$ edges such that*
   (a) *Every edge of Q_ℓ intersects P_ℓ at a single vertex, and*
   (b) *Adjacent edges of Q_ℓ contain vertices of P_ℓ that are at most $\sqrt{N}$ apart in the adjacency sequence of P_ℓ's vertices.*
4. *Define discriminant function f_ℓ as follows.*
   (a) *For all x in the interior or boundary of P_ℓ, $f_\ell(x) = 1$.*
   (b) *For each connected region R in $Q_\ell \setminus P_\ell$ let $A(R)$ denote its area. For $x \in R$ let $f_\ell(x) = (A(R)+1)^{-1}$. If $A(R)$ is infinite let $f_\ell(x) = 0$.*
   (c) *For $x \notin Q_\ell$ let $f_\ell(x) = -1$.*

Figure 5: Assigning values to unlabeled data for linear separators in the plane

Proof Figure 5 shows the algorithm we use to construct discriminant functions; it is not hard to check that the steps can be carried out in polynomial time. Figure 6 illustrates the construction on an example.

Let $h : \mathbb{R}^2 \longrightarrow \{0,1\}$ be the hypothesis constructed from f_0 and f_1. We show below that for $\ell \in \{0,1\}$, $h^{-1}(\ell)$ is a region bounded by $O(\sqrt{N})$ line segments. We also show that h is consistent with the data, i.e. that for $x \in S_\ell$ (the unlabeled sample drawn by $\mathcal{A}_\ell$), we have $h(x) = \ell$. As before, PAC-ness follows from an Occam-algorithm argument; the class of hypotheses has VC dimension $O(\sqrt{N})$, sublinear in the sample size.

To show consistency of the hypothesis, suppose $x \in S_1$, i.e. x is a positive example. Then $f_1(x) = 1$ since x lies in the interior or on the boundary of P_1 (rule 4a). By contrast, when f_0 is constructed, x lies strictly outside the convex hull of the negative data, so either rule 4b or 4c is applied, giving $f_0(x)$ a value less than 1. By symmetry, members of S_0 are also correctly labeled.

Next we prove our claim that the boundary between the points labeled 0 by h, and the points labeled 1, is indeed simple. (Specifically, $h^{-1}(0)$ and $h^{-1}(1)$ are bounded by $O(\sqrt{N})$ line segments.) Let L be the line that defines the target linear threshold function t. Let $\mathcal{R}_\ell$ be the set of connected regions constructed by $\mathcal{A}_\ell$ that lie between P_ℓ and Q_ℓ. For $R \subseteq \mathbb{R}^2$ let $CH(R)$ denote the convex hull of R. Observe that

1. no straight line may pass through more than 2 elements of $\mathcal{R}_\ell$. (If that occurred, suppose the line passes through $R, R', R'' \in \mathcal{R}_\ell$ in that order. Note that $P_\ell \cup R \cup R''$ is convex. That makes it impossible for the line to cut R', which is outside $P_\ell \cup R \cup R''$.)

2. at most one element of $\mathcal{R}_0$ (respectively, $\mathcal{R}_1$) may intersect L. (If two of them intersected L, there would be an edge of Q_ℓ on the opposite side of L from P_ℓ, hence a vertex of P_ℓ on the wrong side of L.)

3. for $R \in \mathcal{R}_\ell$, $CH(R)$ is a region bounded by three line segments. (R has two "outer" edges and a concave sequence of edges from P_ℓ connecting them.)

Suppose that $R_0 \in \mathcal{R}_0$ intersects $R_1 \in \mathcal{R}_1$. Then $CH(R_0)$ intersects $CH(R_1)$. From Observation 3 above, the boundary of $CH(R_0)$ has only 2 line segments on the opposite side of L from P_0, and from Observation 1 the boundary of $CH(R_0)$ intersects at most 4 elements of $\mathcal{R}_1$. For all remaining regions $R'_1 \in \mathcal{R}_1$, either $R'_1 \subset R_0$ (so that for $x \in R'_1$, $f_1(x) > f_0(x)$) or $R'_1 \cap R_0 = \emptyset$ (so that again, for $x \in R'_1$, $f_1(x) > f_0(x)$).

For $\ell \in \{0,1\}$ let $P'_\ell = P_\ell \cup \{R_\ell \ : \ h(R_\ell) = \ell\}$. Note that $P'_\ell \subseteq h^{-1}(\ell)$ and has at most $3\lceil\sqrt{N}\rceil$ edges.

The portion of $t^{-1}(0)$ not in $P'_0 \cup P'_1$ is divided into $O(\sqrt{N})$ regions by the remaining edges of Q_0 and the two edges of Q_1 that intersect $t^{-1}(0)$. h is constant within each of these regions, which allows us to deduce that $h^{-1}(0)$ is indeed bounded by a set of line segments of size $O(\sqrt{N})$. By a similar argument, $h^{-1}(1)$ is bounded by $O(\sqrt{N})$ lines. ■

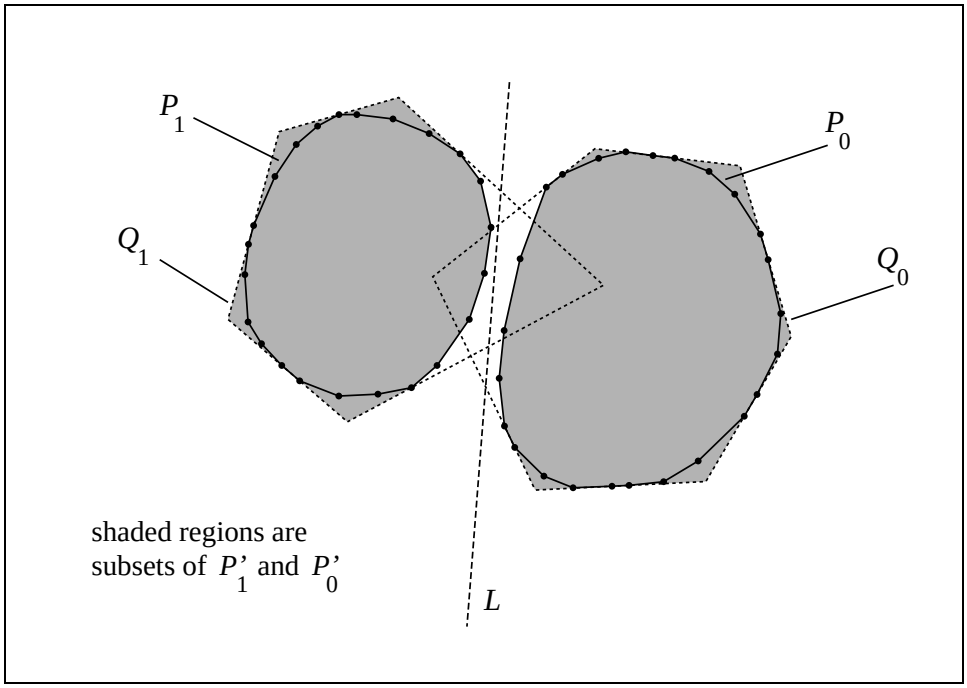


Figure 6: Illustration of algorithm for learning linear separators in two dimensions

3.5 Monomials over Attribute Vectors having a Product Distribution.

Recall that a monomial is a boolean function consisting of the conjunction of a set of literals (where a literal is either a boolean attribute or its negation). Despite the simplicity of this class of functions, we have not resolved its learnability under the restriction of Definition 1, even for monotone (negation-free) monomials. If $\mathcal{A}_0$ and $\mathcal{A}_1$ are allowed to be different algorithms ($\mathcal{A}$ is allowed to treat the positive data differently from the negative data), then the problem does have a simple solution (a property of any class of functions that is learnable from either positive examples only or else negative examples only). f_0 from $\mathcal{A}_0$ assigns a value of $\frac{1}{2}$ to all boolean vectors. $\mathcal{A}_1$ uses its data to find a PAC hypothesis, and assigns a value of 1 to examples satisfying that hypothesis, and 0 to other examples.

The following problem arises when $\mathcal{A}$ is oblivious to whether it is receiving the positive data. The distribution over the negative examples could in fact produce boolean vectors that satisfy some monomial f that differs from target monomial t, but if $D(f^{-1}(1) \cap t^{-1}(1)) > \varepsilon$ this may give excessive error.

In view of the importance of the concept class of monomials, we consider whether they are learnable given that the input distribution D belongs to a given class of probability distributions. This situation is intermediate between knowing D exactly (in which case by Theorem 4 the problem would be solved since monomials are learnable in the presence of uniform misclassification noise (Angluin and Laird, 1988)) and the distribution-independent setting.

1. *Draw a sample S_ℓ of size $N = \tilde{O}((n^3/\varepsilon)^2 \log(\frac{1}{\delta}))$.*
2. *For $x \in X$ let $\widehat{\psi}_\ell^j(x)$ denote the fraction of elements of S_ℓ whose j-th entry is equal to the j-th entry of x.*
3. *For $x \in X$, $f_\ell(x) = \Pi_{j=1}^n \widehat{\psi}_\ell^j(x)$.*

Figure 7: Algorithm for learning monomials

Theorem 11 *Monomials over the boolean domain are learnable via discriminant functions, provided that the input distribution D is known to be a product distribution.*

Comments. We use the algorithm of Figure 7 which simply fits a product distribution to its data and assigns a value to unlabeled vector x that is the estimated likelihood of x. The proof that it works heavily exploits the assumption that D is a product distribution, and does not appear to extend to larger class of distributions (for example, mixtures of product distributions (Cryan et al., 2001; Freund and Mansour, 1999)) or more general classes of boolean functions.

Proof We show that the algorithm given in Figure 7 constructs discriminant functions which, when combined to get h according to Definition 1, ensure that h is PAC.

For $x \sim D$, $x = x_1 x_2 \ldots x_n$ where x_j is a $0/1$ random variable which is independent of x_k for $k \neq j$. By a *relevant attribute* of t we mean any x_j whose value is fixed for all x that satisfy t. Let t_j denote

that value. Let $\mathcal{R}$ denote the set of relevant attributes and let I denote the remaining (irrelevant) attributes.

We say that an example $x' = x'_1 x'_2 \ldots x'_n$ with $t(x') = \ell$ is *ordinary* if for all $b \in \{0,1\}$ and $j \in \{1,\ldots,n\}$ such that $\Pr_{x\sim D_\ell}(x_j = b) > 1 - \frac{\varepsilon}{n}$, we have $x'_j = b$. (Thus, an ordinary example is one that does not have any "very unusual" attribute values in comparison with random examples having the same label. If there happen to be no bit positions that are very "reliable" for random examples with the same label, then the property becomes vacuous, or true for all bit strings.)

Note that for $\ell \in \{0,1\}$, $\Pr_{x\sim D_\ell}(x \text{ is ordinary}) \geq 1-\varepsilon$. Consequently, $\Pr_{x\sim D}(x \text{ is ordinary}) \geq 1-\varepsilon$. We will show that with probability at least $1-\delta$, all ordinary examples end up correctly labeled.

Let $\widehat{\Pr}_{x\in S_\ell}(x_j = b)$ denote the empirical probability that $x_j = b$, and we show that sample size N is large enough to ensure that with probability $1-\delta$, for $b \in \{0,1\}$, $j \in \{1,\ldots,n\}$,

$$|\widehat{\Pr}_{x\in S_\ell}(x_j = b) - \Pr_{x\sim D_\ell}(x_j = b)| \leq \frac{\varepsilon}{8n^3}. \tag{10}$$

Applying the same Hoeffding bound as in Theorem 4, it is sufficient that N should satisfy $2e^{-2N(\varepsilon/8n^3)^2} \leq \frac{\delta}{2n^2}$ which is satisfied by N as prescribed in Figure 7.

For $\ell \in \{0,1\}$, $x \in X$ let $\psi^j_\ell(x)$ denote the probability that a random vector with label ℓ agrees with x on the j-th entry. Note that if $t(x) = \ell$ then $\widehat{\psi}^j_\ell(x)$ (as defined in the algorithm of Figure 7) is an empirical estimate of $\psi^j_\ell(x)$.

Let $\psi_\ell(x) = \Pi_j \psi^j_\ell(x)$. Note that $f_\ell(x)$ is an estimate of $\psi_\ell(x)$ (in the sense that $f_\ell(x)$ converges to $\psi_\ell(x)$ as the sample size increases). We know from (10) that

$$\widehat{\psi}^j_\ell(x) \in \left[\psi^j_\ell(x) - \frac{\varepsilon}{8n^3}, \psi^j_\ell(x) + \frac{\varepsilon}{8n^3}\right].$$

If $\psi^j_\ell(x) > \varepsilon/n$, then

$$\widehat{\psi}^j_\ell(x)/\psi^j_\ell(x) \in \left[1 - \frac{1}{8n^2}, 1 + \frac{1}{8n^2}\right]. \tag{11}$$

Suppose x is ordinary and negative. Observe that $f_1(x) = 0$. (This is because x must have an attribute value that disagrees with all corresponding attribute values in the positive data.) Furthermore, Equation (11) holds for $\ell = 0$ and all j, implying that

$$\widehat{\psi}_0(x)/\psi_0(x) \in \left[1 - \frac{1}{4n}, 1 + \frac{1}{4n}\right]. \tag{12}$$

So with probability $1-\delta$, $f_0(x) > 0$, since $f_0(x) = 0$ would contradict Equation (12) taken with the observation that $\psi_0(x) > 0$. Hence with probability $1-\delta$, all ordinary negative examples are correctly labeled.

Suppose x is ordinary and positive. We will show that with probability $1-\delta$, $f_1(x)/f_0(x) > 1$ for all ordinary positive examples. Observe that for $j \in \mathcal{R}$, $\psi^j_1(x) = \widehat{\psi}^j_1(x) = 1$. (This is because all positive examples must agree on all the relevant attributes.) We have

$$\begin{aligned} f_1(x) &= \Pi_{j\in I}\widehat{\psi}^j_1(x) \\ f_0(x) &= \Pi_{j\in I}\widehat{\psi}^j_0(x)\Pi_{j\in\mathcal{R}}\widehat{\psi}^j_0(x). \end{aligned}$$

For $j \in I$, $\psi_1^j(x) = \psi_0^j(x)$ (for a product distribution D, the value of an irrelevant attribute is selected independently of the label class of a bit string). Hence Equation (11) applies for $\ell = 0$ or 1, $j \in I$.

$$\frac{f_1(x)}{f_0(x)} \geq \frac{(1-(1/8n^2))^n}{(1+(1/8n^2))^n}\Big(\frac{1}{\Pi_{j\in\mathcal{R}}\widehat{\psi}_0^j(x)}\Big).$$

There exists $j^* \in \mathcal{R}$ such that for a fraction at least $\frac{1}{n}$ of elements $x' \in S_0$, $x'_{j^*} \neq x_{j^*}$. (Each negative example must disagree with x on at least one relevant attribute.) Hence,

$$\begin{aligned}\widehat{\psi}_0^{j^*}(x) &\leq 1-(1/n)\\ \psi_0^{j^*}(x) &\leq 1-(1/n)+(\varepsilon/8n^2) < 1-(1/2n).\end{aligned}$$

Hence

$$\frac{f_1(x)}{f_0(x)} \geq \frac{(1-(1/8n^2))^n}{(1+(1/8n^2))^n}\Big(\frac{1}{1-(1/2n)}\Big) > 1,$$

as required. Hence with probability $1-\delta$, all ordinary positive examples are correctly labeled. ■

4. Conclusion and Open Problems

The algorithms we have given differ significantly from previous PAC algorithms, which usually work by minimizing the empirical error rate, and arguing that the way a hypothesis is constructed ensures that the true error is close to the empirical error. The constraint that we expressed in Definition 1 forces the positive data and the negative data to be processed independently—an algorithm does not have access to the empirical error.

This lack of access to the empirical error appears to be quite a severe constraint, one that might render certain learning problems intractable in the context of PAC learning. Indeed, we have so far failed to find an algorithm in this setting which learns monomials over the boolean domain, assuming no knowledge of the input distribution. We have also not obtained an algorithm for learning linear threshold functions in more than two dimensions. Despite those limitations, our positive results have distinguished learnability subject to this constraint from various other constraints on PAC learnability that have been studied in the past.

Clearly, the main open question raised by this paper is to elucidate the relationship between learnability via discriminant functions (Definition 1), and basic PAC learnability. Furthermore, if they are not equivalent, can they be distinguished using a well-known learning problem, such as monomials over the boolean domain?

We have a relatively good understanding of learnability subject to the slightly less severe constraint of Definition 2. Namely, it is intermediate between learnability with uniform misclassification noise, and standard PAC learnability. Furthermore, subject to the Noisy Parity Assumption (that it is hard to learn parity functions in the presence of random misclassification noise given the uniform distribution over input vectors) it is strictly a less severe constraint that learnability with uniform misclassification noise, since we have shown (Section 3.1) how to learn parity functions using the more severe constraint of Definition 1.

Acknowledgments

This work was supported by EPSRC Grant GR/R86188/01. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the author's views. A preliminary version of this paper was presented at the 2001 COLT conference.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, Dec 2000.

D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

M. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1992.

S. Ben-David and E. Dichterman. Learnability with restricted focus of attention guarantees noise-tolerance. In *5th International Workshop on Algorithmic Learning Theory*, pages 248–259, 1994.

S. Ben-David and E. Dichterman. Learning with restricted focus of attention. *Journal of Computer and System Sciences*, 56(3):277–298, 1998.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

A. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. *SIAM Journal on Computing*, 23(5):990–1000, 1994.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

M. Cryan, L. A. Goldberg, and P. W. Goldberg. Evolutionary trees can be learned in polynomial time in the two-state general markov model. *SIAM Journal on Computing*, 31(2):375–397, 2001.

S. Dasgupta. Learning mixtures of gaussians. In *40th IEEE Symposium on Foundations of Computer Science*, pages 634–644, 1999.

F. Denis. Pac learning from positive statistical queries. In *Algorithmic Learning Theory (ALT), 9th International Conference*, volume 1501 of *LNAI*, pages 112–126. Springer, 1998.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

Y. Freund and Y. Mansour. Estimating a mixture of two product distributions. In *Proceedings of the 12th Workshop on Computational Learning Theory (COLT)*, pages 53–62. Morgan Kaufmann, 1999.

A. Frieze, M. R. Jerrum, and R. Kannan. Learning linear transformations. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 359–368, 1996.

V. Guruswami and A. Sahai. Multiclass learning, boosting, and error-correcting codes. In *Proceedings of the 12th Workshop on Computational Learning Theory (COLT)*, pages 145–155, 1999.

D. Haussler, M. J. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

D. Helmbold, R. Sloan, and M. K. Warmuth. Learning integer lattices. *SIAM Journal on Computing*, 21(2):240–266, 1992.

M. J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6): 983–1006, 1998.

M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proceedings of the 26th Annual ACM Symposium on the Theory of Computing*, pages 273–282, 1994.

M. J. Kearns and R. E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497, 1994.

F. Letouzey, F. Denis, and R. Gilleron. Learning from positive and unlabeled examples. In *Proceedings of the 11th International Conference on Algorithmic Learning Theory*, pages 71–85, 2000.

N. Palmer and P. W. Goldberg. Pac classification based on pac estimates of label class distributions. Technical Report 411, University of Warwick, Dept. of Computer Science, 2005.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Proceedings of 12th NIPS conference*, 2000.

R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, second edition, 2000.

Journal of Machine Learning Research 7 (2006) 307–342 Submitted 7/05; Revised 1/06; Published 2/06

Kernels on Prolog Proof Trees: Statistical Learning in the ILP Setting*

Andrea Passerini PASSERINI@DSI.UNIFI.IT
Paolo Frasconi P-F@DSI.UNIFI.IT
Dipartimento di Sistemi e Informatica
Università degli Studi di Firenze
Via di Santa Marta 3
I-50139 Firenze, Italy

Luc De Raedt DERAEDT@INFORMATIK.UNI-FREIBURG.DE
Institute for Computer Science
Albert-Ludwigs Universität Freiburg
Georges-Koehler-Allee 79
D-79110 Freiburg, Germany

Editors: Roland Olsson and Ute Schmid

Abstract

We develop kernels for measuring the similarity between relational instances using background knowledge expressed in first-order logic. The method allows us to bridge the gap between traditional inductive logic programming (ILP) representations and statistical approaches to supervised learning. Logic programs are first used to generate proofs of given visitor programs that use predicates declared in the available background knowledge. A kernel is then defined over pairs of proof trees. The method can be used for supervised learning tasks and is suitable for classification as well as regression. We report positive empirical results on Bongard-like and *M*-of-*N* problems that are difficult or impossible to solve with traditional ILP techniques, as well as on real bioinformatics and chemoinformatics data sets.

Keywords: kernel methods, inductive logic programming, Prolog, learning from program traces

1. Introduction

Within the fields of automated program synthesis, inductive logic programming (ILP) and machine learning, several approaches exist that learn from example-traces. An example-trace is a sequence of steps taken by a program on a particular example input. For instance, Biermann and Krishnaswamy (1976) have sketched how to induce Turing machines from example-traces (in this case sequences of primitive actions and assertions). Mitchell et al. (1983) have developed the LEX system that learned how to solve symbolic integration problems by analyzing traces (or search trees) for particular example problems. Ehud Shapiro's Model Inference System (1983) inductively infers logic programs by reconstructing the proof-trees and traces corresponding to particular facts. Zelle and Mooney (1993) show how to speed-up the execution of logic programs by analyzing example-traces of the underlying logic program. Finally, De Raedt et al. (2005) proposed a method for learning stochastic

*. An early version of this paper was presented at the ICML '05 Workshop on Approaches and Applications of Inductive Programming (AAIP).

logic programs using proof trees as training examples. The diversity of these applications as well as the difficulty of the learning tasks considered illustrate the power of learning from example-traces for a wide range of applications.

In this paper, we generalize the idea of learning from example-traces. Rather than explicitly learning a target program from positive and negative example-traces, we assume that a particular—so-called *visitor* program—is given and that our task consists of learning from the associated traces. The advantage is that in principle any programming language can be used to model the visitor program and that any machine learning system able to use traces as an intermediate representation can be employed. In particular, this allows us to combine two frequently employed frameworks within the field of machine learning: ILP and kernel methods. Logic programs will be used to generate traces corresponding to specific examples and kernels to quantify the similarity between traces. This combination yields an appealing and expressive framework for tackling complex learning tasks involving structured data in a natural manner. We call *trace kernels* the resulting broad family of kernel functions obtainable as a result of this combination. The visitor program is a set of clauses that can be seen as the *interface* between the available background knowledge and the kernel itself. Intuitively, the visitor program plays a role that is similar to that of declarative bias in inductive logic programming systems (Nédellec et al., 1996) (see also Section 6).

Kernels methods have been widely used in many relational learning contexts. Starting from the seminal work of Haussler (1999) (briefly reviewed in Section 4.1) several researchers have proposed kernels over discrete data structures such as sequences (Lodhi et al., 2002; Jaakkola and Haussler, 1999; Leslie et al., 2002; Cortes et al., 2004), trees (Collins and Duffy, 2002; Viswanathan and Smola, 2003), annotated graphs (Gärtner, 2003; Schölkopf and Warmuth, 2003; Kashima et al., 2003; Mahé et al., 2004; Horváth et al., 2004; Menchetti et al., 2005), and complex individuals defined using higher order logic abstractions (Gärtner et al., 2004). Constructing kernels over structured data types, however, is not the only aim of the proposed framework. In many symbolic approaches to learning, logic programs allow us to define background knowledge in a natural way. Similarly, in the case of kernel methods, the notion of similarity between two instances expressed by the kernel function is the main tool for exploiting the available domain knowledge. It seems therefore natural to seek a link between logic programs and kernels, also as a means for embedding knowledge into statistical learning algorithms in a *principled* and *flexible* way. This aspect is one of the main contributions of this paper as few alternatives exist to achieve this goal. Propositionalization, for example, transforms a relational problem into one that can be solved by an attribute-value learner by mapping data structures into a finite set of features (Kramer et al., 2000). Although it is known that in many practical applications propositionalization works well, its flexibility is generally limited. A remarkable exception is the method proposed by Cumby and Roth (2002) that uses description logic to specify features and that has been subsequently extended to specify kernels (Cumby and Roth, 2003). Muggleton et al. (2005) have proposed an approach where the feature space is spanned by a set of first order clauses induced by an ILP learning algorithm. Declarative kernels (Frasconi et al., 2004) are another possible solution towards the above aim. A declarative kernel is essentially based on a background-knowledge dependent relation that allows us to extract *parts* from instances. Instances are reduced in this way to "bags-of-parts" and a combination of sub-kernels between parts is subsequently used to obtain the kernel between instances.

The guiding philosophy of trace kernels is very different from all the above approaches. Intuitively, rather than comparing two given instances directly, these kernels compare the execution traces of a program that takes instances as its input. Similar instances should produce similar traces

when probed with programs that express background knowledge and examine characteristics they have in common. These characteristics can be more general than parts. Hence, trace kernels can be introduced with the aim of achieving a greater generality and flexibility with respect to various decomposition kernels (including declarative kernels). In particular, *any* program to be executed on data can be exploited within the present framework to form a valid kernel function, provided one can give a suitable definition of the *visitor* program to specify how to obtain relevant traces and proofs to compare examples. Although in this paper we only study trace kernels for logic programs, similar ideas could be used in the context of different programming paradigms and in conjunction with alternative models of computation such as finite state automata or Turing machines.

In this paper, we focus on a specific learning framework for Prolog programs. The execution trace of a Prolog program consists of a set of search trees associated with a given goal. To avoid feature explosion due to failed paths, which are typically much more numerous and less informative than successful ones, we resort to a reduced representation of traces based on proof trees (Russell and Norvig, 2002) that only maintain successful search paths. Proof trees can be conveniently represented as Prolog ground terms. Thus, in this case, kernels over traces reduce to Prolog ground terms kernels (PGTKs) (Passerini and Frasconi, 2005). These kernels (which are reviewed in Section 4.3) can be seen as a specialization to Prolog of the kernels between higher order logic individuals earlier introduced by Gärtner et al. (2004). Because of the special nature of terms in the present context, we also suggest some proper choices for comparing logical terms that represent proofs. One central advantage of the proposed method, as compared to inductive logic programming, is that it naturally applies to both classification and regression tasks.

The remainder of this paper is organized as follows. In Section 2 we review the traditional frameworks of statistical learning and ILP. In Section 3 we develop a new framework for statistical learning in the ILP setting and introduce visitor programs and their traces. In Section 4 we derive kernel functions over program traces represented as Prolog proof trees. In Section 5 we report an empirical evaluation of the methodology on some classic ILP benchmarks including Bongard problems, *M*-of-*N* problems on sequences, and real world problems in bioinformatics and chemoinformatics. Section 6 contains a discussion on the relations between our approach and traditional ILP methods, as well as explanation based learning (Mitchell et al., 1986). Finally, conclusions are drawn in Section 7.

2. Notation and Background

In this section, we briefly review some concepts related to supervised learning (from both the statistical and the ILP perspective) that will be used for defining the framework of learning from proof trees presented in the paper.

2.1 Statistical Learning and Kernels

In the usual statistical learning framework (see, e.g., Cucker and Smale, 2002, for a thorough mathematical foundation) a supervised learning algorithm is given a training set of input-output pairs $\mathcal{D} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The set $\mathcal{X}$ is called the input (or instance) space and can be any set. The set $\mathcal{Y}$ is called the output (or target) space; in the case of binary classification $\mathcal{Y} = \{-1, 1\}$ while the case of regression $\mathcal{Y}$ is the set of real numbers. A fixed (but unknown) probability distribution on $\mathcal{X} \times \mathcal{Y}$ links input objects to their output target values. The

learning algorithm outputs a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that approximates the probabilistic relation between inputs and outputs. The class of functions that is searched is called the *hypothesis space*.

A (Mercer) kernel is a positive semi-definite symmetric function[1] $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ that generalizes the notion of inner product to arbitrary domains (see, e.g., Shawe-Taylor and Cristianini, 2004, for details). When using kernel methods in supervised learning, the hypothesis space, denoted $\mathcal{F}_K$, is the so-called reproducing kernel Hilbert space (RKHS) associated with K. Learning consists of solving the following Tikhonov regularized problem:

$$f = \arg\min_{h \in \mathcal{F}_K} C \sum_{i=1}^{m} V(y_i, h(x_i)) + \|h\|_K \tag{1}$$

where $V(y, h(x))$ is a positive function measuring the loss incurred in predicting $h(x)$ when the target is y, C is a positive regularization constant, and $\|\cdot\|_K$ is the norm in the RKHS. Popular algorithms in this framework include support vector machines (SVM) (Cortes and Vapnik, 1995) and kernel ridge regression (Poggio and Smale, 2003; Shawe-Taylor and Cristianini, 2004). The representer theorem (Kimeldorf and Wahba, 1970) shows that the solution to the above problem can be expressed as a linear combination of the kernel between individual training examples x_i and x as follows:

$$f(x) = \sum_{i=1}^{m} c_i K(x, x_i). \tag{2}$$

The above form also encompasses the solution found by other algorithms such as the kernel perceptron (Freund and Schapire, 1999).

2.2 Inductive Logic Programming

Within the field of inductive logic programming, the standard framework is that of learning from entailment. In this setting, the learner is given a set of positive and negative examples $\mathcal{D}^+$ and $\mathcal{D}^-$, respectively (in the form of ground facts), and a background theory $\mathcal{B}$ (as a set of definite clauses) and has to induce a hypothesis $\mathcal{H}$ (also a set of definite clauses) such that $\mathcal{B} \cup \mathcal{H}$ covers all positive examples and none of the negative ones. More formally, $\forall p(x) \in \mathcal{D}^+ : \mathcal{B} \cup \mathcal{H} \models p(x)$ and $\forall p(x) \in \mathcal{D}^- : \mathcal{B} \cup \mathcal{H} \not\models p(x)$. In this paper, as in the work by Lloyd (2003), we shall use examples that are individuals, i.e., first-order logic objects or identifiers. This means that we shall effectively refer to the examples by their identifier x rather than use $p(x)$. The traditional definition of inductive logic programming does not explicitly—as is the case of regularized empirical risk minimization—account for noisy data and the possibility that a complete and consistent hypothesis might not exist. Even though various noise handling techniques exist in inductive logic programming, they are not as principled as those offered by statistical learning theory.

Example 1 *As an illustration of the above concepts, consider the famous mutagenicity benchmark by Srinivasan et al. (1996). There the examples are of the form* `mutagenic(id)` *where* `id` *is a unique identifier of the molecule and the background knowledge contains information about the atoms, bonds and functional groups in the molecule. A hypothesis in this case could be*

```
mutagenic(ID)  ← nitro(ID,R),lumo(ID,L), L<-1.5.
```

1. A symmetric function $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called a *positive semi-definite kernel* iff $\forall m \in \mathbb{N}, \forall x_1, \ldots, x_m \in \mathcal{X}, \forall a_1, \ldots, a_m \in \mathbb{R}$, $\sum_{i,j=1}^{m} a_i a_j K(x_i, x_j) \geq 0$.

```
mutagenic(d26).

lumo(d26, -2.072).
logp(d26, 2.17).
atm(d26,d26_1,c,22,-0.093).
atm(d26,d26_2,c,22,-0.093).
atm(d26,d26_3,c,22,-0.093).
atm(d26,d26_4,c,22,-0.093).
atm(d26,d26_5,c,22,-0.093).
atm(d26,d26_6,c,22,-0.093).
atm(d26,d26_7,h,3,0.167).
atm(d26,d26_8,h,3,0.167).
atm(d26,d26_9,h,3,0.167).
atm(d26,d26_10,cl,93,-0.163).
atm(d26,d26_11,n,38,0.836).
atm(d26,d26_12,n,38,0.836).
atm(d26,d26_13,o,40,-0.363).
atm(d26,d26_14,o,40,-0.363).
atm(d26,d26_15,o,40,-0.363).
atm(d26,d26_16,o,40,-0.363).
bond(d26,d26_1,d26_2,7).
bond(d26,d26_2,d26_3,7).
bond(d26,d26_3,d26_4,7).
bond(d26,d26_4,d26_5,7).
bond(d26,d26_5,d26_6,7).
bond(d26,d26_6,d26_1,7).
bond(d26,d26_1,d26_7,1).
bond(d26,d26_3,d26_8,1).
bond(d26,d26_6,d26_9,1).
bond(d26,d26_10,d26_5,1).
bond(d26,d26_4,d26_11,1).
bond(d26,d26_2,d26_12,1).
bond(d26,d26_13,d26_11,2).
bond(d26,d26_11,d26_14,2).
bond(d26,d26_15,d26_12,2).
bond(d26,d26_12,d26_16,2).
```

```
nitro(X,[Atom0,Atom1,Atom2,Atom3]) :-
    atm(X,Atom1,n,38,_),
    bondd(X,Atom0,Atom1,1),
    bondd(X,Atom1,Atom2,2),
    atm(X,Atom2,o,40,_),
    bondd(X,Atom1,Atom3,2),
    Atom3 @> Atom2,
    atm(X,Atom3,o,40,_).

bondd(X,Atom1,Atom2,Type) :-
    bond(X,Atom1,Atom2,Type).
bondd(X,Atom1,Atom2,Type) :-
    bond(X,Atom2,Atom1,Type).
```

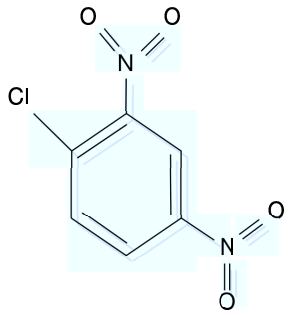


Figure 1: Example from the mutagenesis domain. Top: *extensional* representation of an instance (a molecule). Left: sample fragment of *intensional* background theory. Right: chemical structure of the molecule.

It entails (covers) the molecule listed in Figure 1. It will be convenient to distinguish extensional *predicates, such as* `atm`, `logp`, `lumo` *and* `bond`, *which specify information about* specific *examples, from the* intensional *ones, such as* `bbond` *and* `nitro`*, which specify general properties about all examples.*

Regression can be introduced in ILP in different ways. For example in the First-Order Regression System (Karalič and Bratko, 1997) some arguments of the target predicate (called continuous attributes) are real-valued. For instance, in our example one could use examples of the form `mutagenic(d26, -2.072, 2.17, 6.3)` where the arguments would be the lumo and logp values as well as the target activity. FORS then learns from "positive" examples only, covering subsets of examples on which linear regression between the continuous arguments is solved in a numerical way. An interesting alternative is Structural Regression Trees, a method based on divide-and-conquer, similar to regression trees (Kramer, 1996).

3. A Framework for Statistical Learning in the ILP Setting

In this section we introduce the logical framework for defining program traces and, in particular, the concepts of visitor programs and proof trees.

3.1 General Assumptions

The methods described in this paper are based on a framework that combines some of the advantages of the statistical and the ILP settings, in particular noise robustness and the possibility of describing background knowledge in a flexible declarative language. First, we assume that the instance space $\mathcal{X}$ is a set of first-order logic objects (i.e., individuals in the universe of discourse), each having a unique identifier x. As in the ILP setting, we assume that a background theory $\mathcal{B}$ is available in the form of a set of definite clauses. This background theory is divided into *intensional* predicates, which are relevant to all examples, and *extensional* ones, which specify facts about specific examples. As in the statistical setting, we assume that a fixed and unknown distribution is defined on $\mathcal{X} \times \mathcal{Y}$ and that training data $\mathcal{D}$ consist of input-output pairs (x_i, y_i) (for classification or regression). Rather than having to find a set of clauses $\mathcal{H}$, the learning algorithm outputs a function f that maps instances into their targets and whose general form is given by Equation (2). In this sense, our setting is close to statistical learning and predictions on new instances will be essentially opaque. However, we make the fundamental assumption that f also depends on the available background theory via the kernel function.

3.2 Visitors

A second key difference with respect to the traditional ILP setting is that in addition to data $\mathcal{D}$ and background knowledge $\mathcal{B}$, the learner is given an additional set of clauses forming the so-called *visitor* program. Clauses in this program should be designed to "inspect" examples using other predicates declared in $\mathcal{B}$. In facts, as detailed in Section 4, the kernel function to be plugged in Equation (2) will be defined by means of the trace of this program. To this aim, we are not only interested in determining whether certain clauses succeed or fail on a particular example. In our approach, the execution traces of the visitor programs are recorded and compared, on the rationale that examples having similar traces should be mapped to similar representations in the feature space associated with the kernel. The purpose of visitors is thus to construct useful features during their execution. This is a major difference with respect to other approaches in which features are explicitly constructed by computing the truth value for predicates (Muggleton et al., 2005).

Definition 1 (Visitor programs) *A* visitor *program for a background theory $\mathcal{B}$ and domain $\mathcal{X}$ is a set $\mathcal{V}$ of definite clauses that contains at least one special clause (called a* visitor*) of the form $V \leftarrow B_1, \ldots, B_N$ and such that*

- *V is a predicate of arity 1*
- *for each $j = 1, \ldots, N$, B_j is declared in $\mathcal{B} \cup \mathcal{V}$;*

Intuitively, if `visit/1` is a visitor in $\mathcal{V}$, by answering the query `visit(x)?` we explore the features of the instance whose constant identifier `x` is passed to the visitor. Having multiple visitors in the program $\mathcal{V}$ allows us to explore different aspects of the examples and include multiple sources of information.

Some examples of visitor programs are introduced in the remainder of this section and when presenting empirical results in Section 5.

3.3 Traces and Proof Trees

A visitor program trace for a given domain instance is obtained by recording proofs of visitor goals called on that instance. There are alternative options for choosing the kind of proof to be employed. Therefore in order to give a precise definition of traces, we now need to make a specific design choice. In this paper, we are committed to Prolog-based representations. Hence, a natural option would be the use of SLD-trees, whose paths correspond to execution sequences of the Prolog interpreter. A drawback of this choice is that an SLD-tree is a very complex and rather unstructured representation and also contains information about failed paths, potentially leading to an explosion of redundant and irrelevant features for the purpose of learning. For these reasons we prefer to resort to proof trees (Russell and Norvig, 2002), defined as follows:

Definition 2 (Proof tree) [2] *Let $\mathcal{P}$ be a program and G a goal. If $\mathcal{P} \not\models G$ then the* proof tree *for G is empty. Otherwise, it is a tree t recursively defined as follows:*

- *if there is a fact f in $\mathcal{P}$ and a substitution θ such that $G\theta = f\theta$, then $G\theta$ is a leaf of t.*
- *otherwise there must be a clause $H \leftarrow B_1, ..., B_n \in \mathcal{P}$ and a substitution θ' such that $H\theta' = G\theta'$ and $\mathcal{P} \models B_j\theta' \; \forall j$, $G\theta'$ is the root of t and there is a subtree of t for each $B_j\theta'$ which is a proof tree for $B_j\theta'$.*

The kernels used in this paper work on ground proof trees. In general, however, proof trees or SLD-trees need not be ground. If they are not, they can however always be made ground by skolemization, i.e., by substituting all variables by different constants not yet appearing in the program and goal. The skolemized proof will then still logically follow from the program. Alternatively, one could impose the requirement that all clauses are range-restricted, a requirement that is often imposed in the logic programming community. Range-restrictedness requires that all variables that appear in the head of a clause also appear in its body. It is a sufficient requirement for guaranteeing that all proofs will be ground. Finally, ground proofs can be also obtained by making specific assumptions about the mode of head variables not occurring in the body, so that these variables will be instantiated in proving the goal. All the visitor programs presented in our empirical evaluation (see Section 5) yield ground proofs thanks to such assumptions.

Example 2 *For the sake of illustration, consider again the mutagenesis domain. Consider the atom bond representation of the simple molecule in Figure 1. By looking at the molecule as a graph where atoms are nodes and bonds are edges, we can introduce the common notions of* path *and* cycle*:*

```
1 : cycle(X,A):-
      path(X,A,B,[A]),
      bond(X,B,A,_).
```

```
2 : path(X,A,B,M):-
      atm(X,A,_,_,_),
      bond(X,A,B,_),
      atm(X,B,_,_,_),
      not(member(B,M)).
```

```
3 : path(X,A,B,M):-
      atm(X,A,_,_,_),
      bond(X,A,C,_),
      not(member(C,M)),
      path(X,C,B,[C|M]).
```

The following simple visitor may be used to inspect cycles in the molecule:

```
4 : visit(X):
      cycle(X,A).
```

2. Such trees are sometimes also named *and-trees*.

Note that we numbered each clause in $\mathcal{V}$ and the intensional part of the background theory $\mathcal{B}$ (but not in the extensional part[3]) with a unique identifier. This will allow us to take into account information about the clauses that are used in a proof. The corresponding proof tree for this example is shown in Figure 2.

In general, a goal can be satisfied in more than one way. Therefore, each query generates a (possibly empty) set of proof trees. Since multiple visitors may be available, the trace of an instance is a tuple of sets of proof trees, as formalized in the following definition:

Definition 3 (Trace) *Let N be the number of visitors in $\mathcal{V}$ and for each $l = 1,\ldots,N$ let $T_{lj,x}$ denote the proof tree that represents the j-th proof of the goal $V_l(x)$, i.e., a proof that $\mathcal{B} \cup \mathcal{V} \models V_l(x)$. Let*

$$T_{l,x} = \{T_{l1,x},\ldots,T_{ls_{l,x},x}\} \tag{3}$$

where $s_{l,x} \geq 0$ is the number of alternative proofs of goal $V_l(x)$. The trace of an instance x is the tuple

$$T_x = [T_{1,x},\ldots,T_{N,x}]. \tag{4}$$

3.4 Pruning Proof Trees

In many situations, the proof tree for a given goal will be unnecessary complex in that it may contain several uninteresting subtrees. In these cases, we will often work with *pruned* proof trees, which are trees where subtrees rooted at specific predicates (declared as `leaf` predicates by the user) are turned into leafs. This will reduce the complexity of the feature space associated with the kernel by selectively ignoring subproofs. For instance, consider again the mutagenesis domain described in Srinivasan et al. (1996) where a theory of rings and functional groups is included as background knowledge (see Figure 1). In this domain, it may be useful to define visitors that explore groups such as benzene rings:

```
atoms(X,[]).
atoms(X,[H|T]):-
    atm(X,H,_,_,_),
    atoms(X,T).
```

```
visit_benzene(X):-
    benzene(X,Atoms),
    atoms(X,Atoms).
```

If we believe that the presence of the ring and the nature of the involved atoms represent a sufficient set of features, we may want to ignore details about the proof of the predicate `benzene` by pruning the corresponding proof subtree. This can be accomplished by including the following fact in the visitor program:

```
leaf(benzene(_,_)).
```

3.5 Bridging the Gap

We are finally able to give a complete formalization of our framework for learning from example-traces. The learner is given a data set $\mathcal{D} = \{(x_1,y_1),\ldots,(x_m,y_m)\}$, background knowledge $\mathcal{B}$, and visitor program $\mathcal{V}$. For each instance x_i, a trace T_{x_i} is obtained by running the visitor program

3. The numbers in the extensional part would change from example to example and hence, would not carry any useful information.

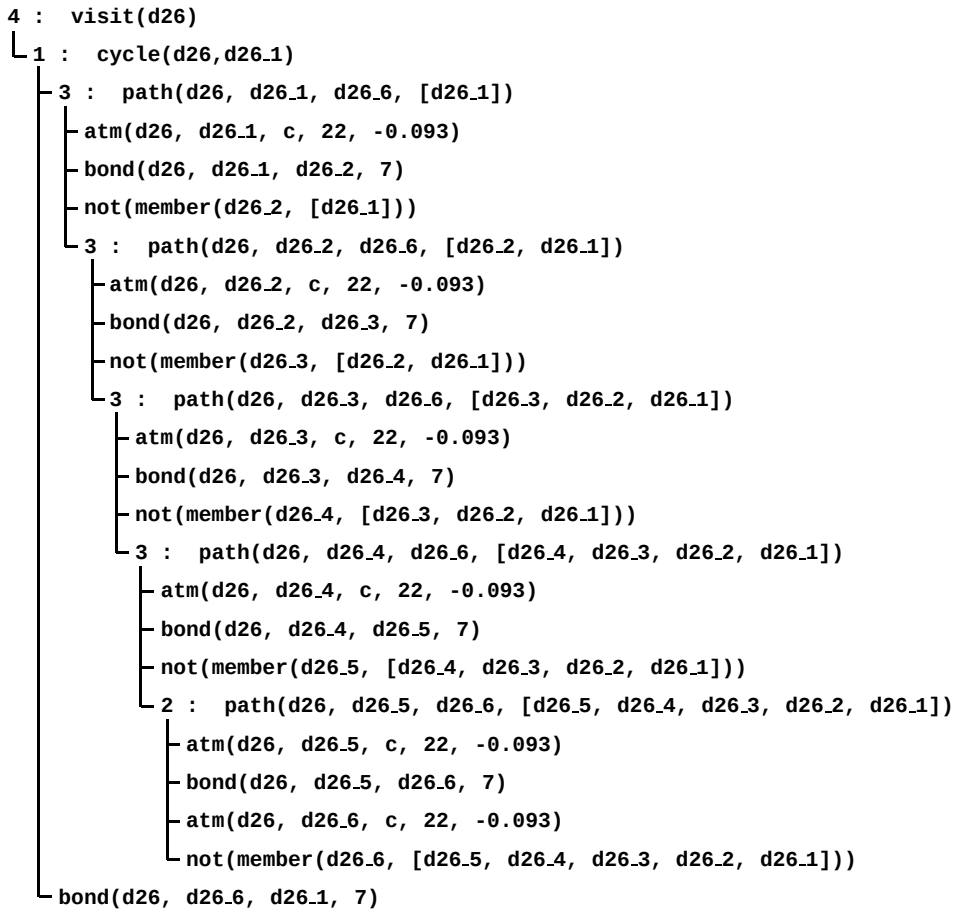


Figure 2: Proof tree resulting from the goal `visit(d26)` in the mutagenesis example.

according to Definition 3. A kernel machine (e.g., an SVM) is then trained to form the function $f : \mathcal{X} \mapsto \mathcal{Y}$ defined as

$$f(x) = \sum_{i=1}^{m} c_i K(T_{x_i}, T_x).$$

The only missing ingredient is the kernel function K for comparing two visitor traces. The definition of this function is detailed in the next section.

4. Kernels over Visitor Traces

In this section, we derive kernel functions for comparing traces of visitor programs. We begin by reviewing some preliminary concepts about convolution kernels (Haussler, 1999), a very general family of kernels on discrete structures that will be used in the rest of the paper to define kernels over the logical structures of interest.

4.1 Kernels for Discrete Structures

For the purposes of this subsection, let $\mathcal{X}$ be a set of composite structures and for $x \in \mathcal{X}$ let $x_1, \ldots, x_D$ denote the "parts" of x, with $x_d \in \mathcal{X}_d$ for all $i \in [1, D]$. This decomposition can be formally represented by a relation R on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_D \times \mathcal{X}$ such that $R(x_1, \ldots, x_D, x)$ is true iff $x_1, \ldots, x_D$ are the parts of x. We also write $(x_1, \ldots, x_D) = R^{-1}(x)$ if $R(x_1, \ldots, x_D, x)$ is true. Note that the relation R used in this definition is very general and does not necessarily satisfy an axiomatic theory for parts and wholes such as those studied in knowledge representation (Varzi, 1996). For example if $\mathcal{X}_1 = \cdots = \mathcal{X}_D = \mathcal{X}$ are sets containing all finite strings over a finite alphabet, we can define a relation $R(x_1, \ldots, x_D, x)$ which is true iff $x = x_1 \circ \cdots \circ x_D$, with $\circ$ denoting concatenation of strings. Note that in this example x can be decomposed in multiple ways. We say that the relation R is finite if the number of such decompositions is finite. Given a set of kernels $K_d : \mathcal{X}_d \times \mathcal{X}_d \to \mathbb{R}$, one for each of the parts of x, the *R-convolution* kernel is defined as

$$K_{R,\otimes}(x, z) = \sum_{(x_1,\ldots,x_D) \in R^{-1}(x)} \sum_{(z_1,\ldots,z_D) \in R^{-1}(z)} \prod_{d=1}^{D} K_d(x_d, z_d) \tag{5}$$

where the sums run over all the possible decompositions of x and z. Similarly, one could use direct sum obtaining

$$K_{R,\oplus}(x, z) = \sum_{(x_1,\ldots,x_D) \in R^{-1}(x)} \sum_{(z_1,\ldots,z_D) \in R^{-1}(z)} \sum_{d=1}^{D} K_d(x_d, z_d). \tag{6}$$

For finite relations R, these functions can be shown to be valid kernels:

Theorem 4 (Haussler 1999) *For any finite R on a space $\mathcal{X}$, the functions $K_{R,\otimes} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ (defined by Equation (5)) and $K_{R,\oplus} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ (defined by Equation (6)) are positive semi-definite kernels on $\mathcal{X} \times \mathcal{X}$.*

Proof: Follows from closure properties of tensor product and direct sum. See Haussler (1999) for details.

The *set kernel* (Shawe-Taylor and Cristianini, 2004) is a special case of convolution kernel that will prove useful in defining kernels between visitor traces. Suppose instances are sets and let us

define the part-of relation as the usual set-membership. The kernel over sets K_{set} is then obtained from kernels between set members K_{member} as follows:

$$K_{set}(x,z) = \sum_{\xi \in x} \sum_{\zeta \in z} K_{member}(\xi, \zeta). \tag{7}$$

In order to reduce the dependence on the dimension of the objects, kernels over discrete structures are often normalized. A common choice is that of using normalization in feature space, i.e., given a convolution kernel K_R:

$$K_{norm}(x,z) = \frac{K_R(x,z)}{\sqrt{K_R(x,x)}\sqrt{K_R(z,z)}}. \tag{8}$$

In the case of set kernels, an alternative is that of dividing by the cardinalities of the two sets, thus computing the mean value between pairwise comparisons:[4]

$$K_{mean}(x,z) = \frac{K_{set}(x,z)}{|x||z|}. \tag{9}$$

Richer families of kernels on data structures can be formed by applying composition to the feature mapping induced by a convolution kernel. For example, a convolution kernel K_R can be combined with a Gaussian kernel as follows:

$$K(x,z) = \exp\left(-\gamma\Big(K_R(x,x) - 2K_R(x,z) + K_R(z,z)\Big)\right). \tag{10}$$

4.2 Kernels over Visitor Programs

Going back to the framework defined in Section 3, let $\mathcal{X}$ be a set of first-order logic objects and for $x,z \in \mathcal{X}$ consider the program traces T_x and T_z defined by Equations (3) and (4). In order to define the kernel over program traces we follow a top-down approach. We begin by decomposing traces into parts associated with different visitors (i.e., the elements of the tuple in Equation (4)) and applying a decomposition kernel based on direct sum as defined by Equation (6):

$$K(T_x, T_z) = \sum_{l=1}^{N} K_l(T_{l,x}, T_{l,z}). \tag{11}$$

Note that there is a unique decomposition of T_x and T_y, that is we just compare proofs of the same visitor. According to Definition 3 for each $l = 1,\ldots,N$, the arguments to K_l are sets of proof trees. Hence, using the set kernel of Equation (7) we further obtain:

$$K_l(T_{l,x}, T_{l,z}) = \sum_{p=1}^{s_{l,x}} \sum_{q=1}^{s_{l,z}} K_{tree}(T_{lp,x}, T_{lq,z}). \tag{12}$$

In this way, we have shown that the problem boils down to defining a kernel K_{tree} over individual proof trees. This will be detailed in the remainder of this section. Note that we can define different kernels for proof trees originating from different visitors.

4. Note that normalizations such as those of Equations (8) and (9) can give indefinite results iff one of the two arguments (say x) is the null vector of the feature space associated to the original kernel (i.e., K_R or K_{set}). In such a case, we will define $K_{norm}(x,z) = K_{mean}(x,z) = 0\ \forall z \in \mathcal{X}, z \neq x$.

At the highest level of kernel between visitor programs (Equation (11)), it is advisable to employ a feature space normalization using Equation (8). In some cases it may also be useful to normalize lower-level kernels, in order to rebalance contributions of individual parts. In particular, the mean normalization of Equation (9) can be applied to the kernel over individual visitors (Equation (12)) and it is also possible to normalize kernels between individual proof trees, in order to reduce the influence of the proof size.

4.3 Representing Proof Trees as Prolog Ground Terms

Proof trees are discrete data structures and, in principle, existing kernels on trees could be applied (e.g. Collins and Duffy, 2002; Viswanathan and Smola, 2003). However, we can gain more expressiveness by representing individual proof trees as typed Prolog ground terms. In so doing we can exploit type information on constants and functors so that different sub-kernels can be applied to different object types. In addition, while traditional tree kernels would typically compare *all* pairs of subtrees between two proofs, the kernel on ground terms presented below results in a more selective approach that compares certain parts of two proofs only when reached by following similar inference steps (a distinction that would be difficult to implement with traditional tree kernels).
We will use the following procedure to represent a proof tree as a Prolog ground term:

- Base step: if a node contains a fact, this is already a ground term.
- Induction: if a node contains a clause, then let n be the number of arguments in the head and m the number of atoms in the body (corresponding to the m children of the node). A ground compound term t having $n+1$ arguments is then formed as follows:
  - the functor name of t is the functor name of the head of the clause;
  - the first n arguments of t are the arguments of the clause head;
  - the last argument of t is a compound term whose functor name is a Prolog constant obtained from the clause number,[5] and whose m arguments are the ground term representations of the m children of the node.

Example 3 *Consider the proof tree of Figure 2 in the mutagenesis domain. The transformation outlined above yields the following representation as a Prolog ground term:*

```
visit(d26,
      cbody4(cycle(d26,
                   d26_1,
                   cbody1(path(d26,
                               d26_1,
                               d26_6,
                               [d26_1],
                               cbody3(...)),
                          bond(d26,d26_6,d26_1,7))))).
```

where we skipped the representation of the children of path *for the sake of readability.*

We are now able to employ kernels on Prolog ground terms as defined in Passerini and Frasconi (2005) to compute kernels over individual proof trees.

5. Since numbers cannot be used as functor names, this constant can be simply obtained by prefixing the clause number by 'cbody'.

4.4 Kernels on Prolog Ground Terms

We begin with kernels on untyped terms. Let $\mathcal{C}$ be a set of constants and $\mathcal{F}$ a set of functors, and denote by $\mathcal{U}$ the corresponding Herbrand universe (the set of all ground terms that can be formed from constants in $\mathcal{C}$ and functors in $\mathcal{F}$). Let $f^{/n} \in \mathcal{F}$ denote a functor having name f and arity n.

Definition 5 (Sum kernels on untyped terms) *The kernel between two terms t and s is a function $K : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}$ defined inductively as follows:*

- *if $s \in \mathcal{C}$ and $t \in \mathcal{C}$ then*
$$K(s,t) = \kappa(s,t) \tag{13}$$
where $\kappa : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}$ is a valid kernel on constants;

- *else if s and t are compound terms and have different functors, i.e., $s = f(s_1,\ldots,s_n)$ and $t = g(t_1,\ldots,t_m)$, then*
$$K(s,t) = \iota(f^{/n}, g^{/m}) \tag{14}$$
where $\iota : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ is a valid kernel on functors;

- *else if s and t are compound terms and have the same functor, i.e., $s = f(s_1,\ldots,s_n)$ and $t = f(t_1,\ldots,t_n)$, then*
$$K(s,t) = \iota(f^{/n}, f^{/n}) + \sum_{i=1}^{n} K(s_i,t_i) \tag{15}$$

- *in all other cases $K(s,t) = 0$.*

Functions κ and ι are called *atomic* kernels as they operate on non-structured symbols. A special but useful case is the atomic delta kernel δ defined as $\delta(x,z) = 1$ if $x = z$ and $\delta(x,z) = 0$ if $x \neq z$.

Example 4 *Consider the two lists $s = [\mathtt{a},\mathtt{b},\mathtt{c}]$ and $t = [\mathtt{a},\mathtt{c}]$. Recall that in Prolog $[\mathtt{a},\mathtt{b}]$ is a shorthand for $.(\mathtt{a},.(\mathtt{b},[]))$ where the functor $./2$ is a data constructor for lists and $[]$ is the data constructor for the empty list. Suppose $\iota(./2,./2) = 0.25$ and $\kappa(x,z) = \delta(x,z)$ for all $x,z \in \mathcal{C}$. Then*

$$\begin{aligned} K(s,t) &= K(.(\mathtt{a},.(\mathtt{b},.(\mathtt{c},[]))),.(\mathtt{a},.(\mathtt{c},[]))) \\ &= \iota(./2,./2) + K(\mathtt{a},\mathtt{a}) + K(.(\mathtt{b},.(\mathtt{c},[])),.(\mathtt{c},[])) \\ &= \iota(./2,./2) + \kappa(\mathtt{a},\mathtt{a}) + \iota(./2,./2) + \kappa(\mathtt{b},\mathtt{c}) + K(.(\mathtt{c},[]),[]) \\ &= 0.25 + 1 + 0.25 + 0 + 0 = 1.5 \end{aligned}$$

The result obtained in the above example is similar to what would be achieved with the kernel on higher-order logic basic terms defined in Gärtner et al. (2004). The following examples illustrate the case of two other common data structures.

Example 5 *Consider the two tuples simulated via a predicate $\mathtt{r}$: $s = \mathtt{r}(\mathtt{a},\mathtt{b},\mathtt{c})$ and $t = \mathtt{r}(\mathtt{d},\mathtt{b},\mathtt{a})$. Suppose $\iota(\mathtt{r}/3,\mathtt{r}/3) = 0$ and $\kappa(x,z) = \delta(x,z)$ for all $x,z \in \mathcal{C}$. Then it immediately follows from the definition that $K(s,t) = 1$.*

Example 6 *As a last example consider data structures intended to describe scientific references:*

$$\begin{aligned} r &= \texttt{article}(\texttt{"Kernels on Gnus and Gnats"}, \texttt{journal}(\texttt{ggj}, 2004)) \\ s &= \texttt{article}(\texttt{"The Logic of Gnats"}, \texttt{conference}(\texttt{icla}, 2004)) \\ t &= \texttt{article}(\texttt{"Armadillos in Hilbert space"}, \texttt{journal}(\texttt{ijaa}, 2004)) \end{aligned}$$

Using $\kappa(x,z) = \delta(x,z)$ *for all* $x,z \in \mathcal{C}$ *and* $\iota(x,z) = \delta(x,z)$ *for all* $x,z \in \mathcal{F}$*, we obtain* $K(r,s) = 1$*,* $K(r,t) = 3$*, and* $K(s,t) = 1$*. The fact that all papers are published in the same year does not contribute to* $K(r,s)$ *or* $K(s,t)$ *since these pairs have different functors describing the venue of the publication; it does contribute to* $K(r,t)$ *as they are both journal papers. Note that strings have been treated as constants (as standard in Prolog). Under our above definition the kernel cannot recognize the fact that r and s share a word in the title.*

A finer level of granularity in the definition of ground term kernels can be gained from the use of typed terms. This extra flexibility may be necessary to specify different kernel functions associated with constants of different type (e.g., numerical vs categorical). Types are also useful to specify different kernels associated to different arguments of compound terms. As detailed below, this allows us to distinguish different roles played by clauses in a proof tree.

Our approach for introducing types is similar to that proposed by Lakshman and Reddy (1991). We denote by $\mathcal{T}$ the ranked set of type constructors, which contains at least the nullary constructor $\perp$. The type signature of a function of arity n has the form $\tau_1 \times, \ldots, \times \tau_n \mapsto \tau'$ where $n \geq 0$ is the number of arguments, $\tau_1, \ldots, \tau_k \in \mathcal{T}$ are their types, and $\tau' \in \mathcal{T}$ is the type of the result. Functions of arity 0 have signature $\perp \mapsto \tau'$ and can therefore be interpreted as constants of type τ'. The type of a function is the type of its result. The type signature of a predicate of arity n has the form $\tau_1 \times, \ldots, \times \tau_n \mapsto \Omega$ where $\Omega \in \mathcal{T}$ is the type of Booleans, and is thus a special case of type signatures of functions. We write $t : \tau$ to assert that t is a term of type τ. We denote by $\mathcal{G}$ the set of all typed ground terms, by $\mathcal{C} \subset \mathcal{G}$ the set of all typed constants, and by $\mathcal{F}$ the set of typed functors. Finally we introduce a (possibly empty) set of *distinguished* type signatures $\mathcal{D} \subset \mathcal{T}$ that can be useful to specify ad-hoc kernel functions on certain compound terms.

Definition 6 (Sum kernels on typed terms) *The kernel between two typed terms t and s is defined inductively as follows:*

- *if* $s \in \mathcal{C}$*,* $t \in \mathcal{C}$*,* $s : \tau$*,* $t : \tau$ *then*

$$K(s,t) = \kappa_\tau(s,t) \tag{16}$$

  where $\kappa_\tau : \mathcal{C} \times \mathcal{C} \mapsto \mathbb{R}$ *is a valid kernel on constants of type* τ*;*

- *else if s and t are compound terms that have the same type but different functors or signatures, i.e.,* $s = f(s_1, \ldots, s_n)$ *and* $t = g(t_1, \ldots, t_m)$*,* $s : \sigma_1 \times, \ldots, \times \sigma_n \mapsto \tau'$*,* $t : \tau_1 \times, \ldots, \times \tau_m \mapsto \tau'$*, then*

$$K(s,t) = \iota_{\tau'}(f^{/n}, g^{/m}) \tag{17}$$

  where $\iota_{\tau'} : \mathcal{F} \times \mathcal{F} \mapsto \mathbb{R}$ *is a valid kernel on functors that construct terms of type* τ'

- *else if s and t are compound terms and have the same functor and type signature, i.e., $s = f(s_1,\ldots,s_n)$, $t = f(t_1,\ldots,t_n)$, and $s,t : \tau_1\times,\ldots,\times\tau_n \mapsto \tau'$, then*

$$K(s,t) = \begin{cases} \kappa_{\tau_1\times,\ldots,\times\tau_n\mapsto\tau'}(s,t) \\ \qquad\qquad if\ (\tau_1\times,\ldots,\times\tau_n \mapsto \tau') \in \mathcal{D} \\ \iota_{\tau'}(f^{/n}, f^{/n}) + \sum_{i=1}^{n} K(s_i,t_i) \quad otherwise \end{cases} \tag{18}$$

  where $\kappa_{\tau_1\times,\ldots,\times\tau_n\mapsto\tau'} : \mathcal{U}\times\mathcal{U} \mapsto \mathbb{R}$ is a valid kernel on terms having distinguished type signature $\tau_1\times,\ldots,\times\tau_n \mapsto \tau' \in \mathcal{D}$.

- *in all other cases $K(s,t) = 0$.*

Versions of the kernels which combine arguments using products instead of sums can be easily defined as follows.

Definition 7 (Product kernels on untyped terms) *Use Definition 5 replacing Equation (15) with*

$$K(s,t) = \iota(f^{/n}, f^{/n}) \prod_{i=1}^{n} K(s_i,t_i) \tag{19}$$

Definition 8 (Product kernels on typed terms) *Use Definition 6 replacing Equation (18) with*

$$K(s,t) = \begin{cases} \kappa_{\tau_1\times,\ldots,\times\tau_n\mapsto\tau'}(s,t) \\ \qquad\qquad if\ (\tau_1\times,\ldots,\times\tau_n \mapsto \tau') \in \mathcal{D} \\ \iota_{\tau'}(f^{/n}, f^{/n}) \prod_{i=1}^{n} K(s_i,t_i) \quad otherwise \end{cases} \tag{20}$$

The families of functions in Definitions 5–8 are special cases of Haussler's decomposition kernels and therefore they are positive semi-definite (see Appendix A for formal results).

4.5 Kernels on Prolog Proof Trees

In order to employ full typed term kernels (as in Definitions 6 and 8) on proof trees, we need a typed syntax for their ground term representation. We will assume the following default types for constants: `num` (numerical) and `cat` (categorical). Types for compounds terms will be either `fact`, corresponding to leaves in the proof tree, `clause` in the case of internal nodes, and `body` when containing the body of a clause. Note that regardless of the specific implementation of kernels between types, such definitions imply that we actually compare the common subpart of proofs starting from the goal (the visitor clause), and stop whenever the two proofs diverge.

A number of special cases of kernels can be implemented with appropriate choices of the kernel for compound and atomic terms. The *equivalence* kernel outputs one iff two proofs are equivalent, and zero otherwise:

$$K_{equiv}(s,t) = \begin{cases} 1 & \text{if } s \equiv t \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

We say that two proofs are equivalent if the same sequence of clauses is proven in the two cases, and the head arguments in corresponding clauses satisfy a given equivalence relation. A trivial

implementation of proof equivalence can be obtained using the product kernel on typed terms (Definition 8) in combination with the delta kernel on constants and functors.

In many cases, we will be interested in ignoring some of the arguments of a pair of ground terms when computing the kernel between them. As an example, consider the atom bond representation of a molecule shown in the upper part of Figure 1. The first arguments of `atm` and `bond` predicates are simply molecule and atom identifiers, and we would like to ignore their values when comparing two molecules together. This can be implemented using a special `ignore` type for arguments that should be ignored in comparisons, and a corresponding *constant* kernel which always outputs a constant value:

$$K_\eta(s,t) = \eta.$$

It is straightforward to see that K_η is a valid kernel provided $\eta \geq 0$. The constant η should be set equal to the identity element of the operation used to combine results for the different arguments of the term under consideration, that is $\eta = 0$ for the sum kernel and $\eta = 1$ for the product one.

The extreme use for this kernel is that of implementing the notion of *functor equality* for proof tree nodes, where two nodes are the same iff they share the same functor, regardless of the specific values taken by their arguments. Given two ground terms $s = f(s_1,\ldots,s_n)$ and $t = g(t_1,\ldots,t_m)$ the functor equality kernel is given by:

$$K_f(s,t) = \begin{cases} 0 & \text{if } type(s) \neq type(t) \\ \delta(f^{/n}, g^{/m}) & \text{if } s,t : \texttt{fact} \\ \delta(f^{/n}, g^{/m}) \star K(s_n, t_m) & \text{if } s,t : \texttt{clause} \\ K(s,t) & \text{if } s,t : \texttt{body} \end{cases} \tag{22}$$

where K is a kernel on ground terms as defined in Section 4.4, and the operator $\star$ can be either sum or product. Note that if s and t represent clauses (i.e., internal nodes of the proof tree), the comparison skips clause head arguments, represented by the first $n-1$ (resp. $m-1$) arguments of the terms, and compares the bodies (the last argument, see Section 4.3) thus proceeding on the children of the nodes. This kernel allows to define a non trivial equivalence between proofs (or parts of them) checking which clauses are proved in sequence and ignoring the specific values of their head arguments.

Moreover, it will often be useful to define custom kernels for specific terms by using distinguished type signatures. Appendix B contains details of possible kernel configurations as sets of Prolog clauses, while Appendix C contains the Prolog code for all visitors and kernel configurations employed in the experimental section.

5. Experiments

We run a number of experiments in order to demonstrate the possibilities of the proposed method. In particular, we aim to empirically show that

1. statistical learning in the ILP setting can be addressed, scaling better than typical ILP algorithms with the complexity of the target hypothesis;

2. problems which are difficult for traditional ILP algorithms can be solved;

3. both classification and regression tasks can be effectively handled;

4. significant improvements on real world applications can be achieved.

For classification tasks, we employed SVM (Cortes and Vapnik, 1995) using the Gist[6] implementation, which permits to separate kernel calculation from training by accepting the complete kernel matrix as input. We compared our method with two popular and diverse ILP algorithms: Tilde (Blockeel and De Raedt, 1998), which upgrades C4.5 to induction of logical decision trees, and Progol (Muggleton, 1995), which learns logical theories using inverse entailment.

Regression is quite a difficult task for ILP techniques, and few algorithms currently exist which are able to address it. Conversely, our definition of kernel over proof trees allows us to apply standard kernel methods for regression, such as kernel ridge regression (KRR, (Poggio and Smale, 2003)) and support vector regression (Vapnik, 1995). We report results using the former approach, as training was more stable and no significant difference in performance could be noted. However, when dealing with large data sets, the latter method would be preferable for efficiency reasons. In Section 5.4 we report regression experiments comparing our approach to a number of propositional as well as relational learners.

5.1 Bongard Problems

In order to provide a full basic example of visitor program construction and exploitation of the proof tree information, we created a very simple Bongard problem (Bongard, 1970). The concept to be learned can be represented with the simple pattern *triangle-X^n-triangle* for a given *n*, meaning that a positive example is a scene containing two triangles nested into one another with exactly *n* objects (possibly triangles) in between. Figure 3 shows a pair of examples of such scenes with their representation as Prolog facts and their classification according to the pattern for $n = 1$.

A possible example of background knowledge introduces the concepts of *nesting* in containment and *polygon* as a generic object, and can be represented as follows:

```
inside(X,A,B):- in(X,A,B).                    % clause nr 1
inside(X,A,B):-                               % clause nr 2
    in(X,A,C),
    inside(X,C,B).
polygon(X,A) :-  triangle(X,A).               % clause nr 3
polygon(X,A) :-  rectangle(X,A).              % clause nr 4
polygon(X,A) :-  circle(X,A).                 % clause nr 5
```

A visitor exploiting such background knowledge, and having hints on the target concept, could be looking for two polygons contained one into the other. This can be represented as:

```
visit(X):-                                    % clause nr 6
    inside(X,A,B),polygon(X,A),polygon(X,B).
```

Figure 4 shows the proofs trees obtained running such a visitor on the first Bongard problem in Figure 3.

A very simple kernel can be employed to solve such a task, namely an equivalence kernel with functor equality for nodewise comparison. For any value of *n*, such a kernel maps the examples into a feature space where there is a single feature discriminating between positive and negative

6. The Gist package by W. Stafford Noble and P. Pavlidis is available from `http://microarray.genomecenter.columbia.edu/gist/`.

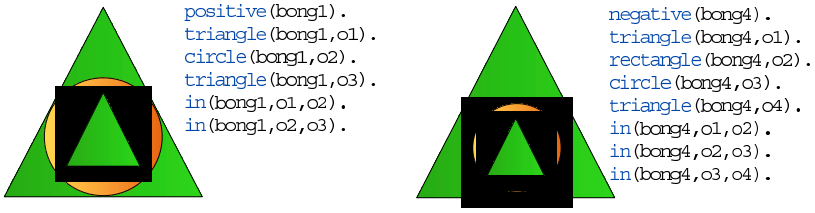


Figure 3: Graphical and Prolog facts representation of two Bongard scenes. The left and right examples are positive and negative, respectively, according to the pattern *triangle-X-triangle*.

examples, while the simple use of ground facts without intensional background knowledge would not provide sufficient information for the task.

The data set was generated by creating m scenes each containing a series of ℓ randomly chosen objects nested one into the other, and repeating the procedure for ℓ varying from 2 to 20. Moreover, we generated two different data sets by choosing $m = 10$ and $m = 50$ respectively. Finally, for each data set we obtained 15 experimental settings denoted by $n \in [0, 14]$. In each setting, positive examples were scenes containing the pattern *triangle-X*^{n}*-triangle*. We run an SVM with the above mentioned proof tree kernel and a fixed value $C = 10$ for the regularization parameter, on the basis that the data set is noise free. We evaluated its performance with a leave-one-out (LOO) procedure, and compared it to the empirical error of Tilde and Progol trained on the same data and background knowledge (including the visitor). Here we focus on showing that ILP algorithms have troubles finding a consistent hypothesis for this problem, hence we did not measure their generalization.

Figure 5(a) plots results for $m = 10$. Both Tilde and Progol stopped learning the concept for $n > 4$. Progol found the trivial empty hypothesis for all $n > 4$ apart from $n = 6$, and Tilde for all $n > 9$. While never learning the concept with 100% generalization accuracy, the SVM performance was much more stable when increasing the nesting level corresponding to positive examples. Figure 5(b) plots results for $m = 50$. Progol was extremely expensive to train with respect to the other methods. It successfully learned the concept for $n \leq 2$, but we stopped training for $n = 3$ after more than one week training time on a 3.20 GHz PENTIUM IV. Tilde stopped learning the concept for $n > 8$, and found the trivial empty hypothesis for $n > 12$. Conversely, the SVM was almost always able to learn the concept with 100% generalization accuracy, regardless of its complexity level.

Note that in order for the ILP algorithms to learn the target concept regardless of the nesting level, it would be necessary to provide a more informed `inside` predicate, which explicitly contains such nesting level as one of its arguments. The ability of the kernel to extract information from the predicate proof, on the other hand, allows our method to be employed when only partial background knowledge is available, which is typically the case in real world applications.

5.2 *M*-of-*N* Problems

The possibility to plug background knowledge into the kernel allows addressing problems that are notoriously hard for ILP approaches. An example of such concepts is the *M*-of-*N* one, which expects the model to be able to count and make the decision accordingly.

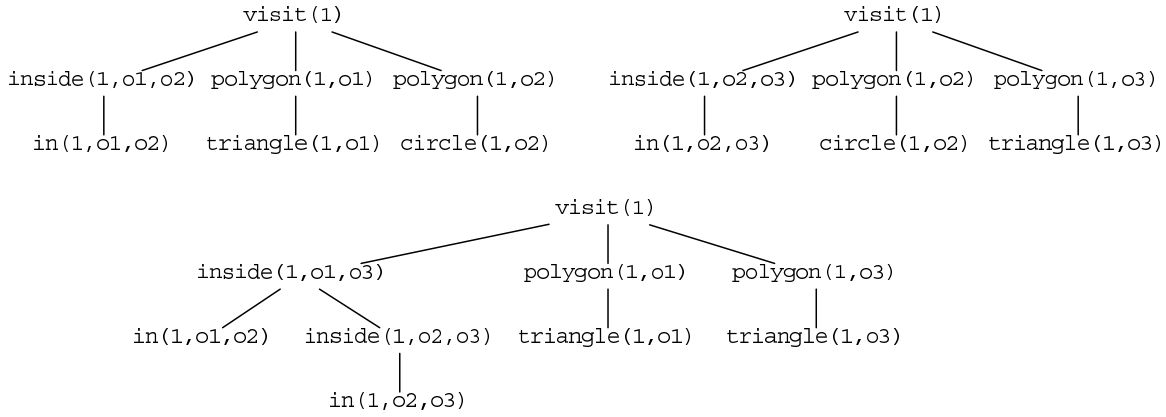


Figure 4: Proof trees obtained by running the visitor on the first Bongard problem in Fig. 3.

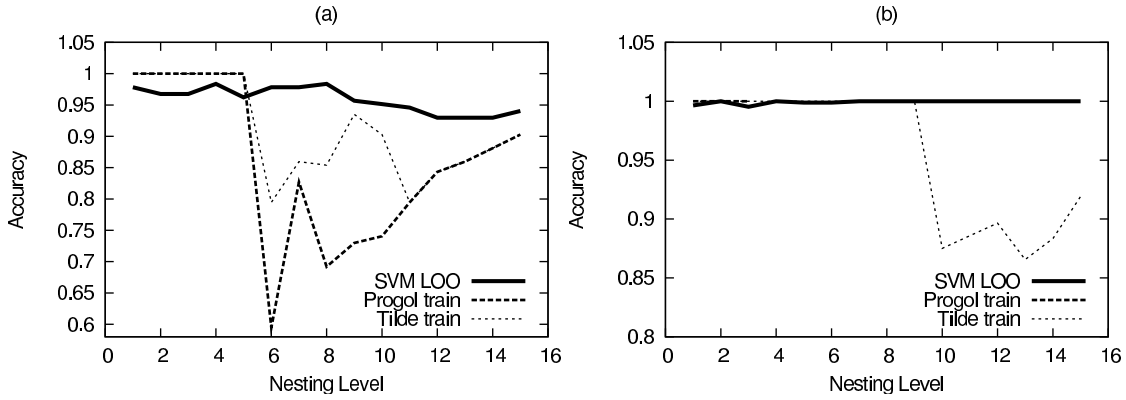


Figure 5: Comparison between SVM leave-one-out error, Progol and Tilde empirical error in learning the *triangle-X*^{n}*-triangle* for different values of n, for data sets corresponding to $m = 10$ (a) and $m = 50$ (b).

		-1	1	
	-1	528	0	True
	1	94	833	
		Predicted		

Table 1: Contingency table for the strings task with default regularization parameter. Predicted class is on columns, true class on rows.

We represented this kind of tasks with a toy problem. Examples are strings of integers $i \in [0,9]$, and a string is positive iff more than a half of its pairs of consecutive elements is ordered, where we employ the partial ordering relation $\leq$ between numbers. In this task, M and N are example dependent, while their ratio is fixed.

As background knowledge, we introduced the concepts of "length two substring" and "pairwise ordering":

```
substr([A,B],[A,B|_T]).
substr([A,B],[_H|T]):-
    substr([A,B],T).

comp(A,B):- A @> B.
comp(A,B):- A @=< B.
```

We then designed a visitor which looks for a substring of length two in the example, and compares its elements:

```
visit(X):-
    string(X,S),substr([A,B],S),comp(A,B).
```

We also declared `substr` to be a leaf predicate, thus pruning the proof tree as explained in Section 3.4, because we are not interested in where the substring is located within the example.

The kernel we employed for this task is a sum kernel with functor equality for nodewise comparison. This kernel basically counts the number of clauses proved in the common subpart of two proof trees, where common means that the same clauses were proved regardless of the specific values of their head arguments.

The data set was created in the following way: the training set was made of 150 randomly generated strings of length 4 and 150 strings of length 5; the test set was made of 1455 randomly generated strings of length from 6 to 100. This allowed to verify the generalization performance of the algorithm for lengths very different from the ones it was trained on.

Accuracy on the test set for a default value of the regularization parameter $C = 1$ was 93.5%, with a contingency table as in Table 1. Moreover, false negatives were the nearest to the decision threshold, and slightly modifying the regularization parameter led to 100% accuracy. On the other hand, neither Tilde nor Progol were able to induce any approximation of the target concept with the available background knowledge. A number of problems prevented them from learning:

1. All proofs of a given predicate (`substr`) were necessary ingredients for the target concept.

2. Counting such proofs was needed, conditioned on the proof details.

3. Gain measures were useless in guiding Tilde hypothesis search, as single atoms forming the target concept had no discriminative power if taken alone.

These problems are due to the need for an aggregation predicate (in this case count) to correctly define the target concept. Dealing with aggregation is known to be difficult for relational learning (Perlich and Provost, 2003; Knobbe et al., 2002).

In order for Progol to learn the target concept, two explicit conditioned counting predicates had to be provided, counting the number of ordered (resp. unordered) length two substrings of a given string. Tilde was still unable to learn the concept with such background knowledge, due the above mentioned problem with gain at intermediate steps of the search, and full lookahead of all building predicates was needed. Again, this is a known problem for decision tree learners (Van de Velde, 1989).

5.3 Protein Fold Classification

In this experiment, we tested our methodology on the protein fold classification problem studied by Turcotte et al. (2001). The task consists of classifying proteins into SCOP folds, given their high-level logical descriptions about secondary structure and amino acid sequence. SCOP is a manually curated database of proteins hierarchically organized according to their structural properties. At the top level SCOP groups proteins into four main classes (all-α, all-β, α/β, and $\alpha+\beta$). Each class is then divided into folds that group together proteins with similar secondary structures and three-dimensional arrangements. We used the data set made available as a supplement to the paper by Turcotte et al. (2001)[7] that consists of the five most populated folds from each of the four main SCOP classes. This setting yields 20 binary classification problems. The data sets for each of the 20 problems are relatively small (from about 30 to about 160 examples per fold, totaling 1143 examples).

We relied on the background knowledge provided in Turcotte et al. (2001), to design a set of visitors managing increasingly complex information. A global visitor was used to extract protein level information, such as its length and the number of its α or β secondary structure segments. A local visitor explored the details of each of such segments, while a connection visitor looked for pairs of adjacent segments within the protein. Numerical values were normalized within each top level fold class. The kernel configuration mainly consisted of type signatures aiming to ignore identifiers and treat some of the numerical features as categorical ones. A functor equality kernel was employed for those nodes of the proofs which did not contain valuable information in their arguments. Code details for visitors and kernel configuration can be found in Appendix-C.3.
Following Turcotte et al. (2001), we measured prediction accuracy by 10-fold cross-validation, micro-averaging the results over the 20 experiments by summing contingency tables. The proof-tree kernel was combined with a Gaussian kernel (see Equation (10)) in order to model nonlinear interactions between the features extracted by the visitor program. Model selection (i.e., choice of the Gaussian width γ and the SVM regularization parameter C) was performed for each binary problem with a LOO procedure before running the 10-fold cross validation. Table 2 shows comparisons between the best setting for Progol (as reported by Turcotte et al. (2001)), which uses both propositional and relational background knowledge, results for Tilde using the same setting, and SVM with our kernel over proof trees. The difference between Tilde and Progol is not significant, while our SVM achieves significantly higher overall accuracy with respect to both methods.

7. http://www.bmm.icnet.uk/ilp/data/ml_2000.tar.gz.

	Tilde	Progol	SVM
All-α:			
Globin-like	97.4	95.1	94.9
DNA-binding 3-helical bundle	81.1	83.0	88.9
4-helical cytokines	83.3	70.7	86.7
lambda repressor-like DNA-binding domains	70.0	73.4	83.3
EF Hand-like	71.4	77.6	85.7
All-β:			
Immunoglobulin-like beta-sandwich	74.1	76.3	85.2
SH3-like barrel	91.7	91.4	93.8
OB-fold	65.0	78.4	83.3
Trypsin-like serine proteases	95.2	93.1	93.7
Lipocalins	83.3	88.3	92.9
α/β:			
beta/alpha (TIM)-barrel	69.7	70.7	73.3
NAD(P)-binding Rossmann-fold domains	79.4	71.6	84.1
P-loop containing nucleotide triphosphate hydrolases	64.3	76.0	76.2
alpha/beta-Hydrolases	58.3	72.2	86.1
Periplasmic binding protein-like II	79.5	68.9	79.5
$\alpha+\beta$:			
Interleukin 8-like chemokines	92.6	92.9	96.3
beta-Grasp	52.8	71.7	88.9
Ferredoxin-like	69.2	83.1	76.9
Zincin-like	51.3	64.3	79.5
SH2-like	82.1	76.8	66.7
Micro average:	75.2 $\pm$2.5	78.3 $\pm$2.4	83.6 $\pm$2.2

Table 2: Protein fold classification: 10-fold cross validation accuracy (%) for Tilde, Progol and SVM for the different classification tasks, and micro averaged accuracies with 95% confidence intervals. Results for Progol are taken from Turcotte et al. (2001).

5.4 QSAR Regression Tasks

Quantitative structure activity relationship (QSAR) tasks deal with the problem of predicting the biological activity of a molecule given its chemical structure. They can thus be naturally represented as regression problems. The chemical structure of molecules is typically represented by atom and bond predicates, possibly specifying also non topological attributes such as atom and bond detailed types and atom partial charge. Additional features include molecule physico-chemical properties, such as its weight, its hydrophobicity (*logP*) and *lumo*, which is the energy of the molecule lowest unoccupied orbital. Intensional background knowledge can be represented by predicates looking for ring structures and functional groups within the molecule, such as benzene, anthracene and nitro. Relational features can also be propositionalized in different ways in order to employ propositional learners.

In the following we focused on two well known QSAR data sets, mutagenesis and biodegradability, and compared to published results for different relational and propositional learners, always attaining to their same experimental settings. In both cases we run a preliminary model selection phase (optimizing Gaussian width and regularization parameter) on an additional 10 fold cross validation procedure. We employed the Pearson correlation coefficient as a standard performance measure, and two tailed Fisher z tests at 0.05 significance level in order to verify if the performance difference between pairs of methods was statistically significant.

5.4.1 MUTAGENESIS

The mutagenicity problem is a standard benchmark for ILP approaches. The problem is treated in Srinivasan et al. (1996) as a binary classification task (mutagenic vs. non-mutagenic). Here we focused on its original formulation as a regression task, and compared to the results presented in Kramer (1999) for the regression friendly data set.

We employed a global visitor exploring physico-chemical properties of the molecule, that is *logp*, *lumo*, *ind1* and *inda*. We then developed a set of visitors exploiting the ring theory for nitro aromatic and heteroaromatic compounds, each looking for compounds of a certain type, and extracting the properties of the atoms belonging to it. We employed pruned trees for such visitors, as described in the example shown in Section 3.4. Kernel configuration was mostly made of type signatures as for the protein fold classification task (Section 5.3, see Appendix-C.4 for code details). Competing algorithms included S-CART (Kramer, 1999), which is an upgrade of CART to first order logic, and M5' (Quinlan, 1993; Wang and Witten, 1997), a propositional regression-tree induction algorithm. Propositionalization was conducted either by (P) counting occurrences of different functional groups (together to physico-chemical global properties), or (SP) running a supervised stochastic propositionalization algorithm as described in Kramer (1999). Table 3 reports experimental comparisons on four 10 fold cross validation procedures. Our method consistently outperforms all other learners, and such difference is significant on four out of five cases.

5.4.2 BIODEGRADABILITY

Degradation is the process by which chemicals are transformed into components which are not considered pollutants. A number of different pathways are responsible for such process, depending on environmental conditions. Blockeel et al. (2004) conducted a study focused on aqueous biodegradation under aerobic conditions. Low and high estimates of half life time degradation rate were collected for 328 molecules. The regression task consisted in predicting the natural logarithm of the

System	r
KRR	$\mathbf{0.898}(\mathbf{0.002})$
S-CART	0.830 (0.020)
P + S-CART	0.834 (0.010)
P + M5'	$\mathbf{0.893}(\mathbf{0.001})$
P + SP + S-CART	0.767 (0.038)
P + SP + M5'	0.835 (0.012)

Table 3: Pearson correlation coefficient for the different learners on the regression friendly mutagenesis data set. Results are averaged over four 10-fold cross validation procedures, and standard deviations over the four procedures are reported. Boldface numbers are significantly better than plain ones. All other differences are not significant. Results for all systems except for KRR are taken from Kramer (1999).

arithmetic mean of the low and high estimate for a given molecule. A comprehensive background knowledge of rings and functional groups was available as for the mutagenesis data set. Moreover, relational features had been propositionalized in two different ways. Four sets of features were thus made available to learning algorithms (Blockeel et al., 2004):

- *Global* consisted of molecule physico-chemical properties, namely weight and logP.
- *P1* were counts of rings and functional groups defined in the background theory.
- *P2* were counts of small substructures of molecules (all connected substructures of two or three atoms, those of four with a star topology).
- *R* contained full relational features: atoms, bonds, ring and functional structures described by their constituent atoms and those connecting them to the rest of the molecule.

We developed appropriate visitors for each of these feature sets. Visitors for full relational features (*R*) explored atoms within rings and functional structures as in the mutagenesis task, additionally including information about atoms connecting each compound to the rest of the molecule. Numerical features[8] were normalized. The kernel configuration was again similar to that in the protein fold classification task (Section 5.3), but we also modified the default combining operator for a few type signatures in order to compared substructures of the same type only (code details in Appendix-C.5).

A number of relational and propositional learners were compared in Blockeel et al. (2004) on different feature sets: apart from S-CART and M5', already introduced for the mutagenesis data set, simple linear regression (LR) and the version of Tilde learning regression trees (Blockeel and De Raedt, 1998). Table 4 reports average and standard deviation of Pearson correlation coefficient on five 10-fold cross validation procedures, for different combinations of the feature sets. Our kernel outperforms all other methods on four out of five scenarios, and in two cases results are significantly better than any competitor (see Figure 6).

8. Apart from those in *P1* which had a small range ($[0,4]$).

System	G	G+P1	G+P2	G+R	G+P1+P2+R
KRR	0.472 (0.005)	0.701 (0.005)	0.683 (0.006)	0.694 (0.005)	0.695 (0.005)
Tilde	0.487 (0.020)	0.596 (0.029)	0.615 (0.014)	0.616 (0.021)	0.595 (0.020)
S-CART	0.476 (0.031)	0.563 (0.010)	0.595 (0.032)	0.605 (0.023)	0.606 (0.032)
M5'	0.503 (0.012)	0.579 (0.024)	0.646 (0.013)		
LR	0.436 (0.004)	0.592 (0.014)	0.443 (0.026)		

Table 4: Pearson correlation coefficient for the different learners for various combinations of features on the biodegradability data set. Results are averaged over five 10-fold cross validation procedures, and standard deviations over the five procedures are reported. Results for all systems except for KRR are taken from Blockeel et al. (2004).

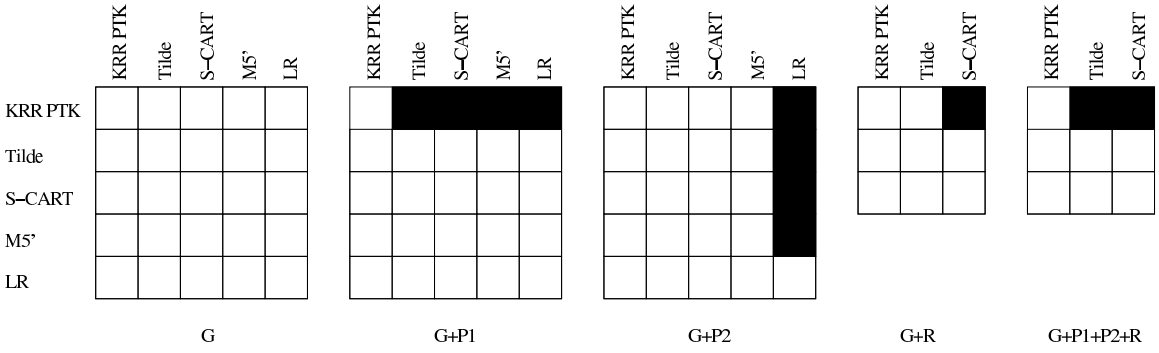


Figure 6: Significance of performance difference between learners for the biodegradability data set. A black box indicates that the learner on the row is significantly better than that on the column for the given feature setting.

System	G	G+P1	G+P2	G+R	G+P1+P2+R
KRR	0.498 (0.004)	0.700 (0.005)	0.683 (0.006)	0.694 (0.005)	0.695 (0.005)
Tilde	0.495 (0.015)	0.612 (0.022)	0.619 (0.021)	0.635 (0.018)	0.618 (0.022)
S-CART	0.478 (0.016)	0.581 (0.015)	0.636 (0.015)	0.659 (0.019)	0.631 (0.026)
M5'	0.502 (0.014)	0.592 (0.013)	0.646 (0.014)		
LR	0.437 (0.005)	0.592 (0.013)	0.455 (0.022)		

Table 5: Pearson correlation coefficient for the different learners for various combinations of features on the biodegradability data set (second batch). Results are averaged over five 10-fold cross validation procedures, and standard deviations over the five procedures are reported. Results for all systems except for KRR are taken from Blockeel et al. (2004).

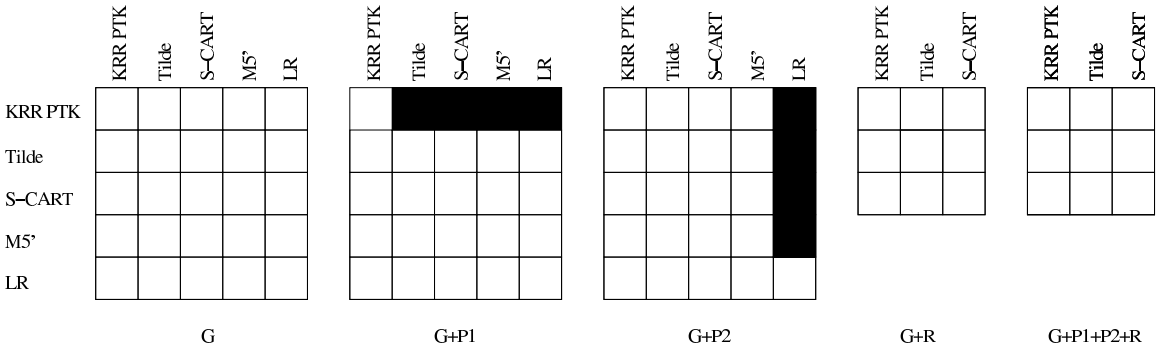


Figure 7: Significance of performance difference between learners for the biodegradability data set (second batch). A black box indicates that the learner on the row is significantly better than that on the column for the given feature setting.

In a second batch of experiments, Blockeel et al. (2004) separately predicted low and high estimates of half life time degradation rate, and reported the mean of such predictions. Results are shown in Table 5. While other methods often improve their performance over the previous batch, our method is almost unaffected. Still, it outperforms all learners on the same four scenarios, and in one case it obtains significantly better results than any other algorithm (Figure 7).

6. Discussion and Related Work

When tackling inductive learning problems using the presented techniques, there are a number of design decisions to be made. These include: the choice of the background theory $\mathcal{B}$, visitor program $\mathcal{V}$ and also the kernel K. As compared to traditional ILP, the background theory $\mathcal{B}$ is similar, the visitor program plays the role of the declarative and inductive bias, and the kernel can perhaps be related to some distance based learning approaches (Ramon and Bruynooghe, 1998; Horvath et al., 2001). The visitor program, however, constitutes an entirely different form of bias than the typical declarative language bias employed in ILP (Nédellec et al., 1996), which is purely syntactic. The visitor incorporates a much more procedural bias, which is perhaps more similar to explanation-

based learning (Mitchell et al., 1986). Indeed, explanation-based learning also starts from proof trees or traces for specific examples and then generalizes them using deductive methods (such as regression or partial evaluation (Van Harmelen and Bundy, 1988)). There has, however, also been a strong interest in integrating inductive concept-learning with explanation-based learning (Mitchell et al., 1986). To some extent, the combination of the proof trees with the kernel realizes such an integration, although it does not produce interpretable rules, due to the use of the kernel. The notion and use of background theory has—to some degree—always been debated. The reason is that, on the one hand, it provides the user with a flexible means to guide and influence the learning process, but, on the other hand, it is not always easy to define a background theory that will yield accurate hypotheses. For instance, in applying traditional ILP systems to the Bongard example (Section 5.1), it is clear that by using only the outcome of the `inside` predicate, one loses the information of how many objects are between the outermost and innermost triangle. But this could easily be fixed by defining `inside(X,Y,Z)` as "X is inside Y with Z objects between them." In other words, a change of background definitions makes it possible to learn the correct concept, even by traditional ILP systems. This is one example that shows that background theory is a powerful but sometimes hard to master tool.

To gain some further insights into the relationship of our method to traditional ILP, let us try to relate the background theory to the visitor program. From an ILP perspective, it does seem natural to add the visitor program to the background theory and run the traditional ILP system. Whereas this is—in principle—possible, there are some major differences that would result. Indeed, the ILP system could only use the predicates mentioned in the visitor program as conditions in its hypotheses, and it would not have any means to look into the trace or proof. For instance, if there is a predicate *v* in the visitor program that is defined using two different clauses, the ILP system will not be able to distinguish instances of *v* proven using the first clause from those proven using the second one. Also, differences and similarities between proof trees could not be discovered unless one would also add the meta-program (implemented as a Prolog predicate) that generates the proof tree to the background theory. In this case, the information about the structure of proof trees and the clauses being used in there could be employed in the hypotheses of the ILP system. This would yield conditions such as `prove(visitor(x), proof-tree)`. However, since ILP systems have severe search problems when dealing with large structured and terms and recursion, this idea cannot be applied in practice. The use of kernels to realize the generalization is much more appealing because there is no search involved in computing the kernel. Finally, let us remark that it would be interesting to further investigate the design choices $(\mathcal{B}, \mathcal{V}, K)$ to be made. In particular, one may wonder under what conditions two possible choices (say $(\mathcal{B}, \mathcal{V}, K)$ and $(\mathcal{B}', \mathcal{V}'', K')$) are equivalent, and whether this would allow us to reformulate one element (say the visitor) as a part of another one (say the background theory).

7. Conclusions

We have introduced the general idea of kernels over program traces and specialized it to the case of Prolog proof trees in the logic programming paradigm. The theory and the experimental results that we have obtained indicate that this method can be seen as a successful attempt to bridge several aspects of symbolic and statistical learning, including the ability of working with relational data, the incorporation of background knowledge in a flexible and principled way, and the use of regularization. Computational complexity is also an advantage compared to typical ILP systems. The kernel

matrix can be computed in time quadratic in the size of the training set and the complexity of the learning problem is that of the kernel method employed (e.g., SVM or KRR) which is typically inferior to ILP algorithms. This may potentially open the road towards some large-scale applications of learning in the ILP setting.

The advantages of the proposed approach were experimentally verified. The Bongard problems showed that our method scales better than typical ILP algorithms with the complexity of the target concept. Furthermore, it is able to effectively address problems (like the *M*-of-*N* one) that require precise counting, and are difficult to solve with classic ILP approaches. Both classification and regression tasks can be naturally handled using appropriate kernel methods. Finally, the robust nature of statistical learning can offer advantages with respect to symbolic approaches when dealing with noisy data sets, as shown by the improved performance on the bioinformatics and chemoinformatics tasks.

Besides the cases of classification and regression that have been studied in this paper, other learning tasks could naturally benefit from the proposed framework including clustering, ranking, and novelty detection. One advantage of ILP as compared to the present work is the intrinsic ability of ILP to *generate* transparent explanations of the learned function. Developing kernel machines capable of providing transparent predictions and the use of kernel-based approaches to guide hypothesis search as in ILP remain interesting open issues.

Acknowledgments

This research is supported by EU Grant APrIL II (contract n° 508861). PF and AP are also partially supported by MIUR Grant 2003091149_002. We would like to thank the anonymous reviewers whose comments contributed to improve the paper substantially.

Appendix A. Proofs of Theorems

We give in this appendix a result showing that the class of functions studied in this paper are positive semi-definite and therefore valid Mercer kernels.

Theorem 9 *The kernel function on Prolog ground terms given in Definition 5 is positive semi-definite.*

Proof. Let us introduce the following decomposition structure (see Shawe-Taylor and Cristianini, 2004): $\mathcal{R} = \langle (X_1, X_2), R, (k_1, k_2) \rangle$ with $X_1 = \mathcal{F}$, $X_2 = (\mathcal{F}, \mathcal{U})$, and

$$R = \left\{ (f^{/n}, (f^{/n}, a), s) \text{s.t. } s \text{ is a term having functor } f^{/n} \text{ and tuple of arguments } a \right\}.$$

Then it can be immediately verified that the kernel function of Equations (14) and (15) correspond to the direct sum decomposition kernel associated with the decomposition structure $\mathcal{R}$ if $k_1 = \iota$ and $k_2((f^{/n}, a), (g^{/m}, b)) = \delta(f^{/n}, g^{/m}) k'(a, b)$ where given $a = (s_1, \ldots, s_n)$ and $b = (t_1, \ldots, t_n)$

$$k'(a, b) = \sum_{i=1}^{n} K(s_i, t_i).$$

Note that k' is a valid kernel if K is (being a direct sum). The proof then follows by induction using the fact that kernels for base steps (κ (Equation (13)) and ι (Equation (14))) are by hypothesis positive semi-definite, and the induction step simply consists of combining positive semi-definite kernels by direct sum which itself produces valid kernels (Theorem 4). □

Theorem 10 *The kernel function on typed Prolog ground terms given in Definitions 6 is positive semi-definite.*

Proof. We can use the same technique as for Theorem 9 but including types in the decomposition structure: $\mathcal{R} = \langle (X_1, X_2), R, (k_1, k_2) \rangle$ with $X_1 = (\mathcal{F}, \mathcal{T})$, $X_2 = (\mathcal{F}, \mathcal{T}, \mathcal{U})$, and

$$R = \{((f^{/n}, \tau), (f^{/n}, (\tau_1 \times, \ldots, \times \tau_n \mapsto \tau), t), s) \text{ s.t. } s \text{ is a term having functor } f^{/n}, \\ \text{tuple of arguments } t, \text{ and type signature } \tau_1 \times, \ldots, \times \tau_n \mapsto \tau\}.$$

The kernel function of Equations (17) and (18) correspond to the direct sum decomposition kernel associated with the decomposition structure $\mathcal{R}$ if:

$$k_1((f^{/n}, \tau), (g^{/m}, \sigma)) = \delta(\tau, \sigma)\iota_\tau(f^{/n}, g^{/m})$$

and

$$k_2((f^{/n}, \tau_1 \times, \ldots, \times \tau_n \mapsto \tau, a), (g^{/m}, \sigma_1 \times, \ldots, \times \sigma_m \mapsto \sigma, b)) = \\ \delta(f^{/n}, g^{/m})\delta(\tau_1 \times, \ldots, \times \tau_n \mapsto \tau, \sigma_1 \times, \ldots, \times \sigma_m \mapsto \sigma)k'(a, b).$$

The proof follows from Theorem 4 and by induction using the fact that κ_τ (Equation (16)), ι_τ (Equation (17)) and kernels on distinguished types (see Equation (18)) are by hypothesis valid kernels. □

Theorem 11 *The kernel functions on Prolog ground terms given in Definitions 7 and 8 are positive semi-definite.*

Proof. Same as in Theorem 9 and 10 respectively, simply replacing direct sums with tensor products. □

Appendix B. Kernel Configuration Details

The kernel specification defines the way in which data and knowledge should be treated. The default way of treating compound terms can be declared to be either *sum* or *product*, by writing `compound_kernel(sum)` or `compound_kernel(product)` respectively.

The default atomic kernel is the delta one for symbols, and the product for numbers. Such behavior can be modified by directly specifying the type signature of a given clause or fact. As an example, the following definition overrides the default kernel between `atm` terms in mutagenesis:

```
type(atm(ignore,ignore,cat,cat,num)).
```

It allows to ignore identifiers for molecule and atom, and change the default behavior for atom type (which is a number) to categorical. At this level, it is possible to specify a combining operator for predicate arguments which is different from the default one:

```
type(atm(ignore,ignore,cat,cat,num),product).
```

Here we are stating that atoms of different types will always have zero similarity. Default behaviors can also be overridden by defining specific kernels for particular clauses or facts. This corresponds to specifying distinguished types together to appropriate kernels for them. Thus, the last kernel between atoms could be equivalently specified by writing:

```
term_kernel(atm(_,_,Xa,Xt,Xc), atm(_,_,Ya,Yt,Yc),K) :-
    delta_kernel(Xa,Ya,Ka),
    delta_kernel(Xt,Yt,Kt),
    dot_kernel(Xc,Yc,Kc),
    K is Ka * Kt * Kc.
```

A useful kernel which can be selected is the *functor_equality* kernel as defined in Equation (22). For example, by writing

```
term_kernel(X,Y,K):-
    functor_equality_kernel(X,Y,K).
```

at the end of the configuration file it is possible to force the default behavior for all remaining terms to functor equality, where the combination operator employed for internal nodes will be the one specified with the `compound_kernel` statement.

Appendix C. Visitors and Kernels Used in Experiments

C.1 Bongard Problems

```
visit(X):-
    inside(X,A,B),polygon(X,A),polygon(X,B).

compound_kernel(product).

term_kernel(X,Y,K):-
    functor_equality_kernel(X,Y,K).
```

C.2 *M*-of-*N* Problems

```
visit(X):-
    string(X,S),substr([A,B],S),comp(A,B).

leaf(substr(_,_)).

compound_kernel(sum).

term_kernel(X,Y,K):-
    functor_equality_kernel(X,Y,K).
```

C.3 Protein Fold Classification

```
visit_global(X):-
    normlen(X,Len),
    normnb_alpha(X,NumAlpha),
    normnb_beta(X,NumBeta).

visit_adjacent(X):-
    adjacent(X,A,B,PosA,TypeA,TypeB),
    normcoil(A,B,LenCoil),
    unit_features(A),
    unit_features(B).


leaf(adjacent(_,_,_,_,_,_)).
leaf(normcoil(_,_,_)).

compound_kernel(sum).

type(normlen(ignore,num)).
type(normnb_alpha(ignore,num)).
type(normnb_beta(ignore,num)).
type(normsst(ignore,ignore,ignore,ignore,ignore,num,ignore,num,num,num,ignore)).
type(adjacent(ignore,ignore,ignore,cat,cat,cat)).
type(normcoil(ignore,ignore,num)).

term_kernel(X,Y,K):-
    functor_equality_kernel(X,Y,K).
```

```
visit_unit(X):-
    sec_struc(X,A),
    unit_features(A).

unit_features(A):-
    normsst(A,B,C,D,E,F,G,H,I,J,K),
    has_pro(A).

unit_features(A):-
    normsst(A,B,C,D,E,F,G,H,I,J,K),
    not(has_pro(A))).
```

C.4 Mutagenesis

```
visit_global(X):-
    lumo(X,Lumo),
    logp(X,Logp),
    ind1(X,Ind1),
    inda(X,Inda).
    visit_benzene(X):-
        benzene(X,Atoms),
        atoms(X,Atoms).

    visit_anthracene(X):-
        anthracene(X,[Ring1,Ring2,Ring3]),
        atoms(X,Ring1),
        atoms(X,Ring2),
        atoms(X,Ring3).

    compound_kernel(sum).

type(atm(ignore,ignore,cat,cat,num)).
type(bond(ignore,ignore,ignore,cat)).
type(lumo(ignore,num)).
type(logp(ignore,num)).
type(ind1(ignore,num)).
type(inda(ignore,num)).
```

```
visit_ring_size_5(X):-
    ring_size_5(X,Atoms),
    atoms(X,Atoms).
% ... etc.

    leaf(benzene(_,_)).
    leaf(anthracene(_,_)).
    leaf(ring_size_5(_,_)).
    % ... etc.

    atoms(X,[]).
    atoms(X,[H|T]):-
        atm(X,H,_,_,_),atoms(X,T).
```

```
term_kernel(X,Y,K):-
    functor_equality_kernel(X,Y,K).
```

C.5 Biodegradability

```
 visit_p1(X):-
     sscount(X,_SSType,_SSCount).

 visit_p2(X):-
     p2countnorm(X,_P2Type,_P2Count).
     visit_alcohol(X):-
         alcohol(X,Atoms,Conns),
         atoms(X,Atoms),
         atoms(X,Conns).

     visit_aldehyde(X):-
         aldehyde(X,Atoms,Conns),
         atoms(X,Atoms),
         atoms(X,Conns).

atoms(X,[]).
atoms(X,[H|T]):-
    atm(X,H,_,_,_),
    atoms(X,T).

compound_kernel(sum).

type(atm(ignore,ignore,cat,ignore,ignore)).
type(normlogP(ignore,num)).
type(normmweight(ignore,num)).
type(sscount(ignore,cat,num),product).
type(normp2count(ignore,cat,num),product).

term_kernel(X,Y,K):-
    functor_equality_kernel(X,Y,K).
```

```
visit_global(X):-
    normlogP(X,_LogP),
    normmweight(X,_Mweight).


    visit_ar_halide(X):-
        ar_halide(X,Atoms,Conns),
        atoms(X,Atoms),
        atoms(X,Conns).
    % ... etc.

    leaf(alcohol(_,_,_)).
    leaf(aldehyde(_,_,_)).
    leaf(ar_halide(_,_,_)).
    % ... etc.
```

References

A. W. Biermann and R. Krishnaswamy. Constructing programs from example computations. *IEEE Transactions on Software Engineering*, 2(3):141–153, 1976.

H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

H. Blockeel, S. Dzeroski, B. Kompare, S. Kramer, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. *Applied Artificial Intelligence*, 18(2):157–181, 2004.

M. Bongard. *Pattern Recognition*. Spartan Books, 1970.

M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the Fortieth Annual Meeting on Association for Computational Linguistics*, pages 263–270, Philadelphia, PA, USA, 2002.

C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5:1035–1062, 2004.

C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin (New Series) of the American Mathematical Society*, 39(1):1–49, 2002.

C. M. Cumby and D. Roth. Learning with feature description logics. In S. Matwin and C. Sammut, editors, *Proceedings of the Twelfth International Conference on Inductive Logic Programming*, volume 2583 of *LNAI*, pages 32–47. Springer-Verlag, 2002.

C. M. Cumby and D. Roth. On kernel methods for relational learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 107–114, Washington, DC, USA, 2003.

L. De Raedt, K. Kersting, and S. Torge. Towards learning stochastic logic programs from proof-banks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI'05)*, pages 752–757, 2005.

P. Frasconi, A. Passerini, S. Muggleton, and H. Lodhi. Declarative kernels. Technical Report RT 2/2004, Dipartimento di Sistemi e Informatica, Università di Firenze, 2004.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

T. Gärtner. A survey of kernels for structured data. *SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

T. Gärtner, J. W. Lloyd, and P. A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.

D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167. ACM Press, 2004.

T. Horvath, S. Wrobel, and U. Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1/2):53–80, April 2001.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

A. Karalič and I. Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328, Washington, DC, USA, 2003.

G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41:495–502, 1970.

A. J. Knobbe, A. Siebes, and B. Marseille. Involving aggregate functions in multi-relational search. In *Proceedings of the Sixth European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 2431 of *LNCS*, pages 287–298. Springer-Verlag, 2002.

S. Kramer. Structural regression trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 812–819, Cambridge/Menlo Park, 1996. AAAI Press/MIT Press.

S. Kramer. *Relational Learning vs. Propositionalization: Investigations in Inductive Logic Programming and Propositional Machine Learning*. PhD thesis, Technischen Universität Wien, Wien, Austria, 1999.

S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In *Relational Data Mining*, pages 262–286. Springer-Verlag, NY, 2000.

T. K. Lakshman and U. S. Reddy. Typed prolog: A semantic reconstruction of the mycroft-O'keefe type system. In V Saraswat and K. Ueda, editors, *Proceedings of the 1991 International Symposium on Logic Programming*, pages 202–220, San Diego, CA, October 1991. MIT Press.

C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for svm protein classification. In *Proceedings of the Seventh Pacific Symposium on Biocomputing*, pages 564–575, Lihue, Hawaii, USA, 2002.

J. W. Lloyd. *Logic for learning: learning comprehensible theories from structured data*. Springer-Verlag, 2003.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In R. Greiner and ACM Press D. Schuurmans, editors, *Proceedings of the Twenty-first International Conference on Machine Learning*, pages 552–559, Banff, Alberta, Canada, 2004.

S. Menchetti, F. Costa, and P. Frasconi. Weighted decomposition kernels. In *Proceedings of the Twenty-second International Conference on Machine Learning*, pages 585–592, New York, NY, USA, 2005. ACM Press.

T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation based generalization: a unifying view. *Machine Learning*, 1:47–80, 1986.

T. M. Mitchell, P. E. Utgoff, and R. Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine learning: An artificial intelligence approach*, volume 1, pages 163–190. Morgan Kaufmann, 1983.

S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

S. H. Muggleton, H. Lodhi, A. Amini, and M. J. E. Sternberg. Support vector inductive logic programming. In *Proceedings of the Eighth International Conference on Discovery Science*, volume 3735 of *LNAI*, pages 163–175, 2005.

C. Nédellec, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, volume 32 of *Frontiers in Artificial Intelligence and Applications*, pages 82–103. IOS Press, 1996.

A. Passerini and P. Frasconi. Kernels on prolog ground terms. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1626–1627, Edinburgh, Scotland, UK, 2005.

C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. In *Proceedings of the Ninth SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 167–176. ACM Press, 2003.

T. Poggio and S. Smale. The mathematics of learning: Dealing with data. *Notices of the American Mathematical Society*, 50(5):537–544, 2003.

J. R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243, Amherst, Massachusetts, 1993. Morgan Kaufmann.

J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In D. Page, editor, *Proceedings of the Eighth International Conference on Inductive Logic Programming*, volume 1446 of *LNAI*, pages 271–280. Springer-Verlag, 1998.

S. Russell and P. Norvig. *Artifical Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition, 2002.

B. Schölkopf and M. K. Warmuth, editors. *Kernels and Regularization on Graphs*, volume 2777 of *LNCS*, 2003. Springer.

E. Y. Shapiro. *Algorithmic program debugging*. MIT Press, 1983.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

A. Srinivasan, S. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996.

M. Turcotte, S. H. Muggleton, and M. J. E. Sternberg. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine Learning*, 43(1,2):81–96, April-May 2001.

W. Van de Velde. IDL, or taming the multiplexer. In K. Morik, editor, *Proceedings of the Third European Working Session on Machine Learning*, pages 211–226. Pitmnann, 1989.

F. Van Harmelen and A. Bundy. Explanation based generalization = partial evaluation. *Artificial Intelligence*, 36:401–412, 1988.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

A. C. Varzi. Parts, wholes, and part-whole relations: the prospects of mereotopology. *Data and Knowledge Engineering*, 20:259–286, 1996.

S. V. N. Viswanathan and A. J. Smola. Fast kernels for string and tree matching. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 569–576. MIT Press, Cambridge, MA, 2003.

Y. Wang and I. Witten. Inducing model trees for continuous classes. In *Proceedings of the Ninth European Conference on Machine Learning*, pages 128–137, Prague, Czech Republic, 1997.

J. M. Zelle and R. J. Mooney. Combining FOIL and EBG to speed-up logic programs. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1106–1111, Chambéry, France, 1993.

Journal of Machine Learning Research 7 (2006) 343–378 Submitted 7/05; Revised 12/05; Published 2/06

Using Machine Learning to Guide Architecture Simulation

Greg Hamerly HAMERLY@CS.BAYLOR.EDU
Department of Computer Science
Baylor University
One Bear Place #97356
Waco, TX 76798-7356, USA

Erez Perelman EPERELMA@CS.UCSD.EDU
Jeremy Lau JL@CS.UCSD.EDU
Brad Calder CALDER@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0404, USA

Timothy Sherwood SHERWOOD@CS.UCSB.EDU
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106, USA

Editor: Haym Hirsh

Abstract

An essential step in designing a new computer architecture is the careful examination of different design options. It is critical that computer architects have efficient means by which they may estimate the impact of various design options on the overall machine. This task is complicated by the fact that different programs, and even different parts of the *same* program, may have distinct behaviors that interact with the hardware in different ways. Researchers use very detailed simulators to estimate processor performance, which models every cycle of an executing program. Unfortunately, simulating every cycle of a real program can take weeks or months.

To address this problem we have created a tool called SimPoint that uses data clustering algorithms from machine learning to automatically find repetitive patterns in a program's execution. By simulating one representative of each repetitive behavior pattern, simulation time can be reduced to minutes instead of weeks for standard benchmark programs, with very little cost in terms of accuracy. We describe this important problem, the data representation and preprocessing methods used by SimPoint, the clustering algorithm at the core of SimPoint, and we evaluate different options for tuning SimPoint.

Keywords: *k*-means, random projection, Bayesian information criterion, simulation, SimPoint

1. Introduction

Understanding the cycle level behavior of a processor during the execution of an application is crucial to modern computer architecture research. To gain this understanding, researchers typically employ detailed simulators that model each and every cycle of the underlying machine. Unfortunately, this level of detail comes at the cost of speed. Even on the fastest simulators, modeling

the full execution of a single benchmark can take weeks or months to complete, and nearly all industry standard benchmarks require the execution of a *suite* of programs. For example, the SPEC benchmark suite consists of 26 different programs, requiring the execution of a combined total of approximately 6 trillion instructions. Still worse, architecture researchers need to simulate each benchmark over a variety of different architectural configurations and design options, to find the set of features that provides an appropriate trade-off between performance, complexity, area, and power. The same program binary, with the exact same input, may be run hundreds or thousands of times to examine how, for example, the effectiveness of a given architecture changes with its cache size. Researchers need techniques which can reduce the number of machine-months required to estimate the impact of an architectural modification without introducing an unacceptable amount of error or excessive simulator complexity. We present a method, distributed as a software package called SimPoint, which can meet this need by exploiting the structured way in which individual programs change behavior over time.

As a program executes its behavior changes. These changes are not random, but rather are often structured as sequences of a small number of recurring behaviors, which we term *phases*. Identifying this repetitive and structured behavior can be of great benefit, since it means we only need to sample each unique behavior once to create a complete representation of the program's execution. This is the underlying philosophy of SimPoint (Sherwood et al., 2001, 2002; Perelman et al., 2003; Biesbrouck et al., 2004; Lau et al., 2004, 2005b). SimPoint intelligently chooses a very small set of samples from an executed program called *simulation points* that, when simulated and weighted appropriately, provide an accurate picture of the complete execution of the program. Simulating in detail only these carefully chosen simulation points can save hours of simulation time over a random sampling of the program, while still providing the accuracy needed to make reliable decisions based on the outcome of the cycle level simulation.

Before we developed SimPoint, architecture researchers would simulate SPEC programs for 300 million instructions from the start of execution, or fast forward 1 billion instructions to try to get past the initialization part of the program. These techniques can result in error rates of up to 3736% in predicting the architecture metrics we wish to measure. SimPoint achieves very low error rates (2% average error, 8% maximum error for the results in this paper) and on average reduces simulation time by a factor of 1,500, compared to simply simulating the whole program. This approach is now used by researchers in the architecture community, and companies such as Intel (Patil et al., 2004). This paper shows how repetitive phase behavior can be found in programs through machine learning and describes how SimPoint automatically finds these phases and picks simulation points.

The rest of the paper is laid out as follows. First, Section 2 describes a summary of the simulation methodology in processor architecture research. Section 3 explains the phase behavior paradigm, and defines terms that are essential in describing the analysis. The correlation between the executing code and performance of a program is described in Section 4, as well as how this code is represented in vector format to capture program behavior. Section 5 describes the machine learning algorithms used to automatically detect phases using the code vectors. Section 6 describes how simulation points are picked from the phases, and the accuracy resulting from representing the entire program execution using the simulation points. Section 7 examines parameters that significantly influence the performance of the SimPoint algorithm in terms of accuracy and run-time. Section 8 examines prior work in phase analysis that uses machine learning. Ongoing and future work is described in Section 9 and our findings are summarized in Section 10.

2. The Application - Simulation

In this section we explain the tools modern computer architects use to evaluate designs and the methods we use to evaluate our solutions.

2.1 Background

Processor architecture research quantifies the effectiveness of a design by executing a program on a software model of the architecture design called an architecture simulator. It is difficult to accurately compare studies that provide results for different sets of programs. To set a standard in the community, the Standard Performance Evaluation Corporation (SPEC) was established to provide a collection of benchmarks to evaluate processor performance. In the same manner, the architecture simulator needs to have a common baseline. SimpleScalar (Burger and Austin, 1997) is a cycle level processor simulator that has become a standard model for architecture research.

2.1.1 SPEC CPU BENCHMARKS

The SPEC CPU2000 benchmark suite has 26 programs, of which 12 are integer programs (primary execution is of integer instructions) and 14 are floating-point programs (primary execution is of floating-point instructions). The benchmark suite is chosen to stress a processor across its many components in a rigorous manner. Each program in the suite has 3 different inputs: *test*, *train*, and *reference*, which respectively correspond to a short test, a more representative training, and a full reference run. The test, train and reference inputs typically execute on the order of a few million, a few billion, and hundreds of billions of instructions respectively. Tables 1 and 2 show all the SPEC CPU2000 benchmarks, divided into integer and floating-point programs. The tables provide a high level description of each benchmark, its source language, and the number of instructions executed (in billions) with the reference and test inputs. These programs were compiled for the Alpha Instruction Set Architecture (ISA) with full optimizations. On average, the reference inputs execute for 223 billion instructions. The program `parser` has the maximum instruction count at 546 billion instructions.

SPEC periodically releases a benchmark suite to evaluate current and future processors. To keep up with the ever increasing rate of processor speeds, SPEC has significantly increased the duration of benchmark execution from the previous suite release in 1995 to the current release of 2000. This is because the reference input needs to run long enough to achieve a valid timing for the benchmark run. This means that with current and future speeds that future releases of the SPEC benchmark suite will need to execute on the order of trillions of instructions for the reference inputs.

2.1.2 SIMPLESCALAR

SimpleScalar is a program that models the cycle level execution of a processor. It takes as input a program-input pair and simulates the execution from beginning to end, while computing relevant statistics for architecture research, such as cycles per instruction (CPI), cache miss rates, branch mispredictions, and power consumption. SimpleScalar has several models to represent different levels of execution detail. At the coarsest level of detail, *sim-fast* models only the functional execution of a program at the instruction level. A more detailed level, *sim-cache*, models the memory hierarchy and computes miss rates for those structures. The level of highest detail, *sim-outorder*, models the cycle-level out-of-order execution of a super-scalar processor. It is a superset of all the other mod-

Benchmark	Ref Length	Test Length	Language	Category
bzip2	143	8.82	C	Compression
crafty	191	4.26	C	Game Playing: Chess
eon	80	0.09	C++	Computer Visualization
gap	269	1.17	C	Group Theory, Interpreter
gcc	46	2.02	C	C Programming Language Compiler
gzip	84	3.37	C	Compression
mcf	61	0.26	C	Combinatorial Optimization
parser	546	4.20	C	Word Processing
perlbmk	111	2.0	C	PERL Programming Language
twolf	346	0.26	C	Place and Route Simulator
vortex	118	9.81	C	Object-oriented Database
vpr	84	0.69	C	FPGA Circuit placement and routing

Table 1: SPEC CPU2000 Integer Benchmarks (lengths in billions of instructions)

Benchmark	Ref Length	Test Length	Language	Category
ammp	326	5.49	C	Computational Chemistry
applu	223	0.18	Fortran 77	Parabolic / Elliptic Partial Differential Equations
apsi	347	5.28	Fortran 77	Meteorology: Pollutant Distribution
art	41	1.48	C	Image Recognition / Neural Networks
equake	131	1.44	C	Seismic Wave Propagation Simulation
facerec	211	4.12	Fortran 90	Image Processing: Face Recognition
fma3d	268	0.00	Fortran 90	Finite-element Crash Simulation
galgel	409	4.34	Fortran 90	Computational Fluid Dynamics
lucas	142	3.71	Fortran 90	Number Theory / Primality Testing
mesa	281	2.88	C	3-D Graphics Library
mgrid	419	16.77	Fortran 77	Multi-grid Solver: 3D Potential Field
sixtrack	470	8.59	Fortran 77	High Energy Nuclear Physics Accelerator Design
swim	225	0.43	Fortran 77	Shallow Water Modeling
wupwise	349	3.63	Fortran 77	Physics / Quantum Chromodynamics

Table 2: SPEC CPU2000 Floating-Point Benchmarks (lengths in billions of instructions)

I Cache	16k 2-way set-associative, 32 byte blocks, 1 cycle latency
D Cache	16k 4-way set-associative, 32 byte blocks, 2 cycle latency
L2 Cache	1Meg 4-way set-associative, 32 byte blocks, 20 cycle latency
Main Memory	150 cycle latency
Branch Pred	hybrid - 8-bit gshare w/ 8k 2-bit predictors + a 8k bimodal predictor
O-O-O Issue	out-of-order issue of up to 8 operations per cycle, 128 entry re-order buffer
Mem Disambig	load/store queue, loads may execute when all prior store addresses are known
Registers	32 integer, 32 floating point
Func Units	8-integer ALU, 4-load/store units, 2-FP adders, 2-integer MULT/DIV, 2-FP MULT/DIV
Virtual Mem	8K byte pages, 30 cycle fixed TLB miss latency after earlier-issued instructions complete

Table 3: Baseline Simulation Model.

els and provides the highest level of execution detail. The architecture research community uses SimpleScalar extensively, and today it is considered a standard architecture simulator.

The different models in SimpleScalar each have a stable execution rate. The fastest model, *sim-fast*, executes on the order of tens of billion instructions per hour on a 1 GHz machine. The slowest yet most accurate model, *sim-outorder*, executes on the order of hundreds of million instructions per hour, which is several orders of magnitude slower than the native hardware. It would take months of computation time to simulate the entire SPEC benchmark suite with *sim-outorder*. What makes matters worse is that researchers need to evaluate many different hardware configurations to measure the effectiveness of a design. This enormous turnaround time for a study makes simulating the full benchmark infeasible, and the majority of researchers only simulate a few hundred million instructions from each benchmark.

2.2 Methodology

For this study, we performed our analysis for the complete set of SPEC CPU2000 programs for multiple inputs using the Alpha binaries from the SimpleScalar website. We collect all of the frequency vector profiles, described in Section 4, using SimpleScalar. To generate our baseline results, we executed all programs from start to completion using SimpleScalar, gathering the hardware metrics. The baseline microarchitecture model is detailed in Table 3.

To examine the accuracy of our approach we provide results in terms of CPI prediction error and *k*-means variance (since SimPoint uses *k*-means clustering). The CPI prediction error is the percent difference between CPI predicted using only simulation points chosen by SimPoint and the baseline (true) CPI of the complete execution of the program. The *k*-means variance is the sum-of-squared distances between every clustered point and its closest center, which is the criterion *k*-means optimizes.

3. Defining Phase Behavior

Since phases are a way of describing the recurring behavior of a program executing over time, we begin by describing phase analysis with a demonstration of the time-varying behavior (Sherwood and Calder, 1999) of two programs from the SPEC 2000 benchmark suite, `gcc` and `gzip`. To characterize the behavior of these programs we have simulated their complete execution from start

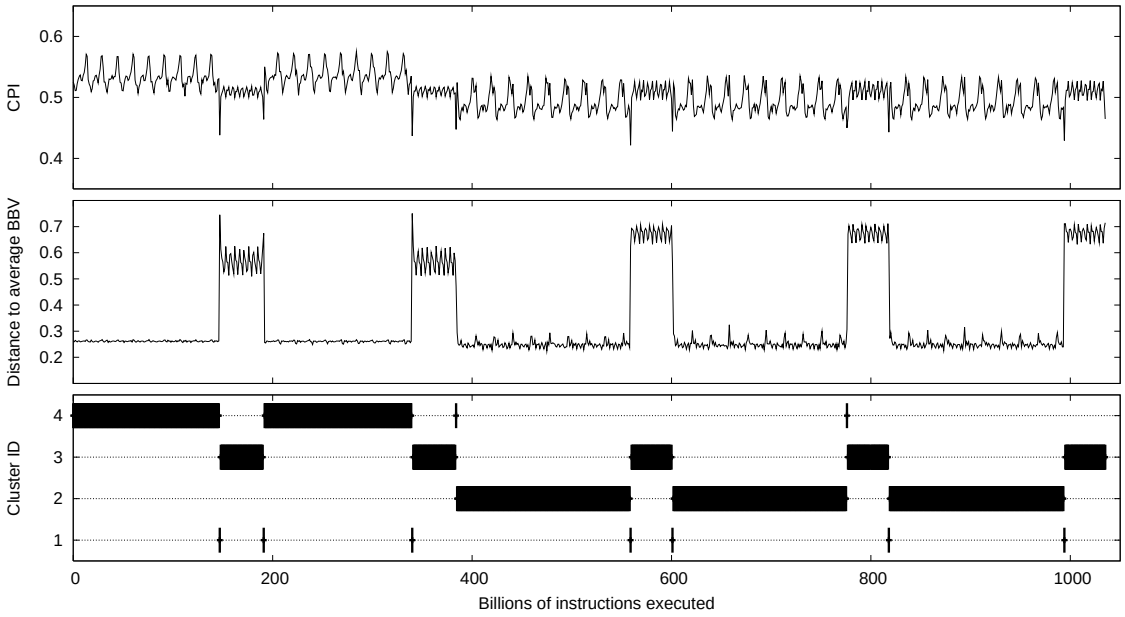


Figure 1: These plots show the relationship between measured performance (CPI) and code usage for the program `gzip-graphic`, and SimPoint's ability to capture phase information by only looking at what code is being executed. For each of the three plots, the horizontal axis represents the execution of the program over time, and each point plotted represents one 10-million instruction interval. The top plot shows the CPI for the executing program. The middle plot shows the distance of each interval's basic block vector (explained in Section 4) to the "target vector", which is a basic block vector that represents the entire program's execution. The target vector is a signature of the program's overall average behavior, and this plot shows how similar the code of each part of the program is to the overall behavior of the program, lower meaning more similar. The bottom plot shows how SimPoint classifies each interval into one of four phases. The phase transitions correspond to changes in the CPI in the top graph, though SimPoint does not use metrics like CPI to classify intervals.

to finish. Each program executes many billions of instructions, and gathering these results took several machine-months of simulation time. The behavior of each program is shown in the top graphs of Figures 1 and 2. Each top graph shows how the CPI rate changes for these two programs over time. CPI is a commonly used metric in the processor architecture community for measuring processor performance. Each point on the graph represents the average CPI taken over a window (we call it an interval) of 10 million executed instructions. These graphs show that programs are fairly complex, changing behaviors frequently.

Note that not only do the behaviors of the programs change over time, they change on the largest of time scales, and even at a large scale one can find repeating behaviors. Programs may have stable behavior for billions of instructions and then change suddenly. In addition to CPI, we have found for the SPEC 95 and 2000 programs that the behavior of *all* of the architecture metrics (branch prediction, cache misses, etc.) tend to change in unison, though not necessarily in the same

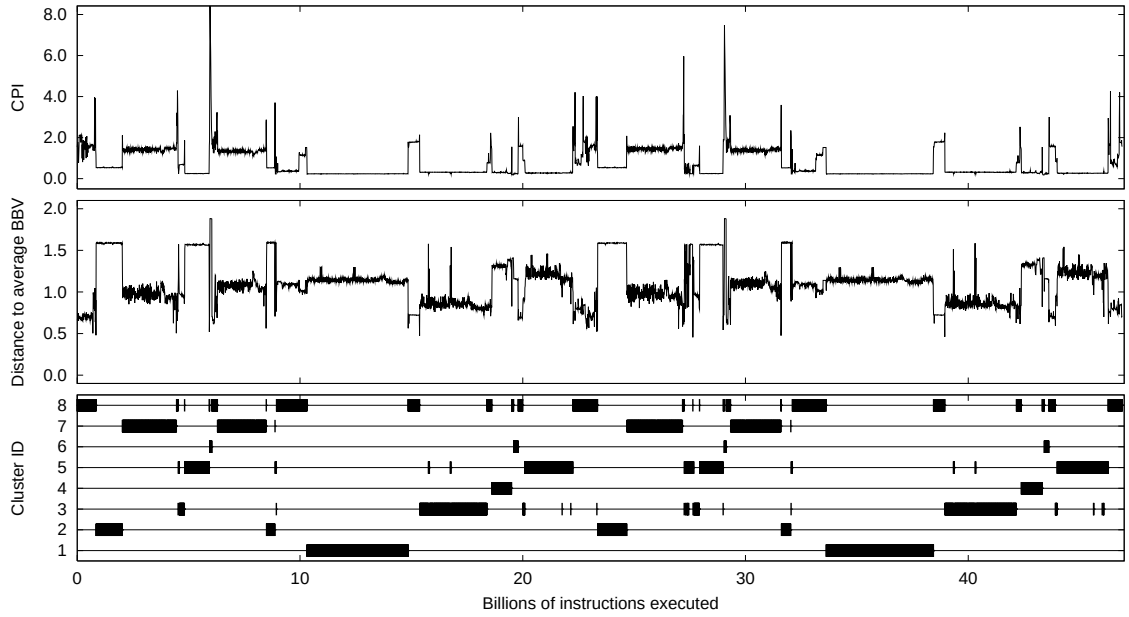


Figure 2: These plots show the relationship between measured performance (CPI) and code usage for the program `gcc-166`, and SimPoint's ability to capture phase information by only looking at what code is being executed. For each of the three plots, the horizontal axis represents the execution of the program over time, and each point plotted represents one 10-million instruction interval. The top plot shows the CPI for the executing program. The middle plot shows the distance of each interval's basic block vector to the "target vector", which is a basic block vector (explained in Section 4) that represents the entire program's execution. The target vector is a signature of the program's overall average behavior, and this plot shows how similar the code of each part of the program is to the overall behavior of the program, lower meaning more similar. The bottom plot shows how SimPoint classifies each interval into one of eight phases. The phase transitions correspond to changes in the CPI in the top graph, though SimPoint does not use metrics like CPI to classify intervals.

direction (Sherwood and Calder, 1999; Sherwood et al., 2002). These corresponding changes are due to underlying changes in program execution.

The underlying methodology used in this work is the ability to automatically identify these underlying program changes *without relying on architectural metrics*. To ground our discussion in a common vocabulary, the following is a list of definitions to describe program behavior and its automated classification.

- Interval – To perform our analysis we break a program's execution up into non-overlapping intervals of execution. An interval is a section of contiguous execution (a time slice) of a program's execution. For example, when using an interval size of 100 million instructions, the first interval of execution starts at instruction 0 and ends at the 100 million instruction executed, the second interval of execution are the instructions 100 million up to 200 million

in the program's execution, the third interval represents instructions 200 to 300 million, etc. For the results in this work all intervals are chosen to be the same length, as measured in the number of instructions committed within an interval. This is usually 1, 10, or 100 million instructions, as used by Perelman et al. (2003).

- Similarity – A similarity metric measures the similarity in behavior between two intervals of a program's execution, and is specific to the representation of those intervals.

- Phase – A set of intervals within a program's execution that all have similar behavior, *regardless* of temporal adjacency. A phase may be made up of intervals which are disjoint in time; we would call this a phase with a repeating behavior. A "well-formed" phase should have intervals with similar behavior across various architecture metrics (e.g. CPI, cache misses, branch misprediction). In this paper we consider the terms 'cluster' and 'phase' to be equivalent.

- Phase Classification – Using machine learning to group intervals from a program/input pair into phases (clusters) with similar behavior.

4. The Strong Correlation Between Code and Performance

In this section we describe how we identify phase behavior in an architecture independent fashion.

4.1 Using an Architecture-Independent Metric for Phase Classification

To find program phases, we need a notion of how similar are two different parts of a program's execution. In creating this metric it is advantageous to not rely on hardware-based statistics such as cache miss rates or performance (i.e. CPI), since using these would tie the phases to those statistics which change depending on the architecture configuration. If such statistics were used, the phases would need to be re-analyzed every time there was a change to some architectural parameter (either statically if the size of the cache changed, or dynamically if some policy changes adaptively). This is not acceptable, since our goal is to find a set of samples that can be used across an architecture design space exploration, where many of these parameters may change. To address this, we need a metric that is *independent* of any particular hardware-based statistic, but still relates to the fundamental changes in behavior like those shown in the top graphs of Figures 1 and 2.

An effective way to design such a metric is to base it on the behavior of a program in terms of the code that is executed over time. We have shown that there is a very strong correlation (Lau et al., 2005b) between the set of paths executed in a program and the time-varying architectural behavior observed. The intuition behind this is that the executed code determines the behavior of the program. With this idea it is possible to find the phases in programs using *only* a metric related to how the code is being exercised (i.e. both what code is touched and how often). The central idea behind SimPoint is that it can find the phase behavior shown in the top graphs of Figures 1 and 2 by examining only the frequency with which the code parts (e.g., basic blocks) are executed over time.

4.2 Basic Block Vector

The basic block vector (BBV) (Sherwood et al., 2001) is a structure designed to concisely capture information about how a program is changing behavior over time. A basic block is a section of

code (e.g. a contiguous set of instructions) that is executed from start to finish with one entry and one exit. The metric we will use for comparing two time intervals in a program is based on the differences in the execution frequencies for each basic block executed during those two intervals. The intuition behind this is that the behavior of the program at a given time is directly related to the code it is executing during that interval, and basic block vectors provide us with this information.

A program, when run for any interval of time, will execute each basic block a certain number of times. Knowing this information provides a code signature for that interval of execution, and shows where the application is spending its time in the code. The basic idea is that knowing the basic block distribution for two different intervals gives two separate signatures which we can then compare to find out how similar the intervals are to one another. If the signatures are similar, then the two intervals spend about the same amount of time in the same code, and the performance of those two intervals should be similar.

We represent a basic block vector as a one-dimensional array, with one element in the array for each static basic block in the program. Each interval in an executed program is represented by one BBV, and at the beginning of each interval, its corresponding BBV has all zeros. During each interval, we count the number of times each basic block has been entered, and record that number into the corresponding element in the vector. This number is weighted by the number of instructions in the basic block, since we want every individual instruction to have the same influence. Therefore, each element in the array is the count of how many times its corresponding basic block has been entered during an interval of execution, multiplied by the number of instructions in that basic block. For example, if the 50th basic block has one instruction and is executed 15 times in an interval, then bbv[50] = 15 for that interval. At the end of an interval's execution, we normalize the BBV to sum to 1.

We call the vectors used to guide phase analysis *Frequency Vectors*, of which basic block vectors are one type. Frequency vectors can represent basic blocks, branch edges, or any other type of program related structure which provides a representative summary of a program's behavior for each interval of execution. We recently examined frequency vector structures other than basic block vectors for the purpose of phase classification. We have looked at frequency vectors for data, loops, procedures, register usage, instruction mix, and memory behavior (Lau et al., 2004). We found that using register usage vectors, which simply counts for a given interval the number of times each register is defined and used, provides similar accuracy to using basic block vectors. In addition, using only loop and procedure branch execution frequencies performs almost as well as using the full basic block information. We also found, for SPEC 2000 programs, that creating frequency vectors by including both code and data access patterns into the vectors did not improve classification over just using code (Lau et al., 2004).

4.3 Basic Block Vector Difference

In order to find patterns in a program we must first have some way of comparing the similarity of two basic block vectors. The operation should take two basic block vectors and return a single number corresponding to how similar (or different) they are.

There are several ways of measuring the similarity of two vectors, such as taking the dot product between the vectors, finding the Euclidean (2-norm) distance of the connecting vector, or Manhattan (1-norm) distance of the connecting vector. The Euclidean distance has been shown to be effective for off-line phase analysis (Sherwood et al., 2002; Perelman et al., 2003). The SimPoint approach

we examine in this paper uses Euclidean distance as the metric for comparing basic block vectors, since it is based on *k*-means. For on-the-fly phase analysis (e.g. predicting phases during computation), the Manhattan distance is more efficiently implemented in hardware. It has been shown to be useful in previous work in online phase prediction (Sherwood et al., 2003; Lau et al., 2005c).

4.4 Showing the Correlation Between Code Signatures and Performance

For a detailed study showing that there is a strong correlation between executed code and real performance, please see Lau et al. (2005b). The top two graphs of Figure 2 give one illustration of this correlation by showing the time-varying CPI and BBV distance graphs next to each other for `gcc-166`. The top graph plots the CPI for each interval executed (at 10M interval length) showing how the program's CPI varies over time. Similarly, the BBV distance graph plots for each interval the Manhattan distance of the BBV (code signature) for that interval from the whole program's target vector. The whole program's target vector is a BBV that comes from viewing the whole program as a single interval. The same information is also provided for `gzip` in the top two graphs of Figure 1. These graphs show that changes in CPI have corresponding changes in code signatures, which is one indication of strong phase behavior for these applications.

These graphs show a strong correlation between code changes and CPI changes even for complex programs like `gcc`. The graphs for `gzip` show that phase behavior can be found even if the intervals' CPIs have small variance. This brings up an important point about classifying intervals based on code similarity rather than based on similarity of CPI or some other hardware metric. Assume we have two intervals with *different code signatures* but they have very *similar CPIs* because both of their working sets fit completely in the cache. During a design space exploration search, as the cache size changes, their CPIs may differ dramatically if one of them no longer fits into the cache. This is why it is important to perform the phase analysis by comparing the code signatures independent of the underlying architecture. We have found that the BBV code signatures correctly identify differences like these, which cannot be seen by looking at just the CPI.

4.5 Basic Block Similarity Matrix

Now that we have methods of comparing program execution intervals, we can use them for finding phase-based behavior. A phase of program behavior can be defined in several ways. Past definitions were built around the idea of a phase being a contiguous interval of execution during which a measured program metric is relatively stable. We extend this notion of a phase to include all similar sections of execution regardless of temporal adjacency. Thus, a phase may appear several times in the execution of a program.

A key observation from this paper is that the phase behavior seen in any program metric is a function of the code being executed. Because of this we can use the comparison between the basic block vectors to get an idea of how closely related any other metrics will be between those two intervals.

To find how all intervals of execution relate to one another we create a *basic block similarity matrix* for a program/input pair. The similarity matrix is an upper-triangular $n \times n$ matrix, where n is the number of intervals in the program's execution. An entry at (x,y) in the matrix represents the Manhattan distance between the basic block vector at interval x and the basic block vector at interval y.

Figures 3 (left and right) and 4 (left) shows the similarity matrices for `gzip`, `bzip`, and `gcc` using the Manhattan distance. The diagonal of the matrix represents the program's execution over time from start to completion. The darker the points, the more similar the intervals are (the Manhattan distance is closer to 0), and the lighter they are the more different they are (the Manhattan distance is closer to the maximum value — which is 2 since each vector is normalized to sum to 1).

Consider the points along the matrix diagonal. The top left corner of each matrix is the start of program execution $(0,0)$, and the bottom right is the point $(n-1,n-1)$ (end of execution). Each interval is perfectly similar to itself, so the points on the diagonal are all dark. Starting from a point on the diagonal, you can compare how its corresponding interval relates to its neighbors forward (backward) in execution by tracing horizontally (vertically) from that point. For example, to compare a given interval x with the interval at $x+m$, start at the point (x,x) on the matrix and trace to the right until you reach $(x,x+m)$.

Let us first examine `gzip` because it has behaviors that are evident at such a large scale that they are easy to see. An interval taken from 70 billion instructions into execution in Figure 3 (left) is directly in the middle of a large phase shown by the triangle of dark points that surround this point. This means that this interval is very similar to its neighbors both forward and backward in time. We can also see that the intervals at 50 billion and 90 billion instructions are also very similar to the program behavior at 70 billion instructions. While it may be hard to see in a printed version, the intervals around 70 billion instructions are similar to the intervals around 10 billion and 30 billion instructions, and even more similar to those around 50 and 90 billion instructions.

Overall, Figure 3 (left) shows that the phase behavior seen in the similarity matrix lines up quite closely with the behavior of the program seen in the top graph of Figure 1, with 5 large regions of self-similar behavior (the first 2 being different from the last 3) each divided by a small region of self-similar behavior. All of the small self-similar regions are also very similar to each other.

The similarity matrix for `bzip` (shown on the right of Figure 3) is very interesting. `Bzip` has complicated behavior, with two large parts to its execution: compression and decompression. This can readily be seen in the figure as the large dark triangular and square patches. The interesting thing about `bzip` is that even within each of these sections of execution there is complex behavior. This, as will be shown later, makes the behavior of `bzip` impossible to capture using only one small contiguous section of execution.

An even more complex case for finding phase behavior is `gcc`, which is shown on the left of Figure 4 (the matrix on the right of that figure will be explained in more detail in Section 5.1.1). The left matrix shows that `gcc` does have regular behavior. Even for such a complex program, we see that there is common code shared between sections of execution, such as the intervals around 13 billion instructions and 36 billion instructions. In fact the strong dark diagonal line cutting through the matrix indicates that there is large-scale repetition between the first half and second half of the program. By analyzing the graph we can see that code at each interval x is very similar to interval (x+23.6B instructions).

5. Automatically Finding Phase Behavior

In this section we describe the algorithms used to automatically detect patterns using the frequency vectors described in the previous section.

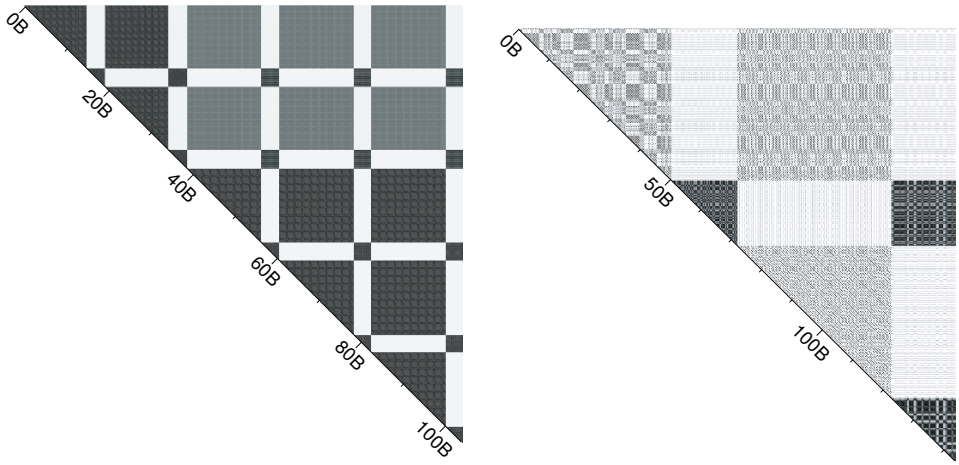


Figure 3: Basic block similarity matrix for the programs `gzip-graphic` (shown left) and `bzip-graphic` (shown right). The diagonal of the matrix represents the program's execution from beginning to end, with units in billions of instructions. The darker the points, the more similar the intervals are (the Manhattan distance is closer to 0), and the lighter the points the more different they are (the Manhattan distance is closer to 2).

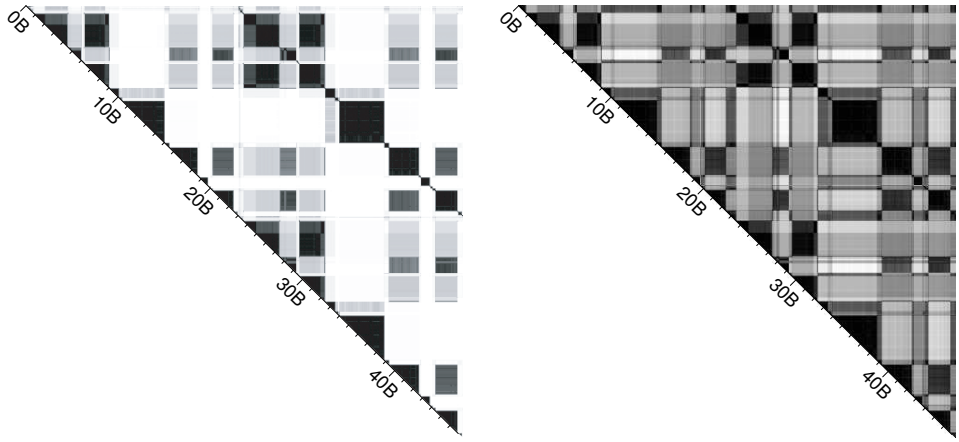


Figure 4: The original similarity matrix for the program `gcc-166` (left), and the similarity matrix for the projection of `gcc-166` (right). The figure on the left uses the original basic block vectors (each of which has over 100,000 dimensions), and uses the Manhattan distance for calculating the difference. The figure on the right uses the same data, but projected down to 15 dimensions, and uses the Euclidean distance for calculating the difference.

5.1 Using Clustering for Phase Classification

A primary goal of SimPoint is to have an automated way of extracting phase information from programs. Data clustering algorithms from unsupervised machine learning have been shown to be very effective at breaking the complete execution of a program into phases that have similar frequency vectors (Sherwood et al., 2002). Because the frequency vectors correlate to the overall performance of the program, grouping intervals based on their frequency vectors produces phases that are similar not only in the distribution of program structures used, but also in every other architecture metric measured, including overall performance.

The goal of clustering is to divide a set of points into clusters such that points within each cluster are similar to one another (by some metric), and points in different clusters are different from one another. We use the machine learning term 'cluster' and the architecture term 'phase' to express the same concept.

The *k*-means algorithm (MacQueen, 1967) is an efficient and well-known clustering algorithm, which we use to split program intervals into phases. Prior to clustering, we use random linear projection (Dasgupta, 2000) to reduce the dimension of the input vectors. One drawback of the *k*-means algorithm is that it requires the number of clusters k as an input to the algorithm, but we do not know beforehand what value is appropriate. To address this, we run the algorithm for several values of k, and then use a penalized likelihood score to guide our final choice for k. Taken to the extreme, if every interval of execution is given its very own cluster, then every cluster will have homogeneous behavior. Our goal is to choose a clustering with a minimum number of clusters which still models the program behavior well.

The following steps summarize the SimPoint phase clustering algorithm at a high level.

1. Profile the program by dividing the program's execution into contiguous intervals of fixed length (e.g., 1 million, 10 million, or 100 million instructions). For each interval, collect a frequency vector tracking the program's use of some program structure (basic blocks, branch edges, loops, register usage, etc.). Each frequency vector is normalized so that the sum of all the elements equals 1.

2. Reduce the dimensionality of the frequency vector data to a much smaller number of dimensions using random linear projection. Using projected data speeds up the *k*-means algorithm significantly and reduces the memory requirements by several orders of magnitude while preserving the essential similarity information.

3. Run the *k*-means clustering algorithm on the projected data with values of k in the range from 1 to K, where K is a user-prescribed maximum number of phases that can be detected. Each run of *k*-means produces a clustering, which is a partition of the data into k different phases/clusters. Each run of *k*-means begins with a random initialization step, which requires a random seed.

4. To compare and evaluate the different clusters formed for different k, we use the Bayesian Information Criterion (BIC) as a measure of the "goodness of fit" of a clustering to a data set. A high BIC score indicates the clustering is a good fit to the data. For each clustering ($k \in \{1, 2, \ldots, K\}$), the fitness of the clustering is scored using the BIC.

5. The final step is to choose the clustering with a small k such that its BIC score is nearly as good as the best observed. The chosen clustering is the final grouping of intervals into phases.

The above algorithm groups intervals into phases. This algorithm has several important parameters: interval length, projected dimension, the maximum number of clusters K, how the BIC is to be used to select the best clustering, etc. Each must be tuned to create accurate and representative simulation points using SimPoint. We discuss these parameters in more detail later in this paper.

5.1.1 RANDOM PROJECTION

For this clustering problem, we have to address the problem of high dimensionality. Many clustering algorithms suffer from the so-called "curse of dimensionality," which refers to the fact that finding an optimal clustering is intractable as the number of dimensions increases. One problem is that geometric optimizations that give significant speedup in low-dimensional data often have the opposite effect in high dimensions (e.g. k-d trees for speeding up nearest neighbor queries). For basic block vectors, the number of dimensions is the number of executed basic blocks in the program, which ranges from 2,756 to 102,038 for the SPEC benchmark suite, and could grow into the millions for very large programs. For example, one Microsoft application we studied consisted of over 800,000 basic blocks, which is representative of desktop applications. Another practical problem is that the running time and memory requirements of k-means depend on the dimension of the data, making the algorithm slow if the dimension grows too large. Also, we observe that k-means tends to get stuck easily in sub-optimal solutions if the dimension is too high. This is evidenced by the small number of iterations k-means requires to converge on high-dimensional data, as we have observed on this data. The algorithm does not improve much over its initialization.

Two broad methods of reducing the dimension of data are dimension selection and dimension reduction. Dimension selection simply removes some of the dimensions, based on a measure of goodness of each dimension for describing the data. However, this can throw away a lot of information in the dimensions which are ignored. Also, in finding a measure to select useful dimensions is not as clear for unsupervised learning as for supervised learning. Dimension reduction reduces the number of dimensions by creating a new lower-dimensional space and then projecting each data point into the new space (where the new space's dimensions are not necessarily related to the old space's dimensions).

For this work we use random linear projection (Dasgupta, 2000) to create a new low-dimensional space into which we orthogonally project the data. This is a simple and fast technique that is very effective at reducing the number of dimensions while retaining the essential structure of the data. There are two steps to projecting a data set down to a lower-dimensional version. Consider a data set X which is represented as a matrix of $n \times d$ real values, where n is the number of vectors, and d is the original dimension. We want a low-dimension version X' which is $n \times d'$, where d' is the projected number of dimensions. To create X', we do the following:

- Create a projection matrix P size $d \times d'$. Fill each entry in the matrix with a random value chosen uniformly in $[-1, 1]$.
- Use a matrix multiplication to obtain $X' = X \times P$.

The analysis given by Dasgupta (Dasgupta, 2000) shows that when using random linear projection for clustering data, there are two primary theoretical benefits. The first is that clusters that are very eccentric will become more spherical in their low-dimensional representation. This is appropriate for the k-means algorithm which searches for spherical clusters. The second is that a mixture of

k Gaussian clusters can be projected into only $O(\log k)$ dimensions while retaining the approximate level of separation between clusters.

Principal components analysis (PCA) is a widely-used method for dimension reduction based on directions of high variance. However, performing PCA on a d-dimensional data set requires $O(d^3)$ operations, which is too expensive for data sets of the size we are considering here that can have hundreds of thousands of dimensions. Constructing the random projection matrix requires only $O(dd')$ time, so it is linear in the original and the new dimension. Dasgupta further showed that there are many simple examples where PCA is not able to reliably reduce k well-separated Gaussian clusters to below $\Omega(k)$ dimensions and keep them well-separated in the low-dimensional projection. Examining the use of PCA for BBV dimension reduction is part of our future research.

For our application, we found that 15 dimensions is low enough to be computationally tractable, but sufficiently high to discover the different phases of execution with clustering. We found this by running experiments which are reported in earlier work (Sherwood et al., 2002). These experiments projected all the data sets we are interested in to a varying number of dimensions and then recorded the number of clusters found by k-means and the BIC. We found that for fewer than 15 dimensions, the number of clusters found dropped off, but for more than 15 dimensions, the number of clusters found did not increase significantly. Similar results were also found using the G-means algorithm to incrementally learn k (without using the BIC) by Hamerly and Elkan (2003). Section 7 evaluates how the choice of dimension affects the accuracy of SimPoint.

Figure 4 shows the similarity matrix for `gcc` on the left using original BBVs, whereas the similarity matrix on the right shows the same matrix but on the data that has been projected down to 15 dimensions. For the reduced dimension data we use the Euclidean distance to measure differences, rather than the Manhattan distance used on the original data. Some information is lost because of the projection, but overall phase behavior we see in the original data is still easily discernible with only 15 dimensions. A scatterplot of the program `gzip` projected to 2 dimensions and clustered into 3 clusters using k-means is shown in Figure 5.

5.1.2 BAYESIAN INFORMATION CRITERION

To compare the different clusterings formed for different k, we use the Bayesian Information Criterion, or BIC (Schwarz, 1978), as a measure of the "goodness of fit" of a clustering to a data set. The BIC is an approximation of the probability of the clustering, given the data that has been clustered. Thus, the larger the BIC score, the higher the probability that the clustering being scored is a "good fit" to the data being clustered. The BIC formulation we use is appropriate for clustering with k-means, however other formulations of the BIC could also be used for other clustering models. The BIC is only one method of choosing a good model from a set of models; other methods such as the Akaike information criterion (AIC) (Akaike, 1974), minimum description length (MDL) (Rissanen, 1978), and Monte-carlo cross-validation (MCCV) (Smyth, 1996) may also be appropriate.

There are two parts of the BIC: the likelihood and the penalty. The likelihood is a measure of how well the clustering models the data. For the k-means likelihood, each cluster's model is considered a spherical Gaussian distribution (which is the assumption k-means makes). The likelihood of a cluster is the product of the probabilities of each point in the cluster given by the cluster's Gaussian. The likelihood for the whole model is just the product of the likelihoods for all clusters. However, the likelihood tends to increase without bound as more clusters are added. Therefore the second term is a penalty that offsets the likelihood growth based on the model complexity (i.e. the

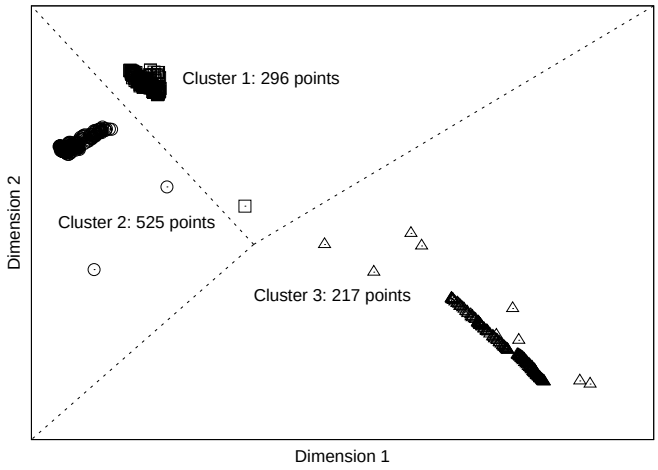


Figure 5: This plot shows a two-dimensional projection of the basic block vectors for the program `gzip`, having 1038 total intervals, and clustered into three clusters with *k*-means. The lines show divisions between the three clusters. Note that SimPoint normally operates in more than two dimensions, but this illustrates the fact that that program behavior does form natural groups that can be found through data clustering.

number of clusters). The BIC is formulated as

$$BIC(X,C_k) = \mathcal{L}(X|C_k) - \frac{p}{2}\log(n)$$

where $\mathcal{L}(X|C_k)$ is the log-likelihood of the clustered data X given the clustering C_k having k clusters, $n = |X|$ is the number of points in the data, and $p = (k-1) + dk + 1 = k(d+1)$ is the number of parameters to estimate: $(k-1)$ cluster probabilities, k cluster center estimates which each requires d mean estimates, and one variance estimate (shared over all clusters). The log-likelihood of the *k*-means model given the data is

$$\mathcal{L}(X|C_k) \quad = \quad -\frac{nd}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{k}\sum_{i\in C_j}||X_i - c_j||^2 + \sum_{j=1}^{k} n_j \log(n_j/n)$$

where n_j is the number of points in the jth cluster (so n_j/n is the estimated prior probability of cluster j), and σ^2 is the average squared Euclidean distance from each point to its cluster center. The term C_j represents the set of all indexes of X that are members of cluster j, X_i is the ith point in data set X, and $c_j = \frac{1}{n_j}\sum_{i\in C_j} X_i$ is the location of the jth cluster center. The center c_j is the maximum likelihood solution for the cluster's center. The maximum likelihood estimator for σ^2 is

$$\hat{\sigma}^2 \quad = \quad \frac{1}{nd}\sum_{j=1}^{k}\sum_{i\in C_j}||X_i - c_j||^2.$$

For the purposes of calculating the BIC, we can substitute this maximum likelihood estimate for σ^2 into the log-likelihood formulation, to get a simpler version:

$$\mathcal{L}(X|C_k) \quad = \quad -\frac{nd}{2}\log(2\pi\sigma^2) - \frac{nd}{2} + \sum_{j=1}^{k} n_j \log(n_j/n).$$

The BIC formulation we present basically follows that given by Pelleg and Moore (2000).

For a given program and inputs, the BIC score is calculated for each *k*-means clustering, for K in the range 1 to K. We then choose the clustering that achieves a BIC score that is close to the highest BIC score seen. This is explained more in Section 7.

5.2 Clusters and Phase Behavior

The bottom plots in Figures 1 and 2 show the results of running our phase-finding clustering algorithm on `gzip` and `gcc`. These results use an interval length of 10 million instructions and the maximum number of phases (K) is set to 10. The horizontal axis corresponds to the execution of the program (in billions of instructions), and each interval is classified to belong to one of the clusters (labeled on the vertical axis).

For `gzip`, the program's execution is partitioned into 4 clusters. Looking at the middle plot for comparison, the cluster behavior captured by our algorithm lines up quite closely with the behavior of the program. Clusters 2 and 4 represent the large sections of execution which are similar to one another. Cluster 3 captures the smaller phase that lies in between these larger phases. Cluster 1 represents the phase transitions between the three dominant phases. The intervals in cluster 1 are grouped into the same phase because they execute a similar combination of code, which happens to be part of the code behavior in either cluster 2 or 4 and part of code executed in cluster 3. These transition points in cluster 1 also correspond to the same intervals that have large spikes in CPI seen in the top graph (these spikes are due to increased cache misses for those regions).

The bottom plot of Figure 2 shows how `gcc` is partitioned into 8 clusters. Comparing this to the middle and top plots in the same figure, we see that even the more complicated behavior of `gcc` is captured well by SimPoint. The dominant behaviors in the top two graphs can be seen grouped together in phases 1, 3, 5,and 7.

6. Choosing Simulation Points from the Phase Classification

After the phase classification algorithm has done its job, intervals with similar code usage will be grouped together into the same phases (clusters). Then from each phase, SimPoint chooses one representative interval that will be simulated in detail to represent the behavior of the whole phase. Therefore, by simulating *only* one representative interval per phase, we can extrapolate and capture the behavior of the entire program.

To choose a representative for a cluster, SimPoint picks the interval that is closest (Euclidean distance) to the cluster's *k*-means center. The center can be viewed as a pseudo-interval which behaves most like the average behavior of the entire phase. Most likely there is no interval that exactly matches the center, so SimPoint chooses the closest interval. The selected interval is called a *simulation point* for that phase (Perelman et al., 2003; Sherwood et al., 2002). We can then perform detailed simulation on the set of simulation points.

As part of its output SimPoint also gives a weight for each simulation point. Each weight is a fraction: it is the total number of instructions represented by the intervals in the cluster from which

the simulation point was taken divided by the number of instructions in the program. With the weights and the detailed simulation results of each simulation point, we can compute a weighted average for the architecture metric of interest (CPI, cache miss rate, etc.) for the entire program's execution.

These simulation points are chosen once for a program/input combination because they are chosen based only on how the code is executed, and not based on architecture metrics. Therefore, they only need to be calculated once for a binary/input combination and can be used repeatedly across all of the runs for an architecture design space exploration.

The number of simulation points that SimPoint chooses has a direct effect on the simulation time that will be required for those points. The maximum number of clusters, K, along with the interval length, represents the maximum amount of simulation time that will be needed. When fixed length intervals are used, $(K * \text{interval length})$ is a limit on the number of simulated instructions.

SimPoint allows users to trade off simulation time with accuracy. Researchers in architecture tend to want to keep simulation time to below a fixed number of instructions (e.g., 300 million) for a run. If this is a goal, we find that an interval length of 10 million instructions with $K = 30$ provides very good accuracy (as we show in this paper) with reasonable simulation time (220 million instructions on average). If even more accuracy is desired, then decreasing the interval length to 1 million and setting $K = 300$ performs well for the SPEC 2000 programs, as does setting $K = \sqrt{n}$ (where n is the number of clustered intervals). Empirically we discovered that as the granularity becomes finer, the number of phases discovered increases at a sub-linear rate. The upper bound defined by this square-root heuristic works well for the SPEC benchmarks.

The length of the interval chosen by users of SimPoint depends upon their simulation infrastructure and how much they want to deal with warmup. Warmup is the process of initializing the simulator's state (caches, branch predictor, etc.) at the start of a simulation point so that it is the same as if we simulated from the beginning of the program to that point. For many programs, using a long interval length (e.g., more than 100 million instructions) will make warmup unnecessary. This is the approach used by Intel's PinPoint for simulation (Patil et al., 2004). They simulate intervals of length 300-500 million instructions so they do not have to worry about implementing warmup in their simulation infrastructure. With such long intervals the architecture structures are warmed up sufficiently during the beginning of the interval's execution to provide accurate simulation results. In comparison, short interval lengths can be used, but this requires having an approach for warming up the architecture state. One way to do this is with an architecture checkpoint, which stores the potential contents of the major architecture components at the start of the simulation point (Biesbrouck et al., 2005). This can significantly reduce warmup time, since warmup consists of just reading the checkpoint from a file and using it to initialize the architecture structures.

6.1 Accuracy of SimPoint

We now show the accuracy of using SimPoint for the complete SPEC 2000 benchmark suite and their reference inputs. Figure 6 shows the simulation accuracy results using SimPoint (and other methods) for the SPEC 2000 programs when compared to the complete execution of the programs. For these results we use an interval length of 100 million instructions and limit the number of simulation points to no more than 10. With the above parameters SimPoint finds 4 phases for `gzip`, and 8 for `gcc`. As described above, one simulation point is chosen for each cluster, so this means

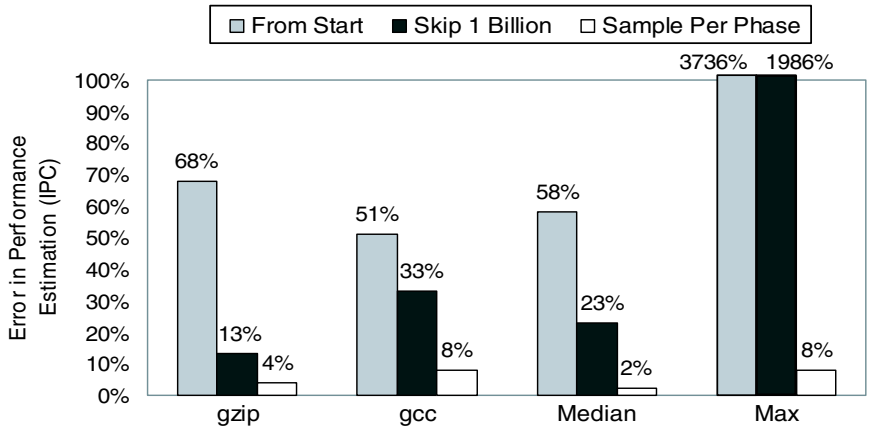


Figure 6: Simulation accuracy for the SPEC 2000 benchmark suite when performing detailed simulation for several hundred million instructions compared to simulating the entire execution of the program. Results are shown for simulating from the start of the program's execution, for fast-forwarding 1 billion instructions before simulating, and for using SimPoint to choose at most ten 100-million-instruction intervals to simulate. The results are shown as percent error of predicted IPC, which is how much the estimated IPC using SimPoint is different from the complete execution of the program. IPC is the inverse of CPI. The median and maximum results are for the complete SPEC 2000 benchmarks.

that a total of 400 million instructions were simulated for `gzip`. The results show that this results in only a 4% error in performance estimation for `gzip`.

For these results, we compare this estimated IPC using SimPoint to the baseline IPC. IPC (Instructions Per Cycle) is the inverse of CPI, and often used instead of CPI when describing performance. The baseline was gathered from spending months of simulation time to simulate the entire execution of each SPEC program. The results in Figure 6 compare SimPoint to how architecture researchers use to choose where to simulate before SimPoint. The first technique was to just simulate the first N million instructions of a benchmark's execution. The second technique was to blindly skip the first billion instructions of execution to get past the initialization of the program's execution, and then simulate for N million instructions. The results show that simulating from the start of execution, for the exact same number of instructions as simulated with SimPoint, results in a median error of 58%. If instead, we fast forwarded for 1 billion instructions and then simulate for the same number of instructions as chosen by SimPoint, we see a median 23% IPC error. When using SimPoint to create multiple simulation points we have a median IPC error of 2%. Note that the maximum error seen for the prior techniques are significant for the SPEC programs, but it is very reasonable (only 8%) for SimPoint.

6.2 Relative Error During Design Space Exploration

The absolute error of a program/input run on one hardware configuration is not as important as tracking the change in metrics across different architecture configurations. There is a lot of discussion and research into getting lower simulation error rates. But what often is not discussed is that a

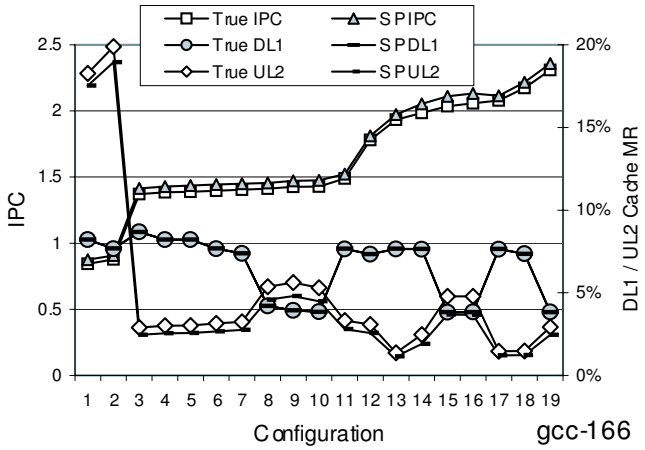


Figure 7: This plot shows the true and estimated IPC and cache miss rates for 19 different architecture configurations for the program `gcc`. The left *y*-axis is for the IPC and the right *y*-axis is for the cache miss rates for the L1 data cache and unified L2 cache. Results are shown for the complete execution of the configuration and when using SimPoint.

low error rate for a single configuration is not as important as achieving the same relative error rates across the design space search and having them all biased in the same direction.

We now examine how SimPoint tracks the relative change in hardware metrics across several different architecture configurations. To examine the independence of the simulation points from the underlying architecture, we used the simulation points for the SimPoint algorithm with an interval length of 1 million instructions and the maximum K set to 300. For the program/input runs examined, we performed full program simulations while varying the memory hierarchy, and for every run we used the same set of simulation points when calculating the SimPoint estimates. We varied the configurations and the latencies of the L1 and L2 caches as described by Perelman et al. (2003).

Figure 7 shows the results across 19 different architecture configurations for `gcc-166`. The left *y*-axis represents the performance in Instructions Per Cycle (IPC) and the *x*-axis represents different memory configurations from the baseline architecture. The right *y*-axis shows the miss rates for the data cache and unified L2 cache, and the L2 miss rate is a local miss rate. For each metric, two lines are shown: "True" for the true metric from the *complete* detailed simulation, and the "SP" for the estimated metric using our simulation points. For the results, the configurations on the *x*-axis are sorted by the IPC of the full run.

This figure shows that the simulation points, which are chosen by only looking at code usage, can be used across different architecture configurations to make accurate architecture design trade-off decisions and comparisons. The simulation points are able to track the relative changes in performance metrics between configurations. This means we are able to make the same decision between two architectures, in terms of which one is better, using SimPoint as the complete simulation of the program. One interesting observation is that although the simulation results from SimPoint have a bias in its predictions, this bias is consistent across the different configurations for a given

program/input. This is true for both IPC and cache miss rates. We believe one reason for the bias is that SimPoint chooses the most representative interval from each phase, and intervals that represent phase change boundaries are less likely to be fully represented across the chosen simulation points.

7. Clustering Analysis

In this section we describe the primary parameters that have influence on how SimPoint and the *k*-means algorithm behave. We first focus on how we achieve a reasonable running time for *k*-means, and then examine how to search over *k* to find a good clustering. For the experiments in this section, we use basic block vectors with 100 million instruction intervals. Where it is not specified, we also use $k = 30$ clusters and 15 projected dimensions.

7.1 Methods for Reducing the Run-Time of *k*-Means

Even though SimPoint only needs to be run once per binary/input combination, we still want a fast clustering algorithm that produces accurate simulation points. To address the run-time of SimPoint, we first look at the three parts which affect most the running time of a single run of *k*-means. The three parts are the number of intervals to cluster, the dimension of the intervals being clustered, and the number of iterations it takes to perform a clustering.

We first examine how the number of intervals affects the running time of the SimPoint algorithm. Figure 8 shows the time (in seconds) for running SimPoint on different numbers of intervals as we vary the number of clusters. For this experiment, the clustered vectors are randomly generated from uniformly random noise in 15 dimensions. We use random data in these experiments because it does not bias these results based on a particular benchmark and it gives comparable results across a wide range of parameter settings. But more importantly, prior theoretical work by Indyk et al. (1999) suggests that it is most difficult to accelerate (i.e. make more efficient using geometric reasoning) clustering algorithms on data without structure, such as uniformly random data. This is supported by experiments by Moore (2000) and Elkan (2003). So these experiments form a comparable set of challenging results for the per-iteration run-time of SimPoint. The number of iterations will vary depending on the structure of the data, however. For example, using *k*-means to cluster data from very well-separated clusters is likely to converge in a low number of iterations, while clusters which overlap are likely to require more iterations.

The first graph shows that for 100,000 vectors and $k = 128$, it took about 3.5 minutes for SimPoint 3.0 to perform the clustering. It is clear that the number of vectors clustered and the value of *k* both have a large effect on the run-time of SimPoint. The run-time changes linearly with the number of clusters and the number of vectors, as expected. Also, we can see that the time per basic operation actually goes down as *k* increases. This is due to a simple optimization called *partial distance search* (McNames, 2000; Cheng et al., 1984) that allows the algorithm to avoid calculating the full distance from a point (interval) to every cluster center in the first step of *k*-means. The goal of this step is to find the closest cluster center to the point, so that the interval may be assigned to that center. To find this closest center, a simple loop searches for the cluster center with the minimum squared Euclidean distance. The squared distance calculation sums the squared dimension difference between the point and the cluster center over all dimensions. While searching for the minimum squared distance from a point to all centers, partial distance search keeps the smallest squared distance seen thus far. When calculating the distance to another center, it may find that the intermediate squared distance result (after processing some of the dimensions) is larger than the

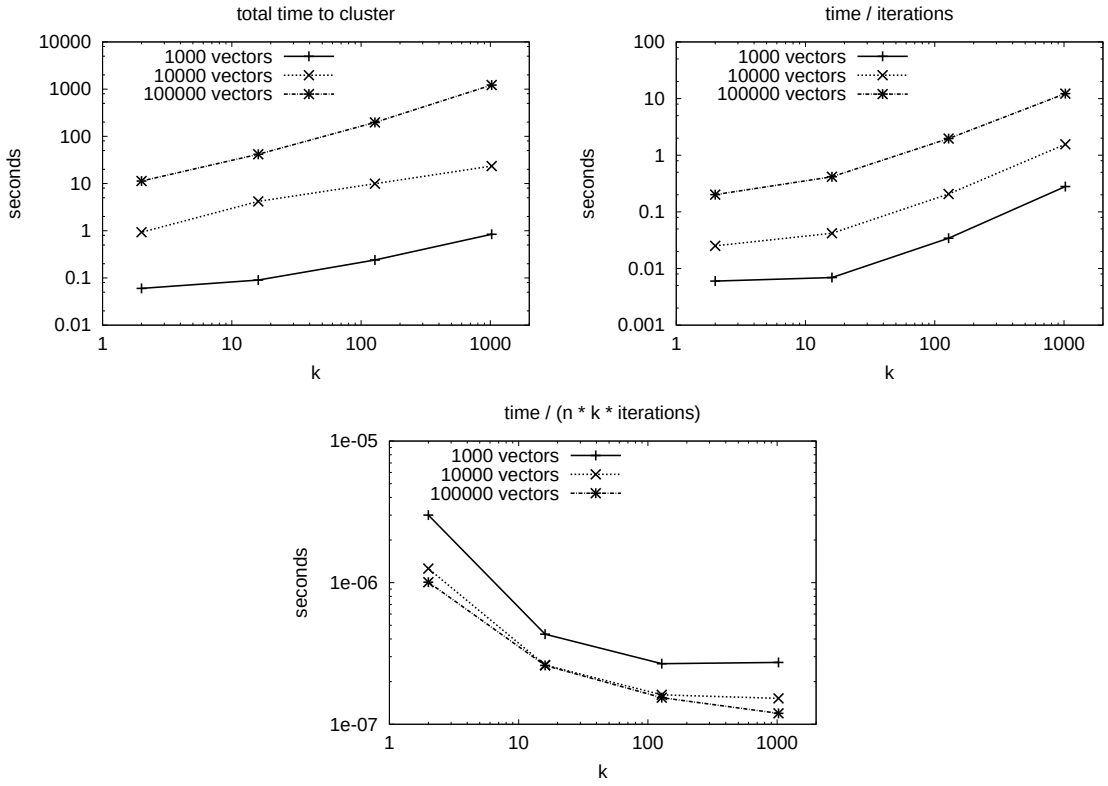


Figure 8: These plots show how varying the number of vectors and clusters affects the amount of time required to cluster with SimPoint 3.0. For this experiment we generated uniformly random data in 15 dimensions. The first plot shows total time, the second plot shows the time normalized by the number of iterations performed, and the third plot shows the time normalized by the number of basic operations performed. Both the number of vectors and the number of clusters have a linear influence on the run-time of *k*-means. The bottom plot shows a decreasing trend due to optimizations in *k*-means which are more beneficial for larger *k*.

smallest squared distance seen to a different center. If this is the case, the distance we are calculating cannot be minimal, so the current calculation is stopped short of calculating the entire squared distance over all of the dimensions. This optimization does not change the correctness of the algorithm. Partial distance search is most beneficial when there are many clusters, since the more centers there are, the more it is likely that there will be a close center that can give a good lower bound for the partial search. Partial distance search is also useful in high dimensional data, since work is saved when computing per-dimension differences, and the more dimensions there are the more computations can potentially be avoided.

Program	# Vecs × # B.B.	SP3-All	SP3-BinS
gcc-166	4692 × 102038	9 min	3.5 min
crafty	19189 × 16970	84 min	10.7 min

Table 4: This table shows the running times (in minutes) by SimPoint 3.0 without using binary search (SP3-All) and SimPoint 3.0 using binary search (SP3-BinS). SimPoint is run searching for the best clustering from k=1 to 100, uses 5 random seeds per k, and projects the vectors to 15 dimensions. The second column shows how many vectors and the size of the vector (static basic blocks) the programs have.

7.1.1 NUMBER OF INTERVALS AND SUB-SAMPLING

Each iteration of the k-means algorithm has a run-time that is linear in the number of clusters, the number of intervals, and the dimensionality. However, since k-means is an iterative algorithm, many iterations may be required to reach convergence. We already found in prior work (Sherwood et al., 2002), and revisit in Section 7.1.2 that we can reduce the number of dimensions down to 15 and still maintain SimPoint's accuracy. Therefore, the main influence on execution time for SimPoint is the number of intervals.

To show this effect, Table 4 shows the SimPoint running time for `gcc-166` and `crafty-ref`, which shows the lower and upper limits for the number of intervals and basic block vectors seen in SPEC 2000 with an interval length of 10 million instructions. The second and third columns show the number of intervals and original number of dimensions for each basic block vector. The last two columns show the time it took to execute SimPoint 3.0 searching for the best clustering from k=1 to 100, with 5 random initializations (seeds) per k. The fourth column shows the time it took to run SimPoint when searching over all k, and the last column shows clustering time when using the new binary search described in Section 7.2.3. The results show that increasing the number of intervals by 4 times increased the running time of SimPoint around 10 times. The results also show that the number of intervals clustered has a large impact on the running time of SimPoint, since it can take many iterations to converge, which is the case for `crafty`. We used 15 dimensions during clustering for these results.

The effect of the number of intervals on the running time of SimPoint becomes critical when using very small interval lengths like 1 million instructions or fewer, which can create millions of intervals to cluster. To speed the execution of SimPoint on these very large inputs, we sub-sample the set of intervals that will be clustered, and run k-means on only this sample. To sample with SimPoint, the user specifies the number of desired interval samples, and then SimPoint chooses that many intervals (without replacement). The probability of each interval being chosen is proportional to the weight of its interval (the number of dynamically executed instructions it represents). For vectors which all represent the same interval length (as we consider in this paper), this weight is uniform. If vectors represent non-uniform interval lengths (called variable-length intervals, or VLIs), then each vector's weight is proportional to its interval length. We summarize our work with variable length intervals in Section 9.

Sampling is common in clustering for data sets which are too large to fit in main memory (Farnstrom et al., 2000; Provost and Kolluri, 1999). After clustering the data set sample, we have a set of clusters with centers found by k-means. SimPoint then makes a single pass through the unclustered

intervals and assigns each interval to the cluster that has the nearest center (centroid) to that interval. This then represents the final clustering from which the simulation points are chosen. We originally examined using sub-sampling for variable length intervals (VLI) in Lau et al. (2005a). When using VLIs we had millions of intervals, and had to sub-sample 10,000 to 100,000 intervals for the clustering to achieve a reasonable running time for SimPoint, while still providing very accurate simulation points.

The experiments shown in Figure 9 show the effects of sub-sampling across all the SPEC 2000 benchmarks using an interval length of 10 million instructions, 30 clusters, and 15 projected dimensions. Results are shown for creating the initial clustering using sub-sampling with only 1/8, 1/4, 1/2, and all of the execution intervals in each program, as described above. The first two plots show the effects of sub-sampling on the CPI errors and *k*-means variance, both of which degrade gracefully when smaller samples are used. The average SPEC INT (integer) and SPEC FP (floating point) average results are shown. It is standard to break the results into these two groupings for architecture results. The CPI error is computed in the following manner:

$$\text{CPI Error} = \frac{|\text{True CPI} - \text{SimPoint Estimated CPI}|}{\text{True CPI}}.$$

The average *k*-means variance is the average squared distance between every frequency vector and its closest cluster center. Lower variances are better. When sub-sampling, we still report the variance based on every vector (not just the sub-sampled ones). The *relative k*-means variance reported in the experiments is measured on a per-input basis as the ratio of the *k*-means variance for clustering on a sample to that of clustering on the whole input.

As shown in the second graph of Figure 9, sub-sampling a program can cause *k*-means to find a slightly less representative clustering, which results in higher *k*-means variance on average. Note that the *k*-means variance for these experiments are reported on all the input vectors, not just the sampled ones. Even so, when sub-sampling, we found in some cases that it can reduce the *k*-means variance and/or CPI error (compared to using all the vectors), because sub-sampling can remove outliers in the data set that *k*-means may be trying to fit. This is a benefit noted in the work of Fayyad et al. (1998) when they use subsampling to initialize iterative clustering algorithms.

It is interesting to note the difference between floating point and integer programs, as shown in the first two plots. The results shown in the first plot show we can capture the behavior of the SPEC floating point programs more easily, that is, without using all the original data. In addition, the second plot suggests that SPEC floating point programs are also easier to cluster than the SPEC INT, as we can do quite well (in terms of *k*-means variance) even with only small samples. This suggests that they have more regular or uniform code usage patterns than integer programs. The third plot shows the effect of the number of vectors on the running time of SimPoint. This plot shows the time required to cluster all of the benchmark/input combinations and their 3 sub-sampled versions. In addition, we have fit a logarithmic curve with least-squares to the points to give a rough idea of the growth of the run-time. Note that two different data sets with the same number of vectors may require different amounts of time to cluster due to the number of *k*-means iterations required for the clustering to converge.

7.1.2 NUMBER OF DIMENSIONS AND RANDOM PROJECTION

Along with the number of vectors, the other most influential aspect in the running time of *k*-means is the number of dimensions of the data. Figure 10 shows the effect of changing the number of pro-

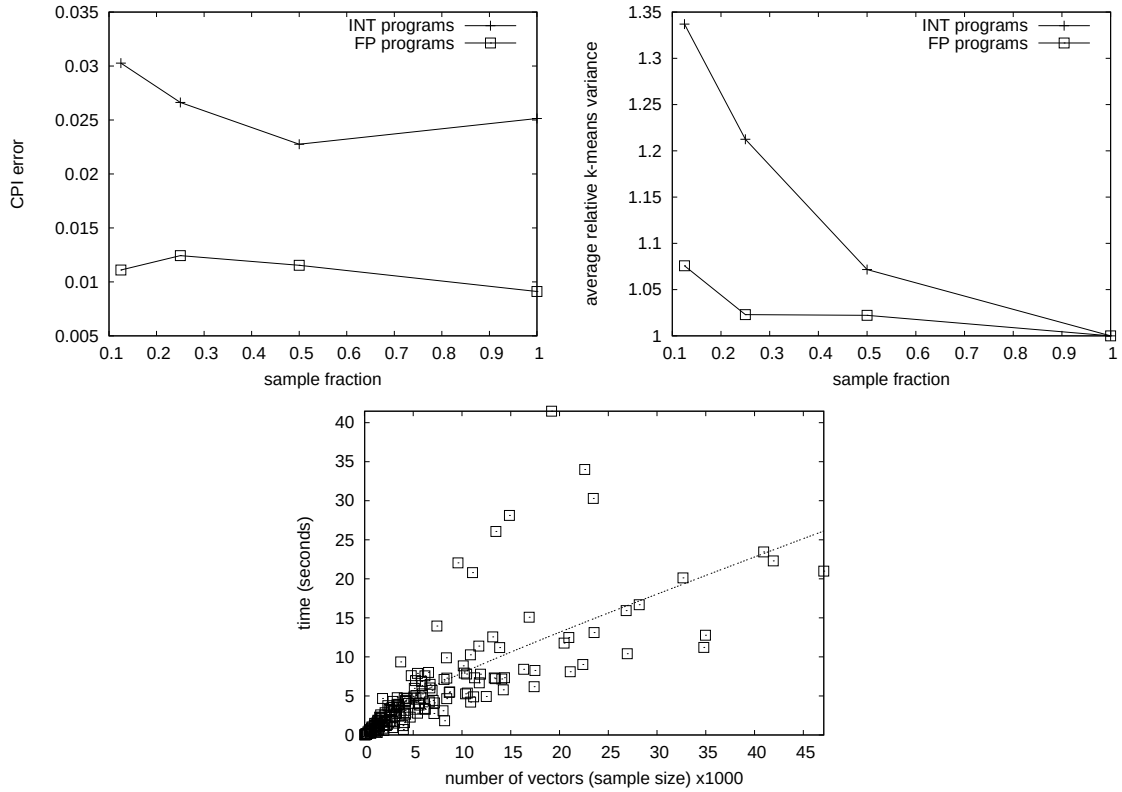


Figure 9: These three plots show how sub-sampling before clustering affects the CPI errors, *k*-means variance, and the run-time of SimPoint. The first plot shows the average CPI error across the integer and floating-point SPEC benchmarks. The second plot shows the average *k*-means clustering variance relative to clustering with all the vectors. The last plot shows a scatter plot of the run-time to cluster the full benchmarks and sub-sampled versions, and a logarithmic curve fit with least squares.

jected dimensions on both the CPI error (left) and the run-time of SimPoint (right). For this experiment, we varied the number of projected dimensions from 1 to 100. As the number of dimensions increases, the time to cluster the vectors increases linearly, as expected. It is more interesting that the run-time also increases for very low dimensions. This is because the points are more "crowded" and the clusters are less well-separated, so *k*-means requires more iterations to converge.

If we use too few dimensions, the data does not retain sufficient information to cluster the data well. This is reflected by the fact that the CPI errors increase rapidly for very low dimensions. However, we can see that at 15 dimensions, the SimPoint default, the CPI errors are quite low, and using a higher number of dimensions does not improve them significantly but requires more computation. Using too many dimensions is also a problem in light of the well-known "curse of dimensionality" (Bellman, 1961), which implies that as the number of dimensions increases, the number of vectors that would be required to densely populate that space grows exponentially. This means that using a higher dimension makes it more likely that a clustering algorithm will converge

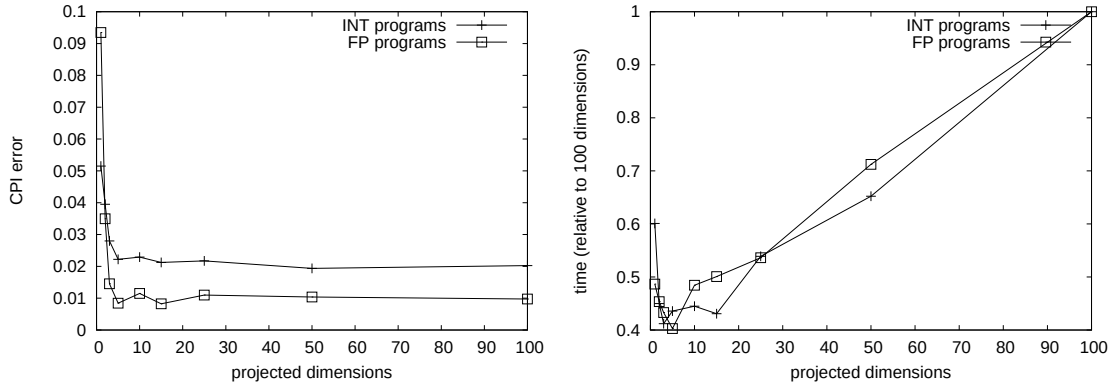


Figure 10: These two plots show the effects of changing the number of projected dimensions when using SimPoint. The default number of projected dimensions SimPoint uses is 15, but here we show results for 1 to 100 dimensions. The left plot shows the average CPI error, and the right plot shows the average time relative to 100 dimensions. Both plots are averaged over all the SPEC 2000 benchmarks, for a fixed $k = 30$ clusters.

to a poor solution, since the input space is not very densely filled. Therefore, it is wise to choose a dimension that is low enough to allow *k*-means to find a good clustering, but not so low that critical information is lost. We find that 15 dimensions works well in these regards.

7.1.3 NUMBER OF ITERATIONS NEEDED

The final aspect we examine for affecting the running time of the *k*-means algorithm is the number of iterations it takes for a run to converge. We provide this analysis to illustrate typical requirements of running SimPoint on a set of benchmarks, and because finding a tight upper-bound on the number of iterations required by *k*-means is an open problem (Dasgupta, 2003), we must rely on evidence to show us what to expect.

The *k*-means algorithm iterates either until it hits a user-specified maximum number of iterations, or until it reaches convergence. In SimPoint, the default limit is 100 iterations, but this can easily be changed. More iterations may be required, especially if the number of intervals is very large compared to the number of clusters. The interaction between the number of intervals and the number of iterations required is the reason for the large SimPoint running time for `crafty-ref` in Table 4.

For our results, we observed that only 1.1% of all runs on all SPEC 2000 benchmarks reach 100 iterations. This experiment was with 10-million instruction intervals, *k*=30, 15 dimensions, and with 10 random initializations of *k*-means. Figure 11 shows the number of iterations required for all runs in this experiment. Out of all of the SPEC program and input combinations run, only `crafty-ref`, `gzip-program`, `perlbmk-splitmail` had runs that had not converged by 100 iterations. The longest-running clusterings for these programs reached convergence in 160, 126, and 101 iterations, respectively. If desired, SimPoint can always run *k*-means to convergence (with no iteration limit).

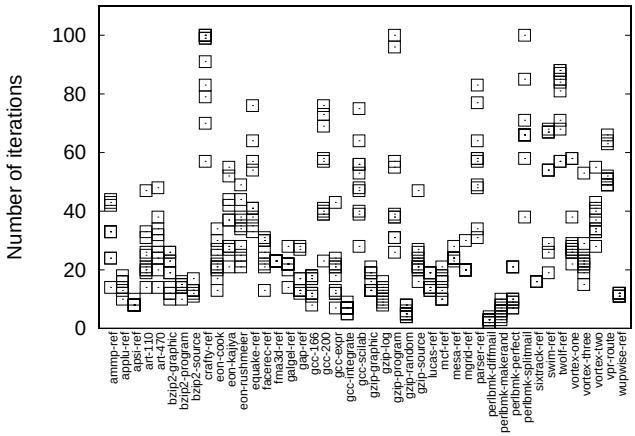


Figure 11: This plot shows the number of iterations required for 10 randomized initializations of each benchmark, with 10 million instruction length intervals, $k = 30$, and 15 dimensions. Note that only three program/inputs had a total of 5 runs that required more than the default limit of 100 iterations, and these all converge within 160 iterations or less.

7.2 Searching for a Small k with a Good Clustering

We suggest setting the maximum number of clusters K as appropriate for the maximum amount of simulation time a user will tolerate for a single simulation. SimPoint uses three techniques to search over the possible clusterings, which we describe here. The goal is to try to pick a small k so that the number of simulation points is also small, thereby reducing the simulation time required.

7.2.1 SETTING THE BIC PERCENTAGE

As we examine several clusterings and values of k, we need to have a method for choosing the best clustering. The Bayesian Information Criterion (BIC) (Pelleg and Moore, 2000) gives a score of the how well a clustering represents the data it clustered. However, we have observed that the BIC score often increases as the number of clusters increase. Thus choosing the clustering with the highest BIC score can lead to often selecting the clustering with the most clusters. Therefore, we look at the range of BIC scores, and select the score which attains some high percentage of this range. The SimPoint default BIC threshold is 90%. When the BIC rises and then levels off as k increases, this method chooses a clustering with the fewest clusters that is near the maximum BIC value. Choosing a lower BIC threshold would prefer fewer clusters, but at the risk of less accurate simulation.

Figure 12 shows the effect of changing the BIC threshold on both the CPI error (left) and the number of simulation points chosen (right). These experiments are for using binary search (explained in Section 7.2.3) with $K = 30$, 15 dimensions, and 5 random seeds. BIC thresholds of 70%, 80%, 90% and 100% are examined. As the BIC threshold decreases, the average number of simulation points decreases, and similarly the average CPI error increases. At the 70% BIC threshold, `perlbmk-splitmail` has the maximum CPI error in the SPEC suite. This anomaly is an artifact

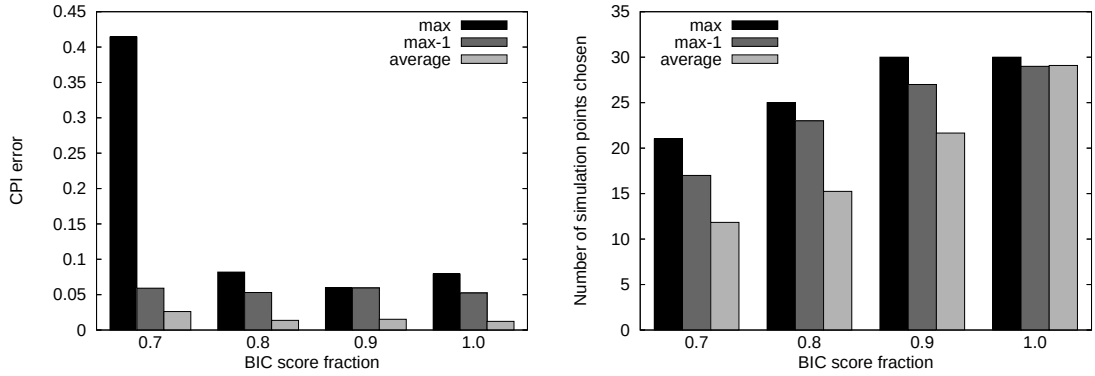


Figure 12: These plots show how the CPI error and number of simulation points chosen are affected by varying the BIC threshold. Bars labeled "max-1" show the second largest value observed.

of the low threshold. Since higher BIC scores point to better clusterings and better error rates, we recommend the BIC threshold to be set at 90%.

7.2.2 VARYING THE NUMBER OF RANDOM SEEDS, AND *k*-MEANS INITIALIZATION

The *k*-means clustering algorithm starts from a randomized initialization, which requires a random seed. Because of this, running *k*-means multiple times can produce very different results depending on the initializations, so *k*-means can sometimes converge to a locally-good solution that is poor compared to the best clustering on the same data for that number of clusters. Therefore, conventional wisdom suggests that it is good to run *k*-means several times using a different randomized starting point each time, and take the best clustering observed, based on the *k*-means variance or the BIC. SimPoint does this, using different random seeds to initialize *k*-means each time. Based on our experience, we have found that using 5 random seeds works well.

SimPoint allows users to provide their own *k*-means initialization, or it will choose an initialization based on one of two methods: sampling and furthest-first (Gonzalez, 1985; Hochbaum and Shmoys, 1985). The sampling method chooses *k* random locations for the initial cluster centers from the input data without replacement. The furthest-first method chooses one input point at random, and then repeatedly chooses a point that is furthest away from all the already-chosen points, until *k* points are chosen. This has the tendency to spread the initially chosen points out along the convex hull of the input space, and subsequently chosen points in the interior.

Figure 13 shows the effect on CPI error of using two different *k*-means initialization methods (furthest-first and sampling) along with different numbers of initial *k*-means seeds. These experiments are for using binary search with $K = 30$, 15 dimensions, and a BIC threshold of 90%. When multiple seeds are used, SimPoint runs *k*-means multiple times with different starting conditions and takes the best result.

Based on these results we see that sampling outperforms furthest-first *k*-means initialization. This can be attributed to the data we are clustering, which can have a large number of outlying points, which furthest-first initialization pays special attention to. The furthest-first method is likely

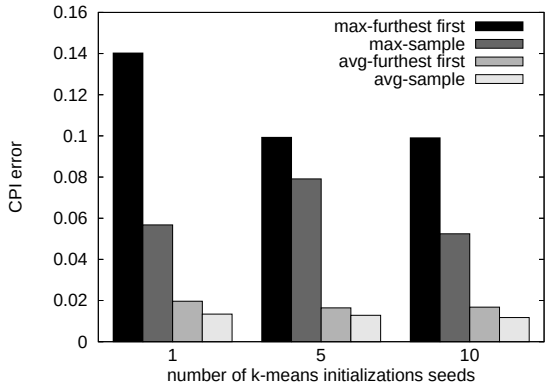


Figure 13: This plot shows the average and maximum CPI errors for both sampling and furthest-first *k*-means initializations, and using 1, 5, or 10 different random seeds. These results are over the SPEC 2000 benchmark suite for 10-million instruction vectors, 15 dimensions, and *k* = 30.

to pick those anomaly points as initial centers since they are the furthest points apart. It is also beneficial to try multiple seed initializations in order to avoid a locally minimal solution. The results in Figure 13 shows that 5 seed initializations should be sufficient in finding good clusterings.

7.2.3 BINARY SEARCH FOR PICKING *k*

SimPoint 3.0 makes it much faster to find the best clustering and simulation points for a program trace over earlier versions. Since the BIC score generally increases as *k* increases, SimPoint 3.0 uses this knowledge to perform a binary search for the best *k*. For example, if the maximum *k* desired is 100, with earlier versions of SimPoint one might search in increments of 5: $k = 5, 10, 15, \ldots, 90, 100$, requiring 20 clusterings. With the binary search method, we can ignore large parts of the set of possible *k* values and examine only about 7 clusterings.

The binary search method first clusters 3 times: at $k = 1$, $k = K$, and $k = (K+1)/2$. It then proceeds to divide the search space and cluster again based on the BIC scores observed for each clustering and the user-specified BIC threshold. Thus the binary search method requires the user only to specify the maximum number of clusters K, and performs at most $\log_2(K)$ clusterings.

Figure 14 shows the comparison between the new binary search method for choosing the best clustering, and the old method, which searched over all *k* values in the same range. The top graph shows the CPI error for each program, and the bottom graph shows the number of simulation points (clusters) chosen. These experiments are for using binary search with $K = 30$, 15 dimensions, 5 random seeds, and a BIC threshold of 90%. Exhaustive search performs slightly better than binary search, since it searches all *k* values. Using the binary search, it possible that it will not find a clustering with as few clusters as found by the exhaustive search. This is shown in the bottom graph of Figure 14, where the exhaustive search picked 19 simulation points on average, and binary search chose 22 simulation points on average. In terms of CPI error rates, the average is about the same across the SPEC programs between exhaustive and binary search. Recall that the binary search

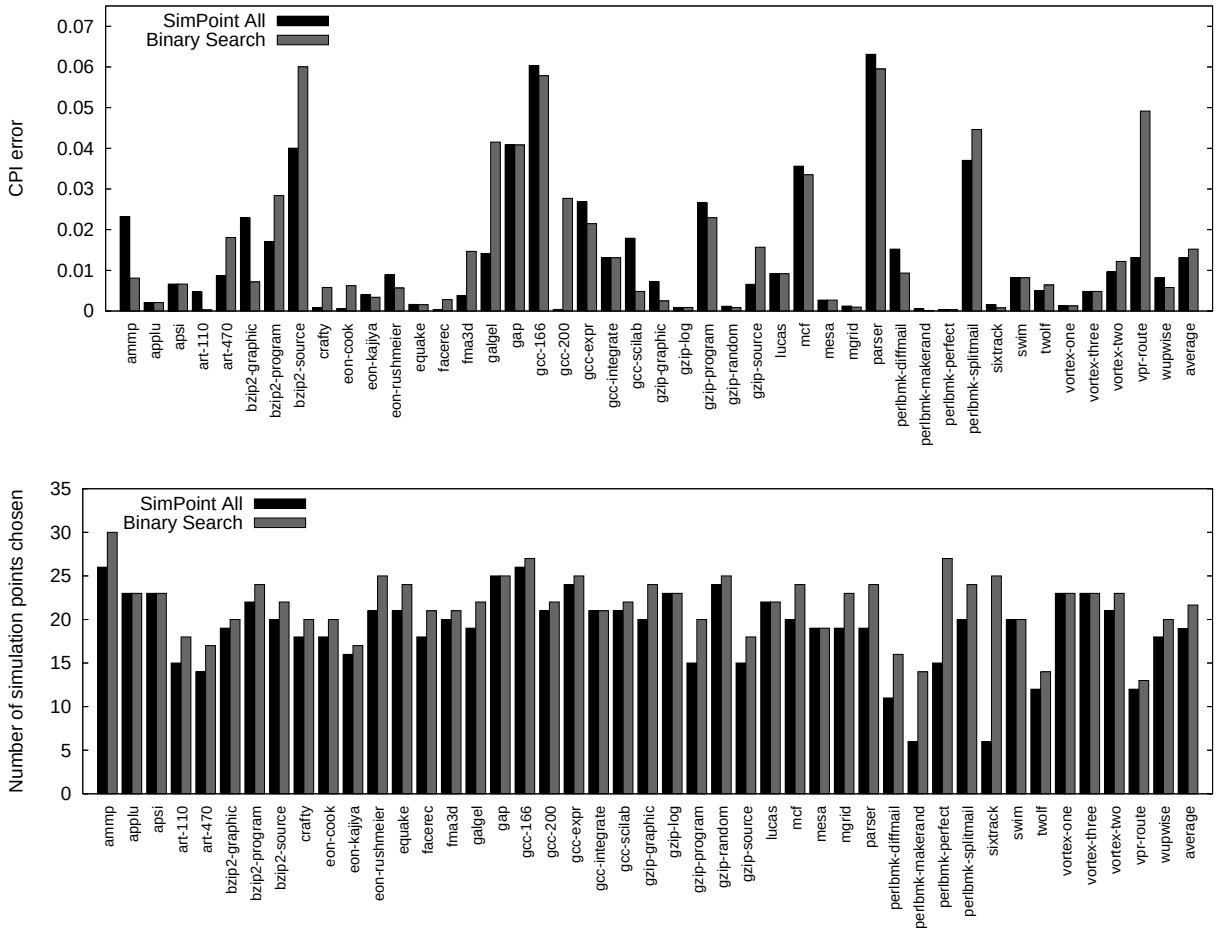


Figure 14: These plots show the CPI error and number of simulation points chosen for two different ways of searching for the best clustering. The first method, which was used in SimPoint 2.0, searches over all k between 1 and 30, and chooses the smallest clustering that achieves the BIC threshold of 90%. The second method is the binary search for $K = 30$, which examines at most 5 clusterings.

method operates many times faster than the brute force search method (see Table 4 for some timing results).

As we can see from the graphs in Figure 14, SimPoint is able to achieve a 1.5% CPI error rate averaged across all SPEC 2000 benchmarks, with a maximum error of around 6%. These results require an average simulation time of about 220 million instructions per program (for the binary search method). These error rates are sufficiently low to make design decisions, and the simulation time is small enough to do large-scale design space explorations.

8. Related Machine Learning Work on Phase Analysis

SimPoint is the first research to apply machine learning techniques (*k*-means, dimension reduction, BIC) to the problem of program phase analysis and workload performance prediction. Recently two other clustering techniques have been examined for SimPoint, which are multinomial clustering (Sanghai et al., 2005) and regression trees (Annavaram et al., 2004). Neither of these perform better than SimPoint with *k*-means clustering.

Sanghai et al. (2005) proposed utilizing mixtures of multinomials trained by EM to cluster program intervals. Unlike a *k*-means cluster, the multinomial is a probability model that explicitly models each dimension. Multinomials are used frequently in machine learning for modeling and clustering text documents, which are high-dimensional and sparse, much like the data we see in program analysis using basic block vectors. Sanghai et al. used a mixture of multinomial models to cluster the program data, and formulated a version of the BIC that applies to multinomial models. They also considered dimension reduction via a different construction of random linear projection. Their random linear projection is based on a sparse matrix where each value may be 0 or 1 (rather than real-valued). This is similar to what Achlioptas (2001) proposed for "database-friendly" projections. Following on their proposed model, we have done a full comparison of multinomial mixtures with *k*-means (Hamerly et al., 2006), and we found that *k*-means performs better for program phase analysis, but that multinomials have some benefits. We summarize that work in Section 9.

Annavaram et al. (2004) employed a regression tree clustering algorithm to predict performance for database applications and SPEC2000. Code signatures were generated through periodic sampling with a tool called VTune that samples the hardware counters. In addition to code signatures, the CPI for each interval of execution was sampled. This is a necessary parameter in the regression tree algorithm. The code signatures are divided into two groups based on the split that would minimize the variance in the CPI for the corresponding execution intervals. Subsequently, each new group is split again based on the same criteria and this is repeated recursively until no more splits can be made. To reduce complexity, up to 50 splits were applied on the data (Annavaram et al., 2004). To determine the number of clusters to be used from the data, a cross-validation step is applied with reserved CPI data that was not used in the splitting process.

The regression tree method may be effective in reducing the variance of CPI within clusters, but the need for CPI in computing clusters is a drawback. It is computationally expensive to compute the CPI for the entire execution of a program via simulation. In addition, the use of CPI data from one architecture configuration to form clusters would bind that clustering to that particular configuration. A different architecture configuration which may produce different CPI values would not necessarily fit under the former clustering formation; thus the method is not architecture independent. The *k*-Means approach employed in SimPoint uses only the code signatures to form clusters, which results in an architecture independent representation that is applicable across many configurations as shown in Section 6.

9. Current Directions

In this section we describe some of our current and future directions for phase analysis.

9.1 Matching Simulation Points to Code Boundaries

With the original SimPoint approach, representatives selected for simulation are identified by dynamic instruction count using fixed length intervals. For example, SimPoint may tell the user to start detailed simulation when 5,000,000 instructions have executed, and stop just before 6,000,000 instructions have executed, using an interval size of 1,000,000 instructions. This ties the simulation points to that specific binary, but the idea of SimPoint should be applicable across different compilations of the same source code. The same phase behavior should occur, though perhaps with different code patterns. If we can identify these behaviors and map them back to the source code level, then we could use the same phase analysis for a program compiled for different compiler optimizations or even architectures with different instruction sets. This will allow us to examine the exact same set of simulation points across different compilations of the same source code.

To address this, we propose breaking the program's execution up at procedure call and loop boundaries instead of breaking up the program's execution using fixed length intervals. Programs exhibit patterns of repetitive behavior, and these patterns are largely due to procedure call and looping behavior. Our software phase marker approach (Lau et al., 2006) detects recurring call chains and looping patterns and identifies the source code instructions to which they correspond. We then mark specific procedure calls and loop branches, so that when they occur during execution, they will indicate the end of one code signature (interval boundary) and the start of another. Therefore, instead of using fixed-length intervals with some fixed number of instructions, intervals are defined by procedure and loop boundaries. This results in *Variable Length Intervals* (VLIs) of execution.

To support VLIs, we had to modify the SimPoint software to allow sub-sampling (since we may be dealing with a huge number of intervals), and clustering with variable-length intervals (Lau et al., 2005a), where the weights of each interval are taken into consideration during the *k*-means clustering. An interesting machine-learning result of clustering variable-length intervals is how we modified the likelihood calculated for the BIC to allow it to consider the length of each interval. Because we view longer intervals as more important than shorter ones, the likelihood should reflect this. Therefore, we reformulate the likelihood we present in this paper to be appropriate for variable-length intervals. When the interval length is uniform, the modified BIC gives the same answer as the BIC presented in this paper.

The accuracy and simulation time results for software phase markers with VLIs are similar to fixed-length-interval SimPoint. Therefore the main advantage of the phase marker approach is portability of the phase analysis across compilations and architectures. In prior work (Lau et al., 2005a), we also showed that there is a clear hierarchy of phase behaviors, from fine-grained to coarse-grained depending upon the interval sizes used, and there is still future research to be done to determine how to pick the correct granularity for the target use of the phase analysis.

9.2 Multinomial Clustering

Recently, Sanghai et al. (2005) proposed using a mixture of multinomial models as a clustering model for phase analysis, as described in Section 8. Their research was a preliminary study; we have performed a more complete set of experiments comparing multinomial clustering with EM to the *k*-means algorithm, as applied to phase analysis (Hamerly et al., 2006).

We found that multinomial clustering does not improve upon *k*-means clustering in terms of performance prediction, despite the fact that basic block vectors seem to be a natural fit to multinomials. We also showed a comparison between different projection methods in conjunction with

multinomial clustering, and alternative methods of choosing the sample to simulate from each cluster. Further, we verified Sanghai et al.'s claim that the number of dimensions required to get good results using multinomial clustering may be much higher than the 15 dimensions we use with *k*-means. Following their work, we used up to 100 projected dimensions to find clustering results that work well for phase analysis with this approach.

We also found that EM clustering is much slower than *k*-means. The hard assignment of *k*-means enables optimizations like partial distance search described in Section 7.1. But for EM clustering, its soft assignment requires that we cannot stop short on examining any dimensions, so it cannot benefit from such optimizations. This together with the increase in number of dimensions required by multinomials makes multinomial EM clustering much slower than *k*-means. Even if we use the same number of dimensions to randomly project to, we still find that EM clustering of multinomials is roughly 10 times slower than *k*-means.

We did find that there are some benefits to using multinomials. One benefit is that multinomial clustering tends to choose fewer clusters on average (according to a BIC score formulated for multinomial mixtures), resulting in lower simulation times. Another benefit is that the EM algorithm uses soft assignment, unlike the hard assignment of *k*-means. This allows us to derive a metric of cluster "purity". The idea is that if many vectors have high membership in multiple clusters, then the clustering is more impure than if each vector (interval) belongs mostly to only one of the clusters. This purity score allows us to see if multinomial clustering is a good solution for a particular data set, and gives us a metric for deciding whether to apply multinomial clustering if the purity score is high enough, or *k*-means otherwise. We found that this combined approach provides a solution which picks fewer simulation points compared with using only *k*-means, and gets lower prediction errors than using only multinomial clustering.

10. Summary

Understanding the cycle level behavior of a processor running an application is crucial to modern computer architecture research, and gaining this understanding can be done efficiently by judiciously applying detailed cycle level simulation to only a few simulation points. By targeting only one or a few carefully chosen samples for each of the small number of behaviors found in real programs, the cost of simulation can be reduced to a reasonable level while achieving very accurate performance estimates.

The main idea behind SimPoint is the realization that programs typically only exhibit a few unique behaviors which are interleaved with one another through time. By finding these behaviors and then determining the relative importance of each one, we can maintain both a high level picture of the program's execution and at the same time quantify the cycle level interaction between the application and the architecture. The key to being able to find these phases in a efficient and robust manner is the development of a metric that can detect the underlying shifts in a program's execution that result in the changes in observed behavior. In this paper we have discussed one such method of quantifying executed code similarity, and use it to find program phases through the application of unsupervised learning techniques.

The methods described in this paper are distributed as part of SimPoint (Perelman et al., 2003; Sherwood et al., 2002). SimPoint automates the process of picking simulation points using an off-line phase classification algorithm based on *k*-means clustering, which significantly reduces the amount of simulation time required. Selecting and simulating only a handful of *intelligently* picked

sections of the full program provides an accurate picture of the complete execution of a program, which gives a highly accurate estimate of performance. The SimPoint software can be downloaded at:

`http://www.cse.ucsd.edu/users/calder/simpoint/`

For the industry-standard SPEC programs, SimPoint has less than a 6% error rate (2% on average) for the results in this paper, and is 1,500 times faster on average than performing simulation for the complete program's execution. Because of this time savings and accuracy, our approach is currently used by architecture researchers and industry companies (e.g. Patil et al. (2004) at Intel) to guide their architecture design exploration.

Acknowledgments

We would like to thank the anonymous reviewers for providing helpful feedback on this paper. This work was funded in part by NSF grant No. CCF-0342522, NSF grant No. CCF-0311710, a UC MICRO grant, and a grant from Intel and Microsoft.

References

D. Achlioptas. Database-friendly random projections. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-361-8.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19: 716–723, 1974.

M. Annavaram, R. Rakvic, M. Polito, J. Bouguet, R. Hankins, and B. Davies. The fuzzy correlation between code and performance predictability. In *International Symposium on Microarchitecture*, December 2004.

R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

M. Van Biesbrouck, L. Eeckhout, and B. Calder. Efficient sampling startup for uniprocessor and simultaneous multithreading simulation. In *International Conference on High Performance Embedded Architectures and Compilers*, November 2005.

M. Van Biesbrouck, T. Sherwood, and B. Calder. A co-phase matrix to guide simultaneous multithreading simulation. In *IEEE International Symposium on Performance Analysis of Systems and Software*, March 2004.

D. C. Burger and T. M. Austin. The SimpleScalar tool set, version 2.0. Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997.

D. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham. Fast search algorithms for vector quantization and pattern matching. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 9.11.1–9.11.4, 1984.

S. Dasgupta. Experiments with random projection. In *Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)*, pages 143–151, 2000.

S. Dasgupta. How fast is *k*-means? In *COLT*, page 735, 2003.

C. Elkan. Using the triangle inequality to accelerate *k*-means. In *ICML*, pages 147–153, 2003.

F. Farnstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Explor. Newsl.*, 2(1):51–57, 2000.

U. Fayyad, C. Reina, and P. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 194–198. AAAI Press, 1998.

T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

G. Hamerly and C. Elkan. Learning the *k* in *k*-means. In *Advances in NIPS*, 2003.

G. Hamerly, E. Perelman, and B. Calder. Comparing multinomial and *k*-means clustering for SimPoint. In *Proceedings of the 2006 IEEE International Symposium on Performance Analysis of Systems and Software*, 2006.

D. Hochbaum and D. Shmoys. A best possible heuristic for the *k*-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

P. Indyk, A. Amir, A. Efrat, and H H. Samet. Efficient algorithms and regular data structures for dilation, location and proximity problems. In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 160–170, 1999.

J. Lau, E. Perelman, and B. Calder. Selecting software phase markers with code structure analysis. In *Proceedings of the International Symposium on Code Generation and Optimization*, March 2006.

J. Lau, E. Perelman, G. Hamerly, T. Sherwood, and B. Calder. Motivation for variable length intervals and hierarchical phase behavior. In *IEEE International Symposium on Performance Analysis of Systems and Software*, March 2005a.

J. Lau, J. Sampson, E. Perelman, G. Hamerly, and B. Calder. The strong correlation between code signatures and performance. In *IEEE International Symposium on Performance Analysis of Systems and Software*, March 2005b.

J. Lau, S. Schoenmackers, and B. Calder. Structures for phase classification. In *IEEE International Symposium on Performance Analysis of Systems and Software*, March 2004.

J. Lau, S. Schoenmackers, and B. Calder. Transition phase classification and prediction. In *11th International Symposium on High Performance Computer Architecture*, February 2005c.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, CA, 1967. University of California Press.

J. McNames. Rotated partial distance search for faster vector quantization encoding. *IEEE Signal Processing Letters*, 7(9), 2000.

A. Moore. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 397–405. AAAI Press, 2000.

H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. Pinpointing representative portions of large Intel Itanium programs with dynamic instrumentation. In *International Symposium on Microarchitecture*, December 2004.

D. Pelleg and A. Moore. *X*-means: Extending *K*-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conf. on Machine Learning*, pages 727–734, 2000.

E. Perelman, G. Hamerly, and B. Calder. Picking statistically valid and early simulation points. In *International Conference on Parallel Architectures and Compilation Techniques*, September 2003.

F. J. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

K. Sanghai, T. Su, J. Dy, and D. Kaeli. A multinomial clustering model for fast simulation of computer architecture designs. In *KDD*, pages 808–813, 2005.

G. Schwarz. Estimating the dimension of a model. *The Annnals of Statistics*, 6(2):461–464, 1978.

T. Sherwood and B. Calder. Time varying behavior of programs. Technical Report UCSD-CS99-630, UC San Diego, August 1999.

T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *International Conference on Parallel Architectures and Compilation Techniques*, September 2001.

T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *10th International Conference on Architectural Support for Programming*, October 2002.

T. Sherwood, S. Sair, and B. Calder. Phase tracking and prediction. In *30th Annual International Symposium on Computer Architecture*, June 2003.

P. Smyth. Clustering using Monte Carlo cross-validation. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, August 1996.

Journal of Machine Learning Research 7 (2006) 379–411 Submitted 8/05; Published 2/06

Superior Guarantees for Sequential Prediction and Lossless Compression via Alphabet Decomposition

Ron Begleiter RONBEG@CS.TECHNION.AC.IL

Ran El-Yaniv RANI@CS.TECHNION.AC.IL

Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel

Editor: Dana Ron

Abstract

We present worst case bounds for the learning rate of a known prediction method that is based on hierarchical applications of binary context tree weighting (CTW) predictors. A heuristic application of this approach that relies on Huffman's alphabet decomposition is known to achieve state-of-the-art performance in prediction and lossless compression benchmarks. We show that our new bound for this heuristic is tighter than the best known performance guarantees for prediction and lossless compression algorithms in various settings. This result substantiates the efficiency of this hierarchical method and provides a compelling explanation for its practical success. In addition, we present the results of a few experiments that examine other possibilities for improving the multi-alphabet prediction performance of CTW-based algorithms.

Keywords: sequential prediction, the context tree weighting method, variable order Markov models, error bounds

1. Introduction

Sequence prediction and entropy estimation are fundamental tasks in numerous machine learning and data mining applications. Here we consider a standard discrete sequence prediction setting where performance is measured via the log-loss (self-information). It is well known that this setting is intimately related to lossless compression, where in fact high quality prediction is essentially equivalent to high quality lossless compression.

Despite the major interest in sequence prediction and the existence of a number of *universal* prediction algorithms, some fundamental issues related to learning from finite (and small) samples are still open. One issue that motivated the current research is that the finite-sample behavior of prediction algorithms is still not sufficiently understood.

Among the numerous compression and prediction algorithms there are very few that offer both finite sample guarantees and good practical performance. The *context tree weighting* (CTW) method of Willems et al. (1995) is a member of this exclusive family of algorithms. The CTW algorithm is an "ensemble method," mixing the predictions of many underlying variable order Markov models (VMMs), where each such model is constructed using zero-order conditional probability estimators. The algorithm is *universal* with respect to the class of bounded-order VMM tree-sources. Moreover, the algorithm has a finite sample point-wise redundancy bound (for any particular sequence).

The high practical performance of the original CTW algorithm is most apparent when applied to *binary* prediction problems, in which case it uses the well-known (binary) KT-estimator (Krichevsky and Trofimov, 1981). When the algorithm is applied to non-binary prediction/compression problems (using the multi-alphabet KT-estimator), its empirical performance is mediocre compared to the best known results (Tjalkens et al., 1997). Nevertheless, a clever *alphabet decomposition* heuristic, suggested by Tjalkens et al. (1994) and further developed by Volf (2002), does achieve state-of-the-art compression and prediction performance on standard benchmarks (see, e.g., Volf, 2002; Sadakane et al., 2000; Shkarin, 2002; Begleiter et al., 2004). In this approach the multi-alphabet problem is hierarchically decomposed into a number of binary prediction problems. We term the resulting procedure "the DECO algorithm." Volf suggested applying the DECO algorithm using Huffman's tree as the decomposition structure, where the tree construction is based on letter frequencies. We are not aware of any previous compelling explanation for the striking empirical success of DECO.

Our main contribution is a general worst case redundancy bound for algorithm DECO applied with any alphabet decomposition structure. The bound proves that the algorithm is *universal* with respect to VMMs. A specialization of the bound to the case of Huffman decompositions results in a tight redundancy bound. To the best of our knowledge, this new bound is the sharpest available for prediction and lossless compression for sufficiently large alphabets and sequences.

We also present a few empirical results that provide some insight into the following questions: (1) Can we improve on the Huffman decomposition structure using an optimized decomposition tree? (2) Can other, perhaps "flat" types of alphabet decomposition schemes outperform the hierarchical approach? (3) Can standard CTW multi-alphabet prediction be improved with other types of (non-KT) zero-order estimators?

Before we start with the technical exposition, we introduce some standard terms and definitions. Throughout the paper, Σ denotes a finite alphabet with $k = |\Sigma|$ symbols. Suppose we are given a sequence $\mathbf{x}_1^n = x_1 x_2 \cdots x_n$. Our goal is to generate a probabilistic prediction $\hat{P}(x_{n+1}|\mathbf{x}_1^n)$ for the next symbol given the previous symbols. Clearly this is equivalent to being able to estimate the probability $\hat{P}(\mathbf{x}_1^n)$ of any complete sequence, since $\hat{P}(x_{n+1}|\mathbf{x}_1^n) = \hat{P}(\mathbf{x}_1^{n+1})/\hat{P}(\mathbf{x}_1^n)$ (provided that the marginality condition $\sum_\sigma \hat{P}(\mathbf{x}_1^n\sigma) = \hat{P}(\mathbf{x}_1^n)$ holds).

We consider a setting where the performance of the prediction algorithm is measured with respect to the best predictor in some reference, which we call here a *comparison class*. In our case the comparison class is the set of all variable order Markov models (see details below). Let ALG be a prediction algorithm that assigns a probability estimate $P_{\mathrm{ALG}}(\mathbf{x}_1^n)$ for any given $\mathbf{x}_1^n$. The *point-wise redundancy* of ALG with respect to the predictor P and the sequence $\mathbf{x}_1^n$ is $R_{\mathrm{ALG}}(\mathbf{x}_1^n, P) = \log P(\mathbf{x}_1^n) - \log P_{\mathrm{ALG}}(\mathbf{x}_1^n)$. The per-symbol point-wise redundancy is $\frac{1}{n} R_{\mathrm{ALG}}(\mathbf{x}_1^n, P)$. ALG is called *universal* with respect to a comparison class $\mathcal{C}$, if

$$\lim_{n\to\infty} \sup_{P\in\mathcal{C}} \max_{\mathbf{x}_1^n} \frac{1}{n} R_{\mathrm{ALG}}(\mathbf{x}_1^n, P) = 0. \tag{1}$$

2. Preliminaries

This section presents the relevant technical background for the present work. The contextual background appears in Section 7. We start by presenting the class of *tree sources*. We then describe the CTW algorithm and discuss some of its known properties and performance guarantees. Finally, we conclude this section with a description of the DECO method for predicting multi-alphabet sequences using binary CTW predictors.

2.1 Tree Sources

The parametric distribution estimated by the CTW algorithm is the set of depth-bounded tree-sources. A tree-source is a variable order Markov model (VMM). Let Σ be an alphabet of size k and D a non-negative integer. A *D-bounded tree source* is any full k-ary tree[1] whose height $\leq D$. Each leaf of the tree is associated with a probability distribution over Σ. For example, in Figure 1 we depict three tree-sources over a binary alphabet. In this case, the trees are full binary trees. The single node tree in Figure 1(c) is a zero-order (Bernoulli) source and the other two trees (Figure 1(a) and (b)) are 2-bounded sources. Another useful way to view a tree-source is as a set $\mathcal{S} \subseteq \Sigma^{\leq D}$ of "suffixes" in which each $s \in \mathcal{S}$ is a path (of length up to D) *from* a (unique) leaf to the root. We also refer to $\mathcal{S}$ as the (tree-source) *topology*. For example, $\mathcal{S} = \{\mathtt{0}, \mathtt{01}, \mathtt{11}\}$ in Figure 1(b). The path from the middle leaf to the root corresponds to the sequence $s = \mathtt{01}$ and therefore we refer to this leaf simply as s. For convenience we also refer to an internal node by the (unique) path from that node to the root. Observe that this path is a suffix of some $s \in \mathcal{S}$. For example, the right child of the root in Figure 1(b) is denoted by the suffix $\mathtt{1}$.

The (zero-order) distribution associated with the leaf s is denoted $\mathbf{z}_s(\sigma), \forall \sigma \in \Sigma$, where $\sum_\sigma \mathbf{z}_s(\sigma) = 1$ and $\mathbf{z}_s(\cdot) \geq 0$.

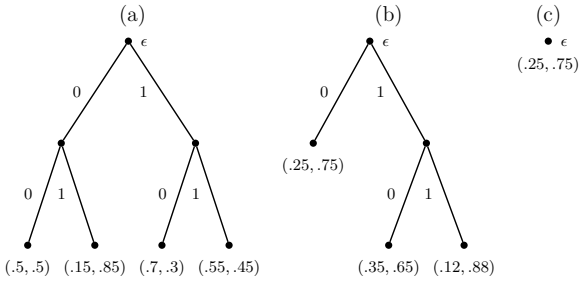


Figure 1: Three examples for $D = 2$ bounded tree-sources over $\Sigma = \{0,1\}$. The corresponding suffix-sets are $\mathcal{S}_{(a)} = \{\mathtt{00}, \mathtt{10}, \mathtt{01}, \mathtt{11}\}$, $\mathcal{S}_{(b)} = \{\mathtt{0}, \mathtt{01}, \mathtt{11}\}$, and $\mathcal{S}_{(c)} = \{\varepsilon\}$ (ε is the empty sequence). The probabilities for generating $\mathbf{x}_1^3 = \mathtt{100}$ given initial context $\mathtt{00}$ are $P_{(a)}(\mathtt{100}|\mathtt{00}) = P_{(a)}(\mathtt{1}|\mathtt{00})P_{(a)}(\mathtt{0}|\mathtt{01})P_{(a)}(\mathtt{0}|\mathtt{10}) = 0.5 \cdot 0.7 \cdot 0.15$, $P_{(b)}(\mathtt{100}|\mathtt{00}) = 0.75 \cdot 0.35 \cdot 0.25$, and $P_{(c)}(\mathtt{100}|\mathtt{00}) = 0.75 \cdot 0.25 \cdot 0.25$.

We denote the set of all D-bounded tree-source topologies (suffix sets) by $\mathcal{C}_D$. For example, $\mathcal{C}_0 = \{\{\varepsilon\}\}$ and $\mathcal{C}_1 = \{\{\varepsilon\}, \{\mathtt{0}, \mathtt{1}\}\}$, where ε is the empty sequence.

For each n, a D-bounded tree-source induces a probability distribution over the set Σ^n of all n-length sequences. This distribution depends on an initial "context" (or "state"), $\mathbf{x}_{1-D}^0 = x_{1-D} \cdots x_0$, which can be any sequence in Σ^D. The tree-source induced probability of the sequence $\mathbf{x}_1^n = x_1 x_2 \cdots x_n$ is, by the chain rule,

$$P_{\mathcal{S}}(\mathbf{x}_1^n) = \prod_{t=1}^{n} P_{\mathcal{S}}(x_t | \mathbf{x}_{t-D}^{t-1}), \tag{2}$$

where $P_{\mathcal{S}}(x_t|\mathbf{x}_{t-D}^{t-1})$ is $\mathbf{z}_s(x_t) = P_{\mathcal{S}}(x_t|s)$ and s is the (unique) suffix of $\mathbf{x}_{t-D}^{t-1}$ in $\mathcal{S}$. Clearly, a tree-source can generate sequences: the ith symbol is randomly drawn using the conditional distribution

1. A full k-ary tree is a tree in which each node has exactly zero or k children.

$P_{\mathcal{S}}(\cdot|\mathbf{x}_{i-D}^{i-1})$. Let $\text{SUB}_s(\mathbf{x}_1^n)$ be the *ordered* non-contiguous sub-sequence of symbols appearing after the context s in $\mathbf{x}_1^n$. For example, if $\mathbf{x}_1^8 = \texttt{01100101}$, and $s = \texttt{0}$, then, $\text{SUB}_s(\mathbf{x}_1^8) = \texttt{1011}$. Let s be any suffix in $\mathcal{S}$ and $\mathbf{y}_1^m = \text{SUB}_s(\mathbf{x}_1^n)$. For every $\mathbf{x}_1^n \neq \varepsilon$ we define $\mathbf{z}_s(\mathbf{x}_1^n) = \prod_{i=1}^m \mathbf{z}_s(y_i)$ and for the empty sequence $\mathbf{z}_s(\varepsilon) = 1$. Thus, we can rewrite Equation (2) as

$$P_{\mathcal{S}}(\mathbf{x}_1^n) = \prod_{s \in \mathcal{S}} \mathbf{z}_s(\mathbf{x}_1^n). \tag{3}$$

2.2 The Context-Tree Weighting Method

Here we describe the CTW prediction algorithm (Willems et al., 1995), originally presented as a lossless compression algorithm.[2] The goal of the CTW algorithm is to predict a sequence (nearly) as good as the the best tree-source. This goal can be divided into two sub-problems. The first is to guess the topology of the best tree-source, and the second is to estimate the distributions associated with its leaves.

Suppose, first, that the best tree topology (i.e., the suffix-set $\mathcal{S}$) is known. A good solution assigns to each $s \in \mathcal{S}$ a *zero-order estimator* $\hat{\mathbf{z}}_s$ that estimates the true probability distribution $\mathbf{z}_s$ associated with s. This can be done using standard statistical methods; that is, by considering all occurrences of s in $\mathbf{x}_1^n$ and constructing $\hat{\mathbf{z}}_s$ via counting and smoothing. We currently consider $\hat{\mathbf{z}}_s$ as a generic estimator and discuss specific implementations later on.

In practice, however, the best tree-source's topology is unknown. Instead of guessing this topology, CTW considers all possible D-bounded topologies $\mathcal{S}$ (each is a subtree of the perfect k-ary tree), and for each $\mathcal{S}$ it constructs a predictor by estimating its zero-order leaf probabilities. CTW then takes a weighted mixture of all these predictors, corresponding to all topologies. Clearly, there are exponentially many D-bounded topologies. The beauty of the CTW algorithm is the efficient computation of this mixture of exponential size.

In the following description of the CTW algorithm, the output of the algorithm is a probability $P_{\text{CTW}}(\mathbf{x}_1^n)$ for the entire sequence $\mathbf{x}_1^n$. Observe that this is equivalent to estimating the next-symbol probabilities because

$$P_{\text{CTW}}(\sigma|\mathbf{x}_1^n) = P_{\text{CTW}}(\mathbf{x}_1^n\sigma)/P_{\text{CTW}}(\mathbf{x}_1^n) \tag{4}$$

for each $\sigma \in \Sigma$ (provided that these probabilities can be marginalized, i.e., $\sum_\sigma P_{\text{CTW}}(\mathbf{x}_1^n\sigma) = P_{\text{CTW}}(\mathbf{x}_1^n)$).

We require the following definitions. Let $\mathbf{x}_1^n$ be any sequence (in Σ^n) and fix a bound D and an initial context $\mathbf{x}_{1-D}^0$. Let s be any context in $\mathcal{S}$, and $\mathbf{y}_1^m = \text{SUB}_s(\mathbf{x}_1^n)$. The *sequential* zero-order estimation for $\mathbf{x}_1^n$ is, by the chain-rule,

$$\hat{\mathbf{z}}_s(\mathbf{x}_1^n) = \prod_{i=1}^m \hat{\mathbf{z}}(y_i|\mathbf{y}_1^{i-1}), \tag{5}$$

where $\mathbf{y}_1^0 = \varepsilon$ and $\hat{\mathbf{z}}(y_i|\mathbf{y}_1^{i-1})$ is a zero-order probability estimate based on the symbol counts in $\mathbf{y}_1^{i-1}$. The product of such predictions is $\hat{\mathbf{z}}_s(\mathbf{x}_1^n)$, and hence, we refer to it as a sequential zero-order estimate.

We now describe the main CTW idea via a simple example and then provide a pseudo-code for the general CTW algorithm. Consider a binary alphabet and the case $D = 1$. Here, CTW works on the perfect binary tree of height one and therefore should mix the predictions associated with two

2. As mentioned above, any lossless compression algorithm can be translated into a sequence prediction algorithm and vice versa (see, e.g., Merhav and Feder, 1998).

topologies: $\mathcal{S}_0 = \{\varepsilon\}$ (where ε is the empty sequence), and $\mathcal{S}_1 = \{0,1\}$. Note that $\mathcal{S}_0$ corresponds to the zero-order topology as in Figure 1(c). The algorithm takes a mixture of the zero-order estimate $\hat{z}_\varepsilon(\mathbf{x}_1^n)$ and the one-order estimate. The latter is exactly $\hat{z}_0(\mathbf{x}_1^n) \cdot \hat{z}_1(\mathbf{x}_1^n)$ because $\hat{z}_0$ and $\hat{z}_1$ are independent. Thus, the final estimate is

$$P_{\text{CTW}}(\mathbf{x}_1^n) = \frac{1}{2}\hat{z}_\varepsilon(\mathbf{x}_1^n) + \frac{1}{2}\left(\hat{z}_0(\mathbf{x}_1^n) \cdot \hat{z}_1(\mathbf{x}_1^n)\right).$$

For larger trees ($D > 1$), CTW uses the same idea, but now, instead of taking zero-order estimates for the root's children, the CTW algorithm recursively computes their estimates. The pseudo-code of the CTW recursive mixture computation appears in Algorithm 1. We later show in Lemma 3 that this code calculates the mixture of all D-bounded tree-source predictions weighted by their complexities, which are defined as follows.

Algorithm 1 The context-tree weighting algorithm

/ This code calculates the* CTW *probability for the (whole) sequence* $\mathbf{x}_1^n$, $P_{\text{CTW}}(\mathbf{x}_1^n|\mathbf{x}_{1-D}^0)$. *The input arguments include the sequence* $\mathbf{x}_1^n$, *an initial context* $\mathbf{x}_{1-D}^0$ *(that determines the suffixes for predicting the first symbols), a bound D on the order, and an implementation for the sequential zero-order estimators* $\hat{z}_s(\cdot)$. *The code uses the* `mix` *procedure (see below). */*

CTW($\mathbf{x}_1^n$, $\mathbf{x}_{1-D}^0$, D, $\hat{z}_s(\cdot)$) {
  for every $s \in \Sigma^{\le D}$ **do**
    calculate and store $\hat{z}_s(\mathbf{x}_1^n)$ as given in Equation (5).
  end for
  return $P_{\text{CTW}}(\mathbf{x}_1^n) = \texttt{mix}(\varepsilon, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0)$.
}

/ This procedure mixes the predictions of all continuations* s'*s of* $s \in \Sigma^{\le D}$, *such that* s'*s is also in* $\Sigma^{\le D}$. *Note that the context of the first few symbols is determined by the initial context* $\mathbf{x}_{1-D}^0$. **/*
`mix`$(s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0)$ {
  if $|s| = D$ **then**
    return $\hat{z}_s(\mathbf{x}_1^n)$.
  else
    return $\frac{1}{2}\hat{z}_s(\mathbf{x}_1^n) + \frac{1}{2}\prod_{\sigma\in\Sigma}\texttt{mix}(\sigma s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0)$.
  end if
}

Definition 1 *Let* $T_{\mathcal{S}}$ *denote the tree associated with the suffix set* $\mathcal{S}$. *The complexity of* $T_{\mathcal{S}}$ *is defined to be*

$$|T_{\mathcal{S}}| = |\{s \in \mathcal{S} : |s| < D\}| + \frac{|\mathcal{S}| - 1}{k-1}.$$

Recall that the number of leaves in $T_{\mathcal{S}}$ *is exactly* $|\mathcal{S}|$ *and there are* $\frac{|\mathcal{S}|-1}{k-1}$ *internal nodes in any full k-ary tree. Therefore,* $|T_{\mathcal{S}}|$ *is the number of nodes in* $T_{\mathcal{S}}$ *minus the number of leaves* $s \in \mathcal{S}$ *with maximal depth D.*

For example, let $T_{(a)}$ be the tree of Figure 1(a) (resp. for (b) and (c)); $|T_{(a)}| = 0+3 = 3$; $|T_{(b)}| = 1+2 = 3$ $(= |T_{(a)}|)$; $|T_{(c)}| = 1+0 = 1$.

Observation 2 *Let $\mathcal{S}_\sigma = \{s : s\sigma \in \mathcal{S}\}$. For any D-bounded topology $\mathcal{S}$, $|\mathcal{S}| > 1$,*

$$|T_\mathcal{S}| = 1 + \sum_{\sigma\in\Sigma} |T_{\mathcal{S}_\sigma}|.$$

Note that $\mathcal{S}_\sigma$ is a $(D-1)$-bounded topology. Note also that the complexity depends on D. Therefore, for the base case (when $|\mathcal{S}| = 1$), the complexity of $T_\mathcal{S}$ is zero if $D = 0$ and one if $D \geq 1$.

The proof of the following lemma is a straightforward generalization of the one for binary alphabets by Willems et al. (1995).

Lemma 3 *Let $0 \leq d \leq D$ and $s \in \Sigma^d$. Then,*

$$\texttt{mix}(s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0) = \sum_{\mathcal{U}\in\mathcal{C}_{D-d}} 2^{-|T_\mathcal{U}|} \prod_{u\in\mathcal{U}} \hat{\mathtt{z}}_{us}(\mathbf{x}_1^n).$$

Recall that $\mathcal{C}_m$ is the set of all m-bounded topologies; `mix` is defined in Algorithm 1.

Proof By induction on $D-d$. When $D-d = 0$, $\mathcal{C}_{D-d} = \mathcal{C}_0$ contains only the single-node topology $\mathcal{U} = \{\varepsilon\}$. In this case $|T_\mathcal{U}| = 0 + \frac{1-1}{k-1} = 0$, by Definition 1. Notice that the size $|s| = d = D$, so $\texttt{mix}(s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0) = \hat{\mathtt{z}}_s(\mathbf{x}_1^n)$. We conclude that,

$$\texttt{mix}(s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0) = \hat{\mathtt{z}}_s(\mathbf{x}_1^n) = 2^{-0-\frac{1-1}{k-1}} \hat{\mathtt{z}}_s(\mathbf{x}_1^n) = \sum_{\mathcal{U}\in\mathcal{C}_0} 2^{-|T_\mathcal{U}|} \prod_{u\in\mathcal{U}} \hat{\mathtt{z}}_{us}(\mathbf{x}_1^n).$$

Assume that the statement holds for some $0 \leq D-d-1$ and consider the case $D-d$; that is, $|s| = d < D$. In this case $\mathcal{U} \in \mathcal{C}_{D-d}$. In the following derivations we also refer to alphabet symbols by their indices, $i = 1, \ldots, k$ (according to some fixed order) or by σ_i. For example, $\mathcal{U}_i$ is the topology corresponding to the subtree of $T_\mathcal{U}$ whose root is defined by σ_i; thus, $\mathcal{U}_i$ is a $D-d$ bounded tree-source. We thus have

$$\begin{aligned}
\texttt{mix}(s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0) &= \frac{1}{2}\hat{\mathtt{z}}_s(\mathbf{x}_1^n) + \frac{1}{2}\prod_{\sigma\in\Sigma} \texttt{mix}(\sigma s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0) && (6)\\
&= \frac{1}{2}\hat{\mathtt{z}}_s(\mathbf{x}_1^n) + \frac{1}{2}\prod_{\sigma\in\Sigma} \left\{ \sum_{\mathcal{U}\in\mathcal{C}_{D-d}} 2^{-|T_\mathcal{U}|} \prod_{u\in\mathcal{U}} \hat{\mathtt{z}}_{u\sigma s}(\mathbf{x}_1^n) \right\} && (7)\\
&= \frac{1}{2}\hat{\mathtt{z}}_s(\mathbf{x}_1^n) + \\
&\quad \sum_{\mathcal{U}_1} \cdots \sum_{\mathcal{U}_k} 2^{-\left(1+\sum_{i=1}^k |T_{\mathcal{U}_i}|\right)} \prod_{u\in\mathcal{U}_1} \hat{\mathtt{z}}_{u\sigma_1 s}(\mathbf{x}_1^n) \cdots \prod_{u\in\mathcal{U}_k} \hat{\mathtt{z}}_{u\sigma_k s}(\mathbf{x}_1^n) && (8)\\
&= \sum_{\mathcal{U}\in\mathcal{C}_{D-d}} 2^{-|T_\mathcal{U}|} \prod_{u\in\mathcal{U}} \hat{\mathtt{z}}_{us}(\mathbf{x}_1^n), && (9)
\end{aligned}$$

where step (6) is by the definition of $\texttt{mix}(s, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0)$; (7) is by the induction hypothesis; (8) is by exchanging the product of sums with sums of products; and finally, (9) follows from Observation 2. ■

The next corollary expresses the CTW prediction as a mixture of all D-bounded tree-sources. The proof of this corollary directly follows from Lemma 3 and from the definition of $P_{\text{CTW}}(\mathbf{x}_1^n)$ in Algorithm 1.

Corollary 4

$$P_{\text{CTW}}(\mathbf{x}_1^n) = \mathit{mix}(\varepsilon, \mathbf{x}_1^n, \mathbf{x}_{1-D}^0) = \sum_{\mathcal{S}\in\mathcal{C}_D} 2^{-|T_\mathcal{S}|} \prod_{s\in\mathcal{S}} \hat{z}_s(\mathbf{x}_1^n). \tag{10}$$

Remark 5 *The number of tree-source topologies in $\mathcal{C}_D$ is superexponential (recall that each $\mathcal{S} \in \mathcal{C}$ is a pruning of the perfect k-ary tree of height D). Thus, for practical reasons, the calculation of Equation (10) must be efficient. The pseudo-code of the* CTW *in Algorithm 1 is conceptual rather than efficient. However, the beauty of the* CTW *is that it can calculate the tree-source mixture in linear time with respect to n. For a description of an efficient implementation of the* CTW *algorithm, see for example, Sadakane et al. (2000) and Chapter 4.4 of Volf (2002). Our Java implementation of the* CTW *algorithm can be found at* `http://www.cs.technion.ac.il/~rani/code/vmm`.

2.3 Analysis of CTW for Multi-Alphabets

The analysis of CTW for multi-alphabets (multi-CTW) relies upon specific implementations of the sequential zero-order estimators $\hat{z}_s(\cdot)$. Such estimators are in general counters of past events. However, these estimators should not neglect unobserved events. In the context of log-loss prediction, assigning zero probability to these "zero frequency" events is harmful because the log-loss of an unobserved but possible event is infinite. The problem of assigning probability mass to unobserved events is also called the "missing-mass problem" (or the "zero frequency problem").

The original CTW algorithm applies the well-known KT estimator (Krichevsky and Trofimov, 1981).

Definition 6 *Fix any $\mathbf{x}_1^n$ and let N_σ be the frequency of $\sigma \in \Sigma$ in $\mathbf{x}_1^n$. The* KT *estimator assigns the following (sequential zero-order) probability to the sequence $\mathbf{x}_1^n$,*

$$\hat{z}^{\text{KT}}(\mathbf{x}_1^n) = \hat{z}^{\text{KT}}(\mathbf{x}_1^{n-1}) \frac{N_{x_n} + 1/2}{\sum_{\sigma\in\Sigma} N_\sigma + k/2}, \tag{11}$$

where $\hat{z}^{\text{KT}}(\varepsilon) = 1$.

Observe that the term $P(\sigma|\mathbf{x}_1^n) = \frac{N_\sigma + 1/2}{\sum_{\sigma\in\Sigma} N_\sigma + k/2}$, is an *add-half* predictor that belongs to the family of add-constant predictors.[3]

The KT estimator provides a prediction that is uniformly close to the set $\mathcal{Z}$ of zero-order distributions over Σ. Each distribution $\mathbf{z} \in \mathcal{Z}$ is a probability vector from $(\mathbb{R}_+)^k$, and $\mathbf{z}(\sigma)$ denotes the probability of σ. Thus, $\mathbf{z}(\mathbf{x}_1^n) = \prod_\sigma \mathbf{z}(\sigma)^{N_\sigma}$. The next theorem provides a performance guarantee on the worst-case redundancy of the KT estimator. This guarantee is for a whole sequence $\mathbf{x}_1^n$. Notice that the per-symbol redundancy of KT diminishes with n at a rate $\frac{\log n}{n}$. For completeness, the proof of the following theorem is provided in Appendix A.

Theorem 7 (Krichevsky and Trofimov) *Let Σ be any alphabet with $|\Sigma| = k \geq 2$. For any sequence $\mathbf{x}_1^n \in \Sigma^n$,*

$$R_{\text{KT}}(\mathbf{x}_1^n) = \log \sup_{\mathbf{z}\in\mathcal{Z}} \mathbf{z}(\mathbf{x}_1^n) - \log \hat{z}^{\text{KT}}(\mathbf{x}_1^n) \leq \frac{k-1}{2} \log n + \log k. \tag{12}$$

3. Another famous add-constant predictor is the add-one predictor, also called *Laplace's law of succession* (Laplace, 1995).

Remark 8 *Krichevsky and Trofimov (1981) originally defined* KT *to be a mixture of all zero-order distributions in $\mathcal{Z}$, weighted by the Dirichlet $(1/2)$ distribution. Thus, this mixture is*

$$\hat{\mathbf{z}}^{\mathrm{KT}}(\mathbf{x}_1^n) = \int_{\mathcal{Z}} w(d\mathbf{z})\mathbf{z}(\mathbf{x}_1^n),$$

where $w(dz)$ is the Dirichlet distribution with parameter $1/2$ defined by

$$w(dz) = \frac{1}{\sqrt{k}} \frac{\Gamma(\frac{k}{2})}{\Gamma(\frac{1}{2})^k} \prod_{i=1}^{k} z(i)^{-1/2} \lambda(dz), \tag{13}$$

$\Gamma(x) = \int_{\mathbb{R}^+} t^{x-1}\exp(-t)dt$ is the gamma function (see, for example, Courant and John, 1989), and $\lambda(\cdot)$ is a measure on $\mathcal{Z}$. Shtarkov (1987) was the first to show that this mixture can be calculated sequentially as in Definition 6.

The upper bound of Theorem 7 on the redundancy of the KT estimator is a key element in the proof of the following theorem, providing a finite-sample point-wise redundancy bound for the multi-CTW (see, e.g., Tjalkens et al., 1993; Catoni, 2004).

Theorem 9 (Willems et al.) *Let Σ be any alphabet with $|\Sigma| = k \geq 2$. For any sequence $\mathbf{x}_1^n \in \Sigma^n$ and any D-bounded tree-source with a topology $\mathcal{S}$ and distribution $P_{\mathcal{S}}$, the following holds:*

$$R_{\mathrm{CTW}}(\mathbf{x}_1^n, P_{\mathcal{S}}) \leq \begin{cases} n\log k + \frac{k|\mathcal{S}|-1}{k-1}, & n < |\mathcal{S}|; \\ \frac{(k-1)|\mathcal{S}|}{2}\log\frac{n}{|\mathcal{S}|} + |\mathcal{S}|\log k + \frac{k|\mathcal{S}|-1}{k-1}, & n \geq |\mathcal{S}|. \end{cases}$$

Proof

$$\begin{aligned} R_{\mathrm{CTW}}(\mathbf{x}_1^n, P_{\mathcal{S}}) &= \log P_{\mathcal{S}}(\mathbf{x}_1^n) - \log P_{\mathrm{CTW}}(\mathbf{x}_1^n) \\ &= \underbrace{\log \frac{P_{\mathcal{S}}(\mathbf{x}_1^n)}{\prod_{s\in\mathcal{S}} \hat{\mathbf{z}}_s(\mathbf{x}_1^n)}}_{(i)} + \underbrace{\log \frac{\prod_{s\in\mathcal{S}} \hat{\mathbf{z}}_s(\mathbf{x}_1^n)}{P_{\mathrm{CTW}}(\mathbf{x}_1^n)}}_{(ii)} \end{aligned} \tag{14}$$

We now bound the term (14)(i) and define the following auxiliary function:

$$f(x) = \begin{cases} x\log k & ,0 \leq x < 1; \\ \frac{k-1}{2}\log x + \log k & ,x \geq 1. \end{cases}$$

Note that this function is continuous and concave in $[0,\infty)$. Let $N_\sigma(s)$ denote the frequency of σ in $\text{SUB}_s(\mathbf{x}_1^n)$. Thus,

$$\begin{aligned}
\log\frac{P_{\mathcal{S}}(\mathbf{x}_1^n)}{\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)} &= \sum_{s\in\mathcal{S}}\log\frac{\mathbf{z}_s(\mathbf{x}_1^n)}{\hat{z}_s(\mathbf{x}_1^n)} && (15)\\
&\le \sum_{\substack{s\in\mathcal{S},\text{ s.t.}\\ \sum N_\sigma(s)>0}}\left(\frac{k-1}{2}\log(\sum_\sigma N_\sigma(s))+\log k\right) && (16)\\
&= |\mathcal{S}|\sum_{s\in\mathcal{S}}\frac{1}{|\mathcal{S}|}f(\sum_\sigma N_\sigma(s))\\
&\le |\mathcal{S}|f(\frac{\sum_{s\in\mathcal{S}}\sum_\sigma N_\sigma(s)}{|\mathcal{S}|}) && (17)\\
&= |\mathcal{S}|f(\frac{n}{|\mathcal{S}|})\\
&= \begin{cases} n\log k, & n<|\mathcal{S}|;\\ \frac{(k-1)|\mathcal{S}|}{2}\log\frac{n}{|\mathcal{S}|}+|\mathcal{S}|\log k, & n\ge|\mathcal{S}|, \end{cases} && (18)
\end{aligned}$$

where step (15) follows from an application of Equation (3); step (16) is by the performance guarantee for the KT prediction, as given in Theorem 7; and step (17) is by Jensen's inequality.

We now bound the term (14)(*ii*)

$$\begin{aligned}
\log\frac{\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)}{P_{\text{CTW}}(\mathbf{x}_1^n)} &= \log\frac{\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)}{\sum_{\mathcal{S}\in\mathcal{C}_D}2^{-|T_{\mathcal{S}}|}\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)} && (19)\\
&\le \log\frac{\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)}{\sum_{\mathcal{S}\in\mathcal{C}_D}2^{-\frac{k|\mathcal{S}|-1}{k-1}}\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)} && (20)\\
&\le \log\frac{\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)}{2^{-\frac{k|\mathcal{S}|-1}{k-1}}\prod_{s\in\mathcal{S}}\hat{z}_s(\mathbf{x}_1^n)}\\
&= \log 2^{\frac{k|\mathcal{S}|-1}{k-1}}\\
&= \frac{k|\mathcal{S}|-1}{k-1}, && (21)
\end{aligned}$$

where in step (19) we applied Equation (10) and the justification for (20) is that $|\{s\in\mathcal{S}:|s|<D\}|\le|\mathcal{S}|$. Thus, according to Definition 1, $|T_{\mathcal{S}}|\le|\mathcal{S}|+\frac{|\mathcal{S}|-1}{k-1}=\frac{k|\mathcal{S}|-1}{k-1}$. We complete the proof by summing up (18) and (21). ■

Remark 10 *The* CTW *bound used by Catoni (2004) is somewhat tighter than the bound of Theorem 9 but contains some implicit terms.*

Remark 11 *Willems (1998) provided extensions for the* CTW *algorithm that eliminate its dependency on the maximal bound D and the initial context* $\mathbf{x}_{1-D}^0$*. For the extended algorithm and binary prediction problems, Willems derived a point-wise redundancy bound of*

$$\frac{|\mathcal{S}|}{2}\log\frac{n-\Delta_s(\mathbf{x}_1^n)}{|\mathcal{S}|}+2|\mathcal{S}|-1+\Delta_s(\mathbf{x}_1^n),$$

where $\Delta_s(\mathbf{x}_1^n) \leq D$ denotes the number of symbols in the prefix of $\mathbf{x}_1^n$ that do not appear after a suffix $s \in \mathcal{S}$.

Remark 12 *Interestingly, it can be shown that the* CTW *algorithm is an instance of the well-known generic* expert-advice *algorithm of Vovk (1990). This observation is new, to the best of our knowledge, although there are citations that connect the* CTW *algorithm with the expert-advice scheme (see, e.g., Merhav and Feder, 1998; Helmbold and Schapire, 1997).*

It can be shown that these two algorithms are identical when Vovk's algorithm is applied with the log-loss (see, e.g., Haussler et al., 1998, example 3.12). In this case, the set of experts in Vovk's algorithm consists of all D-bounded tree-sources, $\mathcal{C}_D$; the initial weight of each expert, $\mathcal{S}$, corresponds to its complexity $|T_S|$; and the weight of each expert at round t equals $2^{-|T_S|}P_S(\mathbf{x}_1^{t-1})$. Note, however, that the power of the CTW *method is in its efficiency in mixing exponentially many sources (or experts). Vovk's algorithm is not concerned with how to compute this average.*

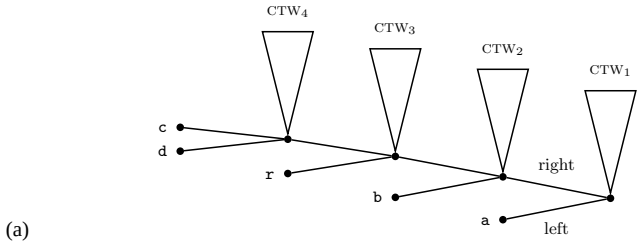
2.4 Hierarchical CTW Decompositions

The CTW algorithm is known to achieve excellent empirical performance in *binary* prediction problems. However, when applying CTW on sequences over larger alphabets, the resulting performance falls short of the best known (Tjalkens et al., 1997). This fact motivates different approaches for applying the CTW algorithm on multi-alphabet sequences. Volf targeted this issue in his Ph.D. thesis (2002). Following Tjalkens et al. (1994), who proposed a rudimentary alphabet decomposition approach, he studied a solution to the multi-alphabet prediction problem that is based on a tree hierarchy of binary problems. Each of these binary problems is solved using a slight variation of the binary CTW algorithm. We now describe the resulting 'decomposed CTW' approach, which we term for short the "DECO" algorithm.

Consider a full binary *decomposition tree* T with $k = |\Sigma|$ leaves, where each leaf is uniquely associated with a symbol in Σ. Each internal node v of T corresponds to the binary problem of predicting whether the next symbol is a leaf on v's left subtree or a leaf on v's right subtree. For example, for $\Sigma = \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{r}\}$, Figure 2 depicts a decomposition tree T such that its root corresponds to the problem of predicting whether the next symbol is $\mathtt{a}$ or one of the symbols in $\{\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{r}\}$. The idea is to learn a binary predictor that is based on the CTW algorithm, for each internal node.

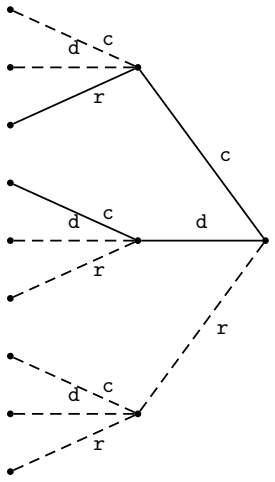
Let v be any internal node of T and let $\mathtt{L}(v)$ (resp., $\mathtt{R}(v)$) be the left (resp., right) child of v. Also, let Σ_v be the set of leaves (symbols) in the sub-tree rooted by v. We denote by CTW_v any perfect k-ary tree that provides binary predictions over the binary alphabet $\{0_v, 1_v\}$. The supersymbol 0_v (resp., 1_v) represents *any* of the symbols in $\Sigma_{\mathtt{L}(v)}$ (resp., $\Sigma_{\mathtt{R}(v)}$). While CTW_v generates binary predictions (for its supersymbols), it still depends on a suffix set over the entire k-ary alphabet Σ. Thus, internal node v yields the probability $P_{\text{CTW}_v}(\sigma_{\mathtt{super}}|s)$, where $\sigma_{\mathtt{super}} \in \{0_v, 1_v\}$ and $s \in \mathcal{S} \subseteq \Sigma^{\leq D}$. For example, in Figure 2(b) we depict CTW_3. Observe that $\hat{\mathtt{z}}_s$ estimates a binary distribution that is based on the counts appearing in the table of Figure 2(b).

Let $\mathbf{x}$ be any sequence and $\sigma \in \Sigma$. Algorithm DECO generates the multi-alphabet prediction $P_{\text{DECO}}(\sigma|\mathbf{x})$ by multiplying the binary predictions of all CTW_v along the path from the root of T to the leaf σ. Hence, $P_{\text{DECO}}(\sigma|\mathbf{x}) = \prod_{v,\ \text{s.t.},\ \sigma\in\Sigma_v} P_{\text{CTW}_v}(\sigma|\mathbf{x})$, where $P_{\text{CTW}_v}(\sigma|\mathbf{x})$ is the binary prediction of the appropriate supersymbol (either 0_v or 1_v).



(a)

CTW_3 (for $\mathbf{x} = \mathtt{rcdr}$)		
s	$N_{\{\mathtt{c},\mathtt{d}\}}(s)$	$N_{\{\mathtt{r}\}}(s)$
ϵ	1	1
c	1	0
d	0	1
r	0	0
rc	1	0
cd	0	1
cc	0	0
⋮	⋮	⋮
rr	0	0



(b)

Figure 2: A DECO predictor corresponding to the sequence abracadabra. (a) depicts the decomposition tree T. Each internal node in T utilizes a CTW predictor to "solve" a binary problem. In (b) we depict CTW_3, a 2-bounded predictor whose binary problem is: "determine if $\sigma \in \{\mathtt{c},\mathtt{d}\}$ (or $\sigma = \mathtt{r}$)." ($N_\sigma(s)$ denotes the frequency of σ in $\text{SUB}_s(\mathbf{x})$ and dashed lines mark tree paths with zero counts).

There are many possibilities for constructing the decomposition tree T.[4] A major open problem is how to identify useful decomposition trees. Intuitively, it appears that placing high frequency symbols close to the root is a good idea for two reasons: (i) When traversing the tree from the root to such symbols, the number of visits to other internal nodes is minimized, thus reducing extra loss; (ii) High frequency symbols appearing closer to the root could be involved in "easier" binary problems because of the denser statistics we have on them.

Tjalkens et al. (1997) and Volf (2002, Chapter 5) suggested taking T as the Huffman coding tree computed with respect to the frequency counts of the symbols in $\mathbf{x}_1^n$. While intuitively appealing, there is currently no compelling explanation for this heuristic. In Section 3.1 we provide a formal motivation for Huffman decompositions.

4. We can map every decomposition tree with the partition of 1 into sums of k terms, each of which is a power of $1/2$, where each leaf σ at level ℓ_σ defines the power $(1/2)^{\ell_\sigma}$. (This is possible due to Kraft's inequality.) Therefore, the number of such decomposition trees is obtained by multiplying $k!$ (all permutations of Σ) with this number of partitions. The former is known as sequence A002572 in Sloane and Plouffe (1995). For example, for $k = 26$ we have $26! \cdot 565168 = 227927428502001453851738112000000$ possible decomposition trees.

3. Redundancy Bounds For the DECO Algorithm

We start this section with some definitions that formalize the hierarchical alphabet decomposition approach. We also define a new category of sources called "decomposed sources," which will aid in the analysis of algorithm DECO. To this end, we use an equivalence between decomposed sources and the ordinary tree-sources of Section 2.1. The main result of this section is Theorem 19, providing a pointwise redundancy bound for the DECO algorithm. This bound, in particular, implies a performance guarantee for Huffman decomposition trees, which is given in Corollary 23.

Let Σ be a multi-alphabet with k symbols and fix some order bound D and initial context $\mathbf{x}_{1-D}^{0}$. We refer to a decomposition-tree (see Section 2.4) simply as a *tree* and to an ordinary tree source as a *multi-source*, denoted by $M = (\mathcal{S}, P_{\mathcal{S}})$.

Definition 13 (Decomposed Source) *A (D-bounded) decomposed source $\mathcal{T}$ over Σ is a pair*

$$\mathcal{T} = (T, \{M_1, M_2, \cdots, M_{k-1}\}),$$

where T is a (decomposition) tree over Σ and for each internal node, $v \in T$, there is a matching source $M_v = (\mathcal{S}_v, P_v)$ whose suffix set, $\mathcal{S}_v$, contains all paths of some full k-ary tree (of maximal height D). Additionally, for every $s \in \mathcal{S}_v$, $P_v(\cdot|s)$ is a binary *distribution over $\{0_v, 1_v\}$. Note that M_i is not a standard multi-source because it predicts binary sequences of supersymbols while depending on multi-alphabet contexts. Such sources will always be denoted by M_v for some internal node v. Let $\mathbf{x} \in \Sigma^D$ be any sequence and $\sigma \in \Sigma$. The prediction induced by $\mathcal{T}$ is*

$$P_{\mathcal{T}}(\sigma|\mathbf{x}) = \prod_{v,\ \text{s.t.},\ \sigma \in \Sigma_v} P_v(\sigma|\mathbf{x}). \tag{22}$$

We say that two probabilistic tree-sources over Σ are *equivalent* if they agree on the probability of every sequence $\mathbf{x} \in \Sigma^*$. Note that two structurally different tree-sources can be equivalent. A multi-source is *minimal* if it has no redundant suffixes. A decomposed source is minimal if all its M_v models are minimal. The formal definitions follow.

Definition 14 (Minimal Sources) *(i) A multi-source $M = (\mathcal{S}, P_{\mathcal{S}})$ is minimal if there is no $s \in \Sigma^{<D}$ for which $P_{\mathcal{S}}(\cdot|\sigma_i s) = P_{\mathcal{S}}(\cdot|\sigma_j s)$ for all $\sigma_i \neq \sigma_j$ and both $\sigma_i s$ and $\sigma_j s$ are in $\mathcal{S}$. (ii) We say that $\mathcal{T} = (T, \{M_v\})$ is a minimal decomposed source if for all internal nodes v of T, M_v is minimal.*

For example, we depict in Figure 3 two equivalent multi-sources. The multi-source in Figure 3 (a) is a minimal multi-source while the multi-source in Figure 3 (b) is not minimal.

There is a simple procedure for transforming a non-minimal source into its equivalent minimal form: Replace each redundant suffix, σs, with its suffix s. That is, trim all children of s and assign $P_{\mathcal{S}}(\cdot|s) = P_{\mathcal{S}}(\cdot|\sigma s)$ for some σ.

The following two lemmas facilitate a "translation" between decomposed and multi sources.

Lemma 15 *For every multi-source M and tree T there exists a minimal decomposed source $\mathcal{T} = (T, \{M_v\})$ such that M and $\mathcal{T}$ are equivalent.*

Proof Let $M = (\mathcal{S}, P)$ be a D-bounded multi-source and let T be a tree. We start with the definition of the models $M_v = (\mathcal{S}_v, P_v)$ and set the suffix set $\mathcal{S}_v = \mathcal{S}$, for every internal node v. Let $P_{\mathcal{S}}(0_v|s) =$

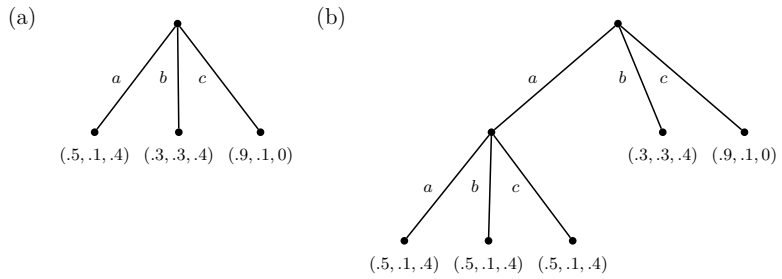


Figure 3: An example of two equivalent multi-sources. Both sources generate the same probability to every sequence of length larger than two. Take for example $\mathbf{x} = \mathtt{aaba}$ with initial context $\mathtt{ba}$. Both sources will induce the following prediction: $P(\mathtt{aaba}|\mathtt{ba}) = 0.5 \cdot 0.5 \cdot 0.1 \cdot 0.3 = 0.0075$. Observe that the source in (a) is minimal while the other source is not.

$\sum_{\sigma \in 0_v} P_{\mathcal{S}}(\sigma|s)$ for any (internal node) v and $s \in \mathcal{S}_v (= \mathcal{S})$. Similarly, $P_{\mathcal{S}}(1_v|s) = \sum_{\sigma \in 1_v} P_{\mathcal{S}}(\sigma|s)$. Let $\mathtt{parent}(v)$ denote the parent of node v. For every internal node v and $s \in \mathcal{S}_v$ we define

$$P_v(0_v|s) = P_{\mathcal{S}}(0_v|s)/P_{\mathtt{parent}(v)}(0_v \cup 1_v|s),$$

and similarly for $P_v(1_v|s)$. For the base case (i.e., the root) we do not divide by the denominator. Clearly, $P_v(\cdot|s)$ is a valid distribution and the resulting structure $\mathcal{T} = (T, \{M_v\})$ is a decomposed source.

We shall now prove that $\mathcal{T}$ and M are equivalent. Recall that $\mathcal{S} = \mathcal{S}_v$ for every internal node v. Let v ($\neq$ root) be any internal node in T and $u = \mathtt{parent}(v)$. Assume, without loss of generality, that $0_v \subset 1_u$, and therefore, $0_v \cup 1_v = 1_u$. Note that, for every $s \in \mathcal{S}$,

$$P_u(1_u|s)P_v(0_v|s) = P_u(1_u|s)\,(P_{\mathcal{S}}(0_v|s)/P_u(0_v \cup 1_v|s)) = P_{\mathcal{S}}(0_v|s).$$

Therefore, $P_{\mathcal{T}}(\sigma|s)$ of Equation (22) is a telescopic product; hence, for every $\sigma \in \Sigma$ and $s \in \mathcal{S}$, $P_{\mathcal{T}}(\sigma|s) = P_{\mathcal{S}}(\sigma|s)$. This proves that M and $\mathcal{T}$ are equivalent. Finally, for minimality, we replace every $\mathcal{S}_v$ with its minimal source. ■

Lemma 16 *For every decomposed source $\mathcal{T}$ there exists a minimal multi-source M that is equivalent to $\mathcal{T}$.*

Proof Let $\mathcal{T} = (T, \{M_v = \{\mathcal{S}_v, P_v\}\})$ be a decomposed source. We provide the following constructive scheme for building the equivalent multi-source, $M = (\mathcal{S}, P_{\mathcal{S}})$. Start with $M = M_{\mathtt{root}}$ (the model corresponding to the root of T). We traverse the internal vertices of T (minus the root) such that parent nodes are visited before their descendants (i.e., using preorder). We start with one of the root's children and for each internal node in T we do the following. For each (internal node) $v \in T$ and for every $s_v \in \mathcal{S}_v$, exactly one of the following three cases holds (because both $\mathcal{S}_v$ and $\mathcal{S}$ form a full k-ary tree): (a) $s_v = s \in \mathcal{S}$; or (b) $\exists s \in \mathcal{S}$ such that s is a suffix of s_v; or (c) $\exists s \in \mathcal{S}$ such that s_v is a suffix of s. We treat these cases as follows. For the first case, we refine the support set of $P_{\mathcal{S}}(\cdot|s)$

by replacing the supersymbol corresponding to $0_v \cup 1_v$ with the two (super) symbols 0_v and 1_v, and define,

$$\begin{aligned} P_{\mathcal{S}}(0_v|s) &= P_{\mathcal{S}}(0_v \cup 1_v|s) \cdot P_v(0_v|s); \\ P_{\mathcal{S}}(1_v|s) &= P_{\mathcal{S}}(0_v \cup 1_v|s) \cdot P_v(1_v|s). \end{aligned} \tag{23}$$

Note that $P_{\mathcal{S}}(0_v \cup 1_v|s)$ has been already assigned (due to the preorder node traversal). Cases (b) and (c) are treated in exactly the same manner. In case (b) also replace s with its extension s_v.

We should now prove that the resulting $M = (\mathcal{S}, P_{\mathcal{S}})$ is a multi-source and that M is equivalent to $\mathcal{T}$. Both proofs are by induction on $|\Sigma| = k$. For $k = 2$, $\mathcal{T}$ consists only of $M_{\texttt{root}}$, which is a binary tree source. Hence, obviously, $M = M_{\texttt{root}}$ is a tree source equivalent to $\mathcal{T}$. Assume the statement holds for $k-1 \geq 2$ and examine $|\Sigma| = k$. Let $v \in T$ be the last visited node in the constructive scheme. Clearly, by the preorder traversal, the children of v are both leaves (both 0_v and 1_v are singletons). Merge the two symbols in $\Sigma_v \subseteq \Sigma$ into some supersymbol σ_v and consider $\mathcal{T}' = (T', \{M_{v'}\})$, which is the decomposed source induced by this replacement. The number of leaves of $\mathcal{T}'$, which can be denoted $\Sigma' = \Sigma \setminus \Sigma_v \cup \{\sigma_v\}$, is equal to $k-1$. Thus, by the inductive hypothesis, we construct $M' = \{\mathcal{S}', P_{\mathcal{S}'}\}$, a multi-source that is equivalent to $\mathcal{T}'$. We now apply the constructive step on M' and v, resulting with $M = (\mathcal{S}, P_{\mathcal{S}})$. Case (b) of the constructive scheme is the only place that we change $\mathcal{S}'$ (to retrieve $\mathcal{S}$). $\mathcal{S}'$ is a tree source topology by the induction hypothesis; so is $\mathcal{S}_v$ and clearly, the treatment of case (b) induces a valid tree-source topology (that corresponds to a full k-ary tree). Therefore, $\mathcal{S}$ is a tree-source topology. It is also easy to see that the refinement of the support set of $\mathcal{S}'$, as in (23), induces a valid distribution over Σ. We conclude that $M = (\mathcal{S}, P_{\mathcal{S}})$ is a multi-source over Σ.

We now turn to prove the equivalence. For every $s \in \mathcal{S}$ and any symbol $\sigma \in \Sigma \setminus \Sigma_v$, we have by Equation (22) that $P_{\mathcal{T}}(\sigma|s) = P_{\mathcal{T}'}(\sigma|s)$, and by the induction hypothesis, $P_{\mathcal{T}'}(\sigma|s) = P_{\mathcal{S}'}(\sigma|s)$. Note that, by the construction, every $s' \in \mathcal{S}'$ is a *suffix* of some $s \in \mathcal{S}$. Therefore, for symbols $\sigma \in \Sigma \setminus \Sigma_v$, $P_{\mathcal{S}'}(\sigma|s') = P_{\mathcal{S}'}(\sigma|s) = P_{\mathcal{S}}(\sigma|s)$ (where s' is the suffix of s). Now for symbols $\sigma \in \Sigma_v$, recall that $|\Sigma_v| = 2$ and therefore, 0_v represents some (ordinary) symbol $\sigma \in \Sigma$ (resp., 1_v). Thus,

$$\begin{aligned} P_{\mathcal{S}}(\sigma|s) &= P_{\mathcal{S}'}(\sigma_v|s) P_v(\sigma|s) && (24) \\ &= P_{\mathcal{T}'}(\sigma_v|s) P_v(\sigma|s) && (25) \\ &= \left(\prod_{u,\ \text{s.t.},\ u \in T' \wedge \sigma \in \Sigma_u} P_u(\sigma|s) \right) P_v(\sigma|s) && (26) \\ &= \prod_{u,\ \text{s.t.},\ u \in T \wedge \sigma \in \Sigma_u} P_u(\sigma|s) && (27) \\ &= P_{\mathcal{T}}(\sigma|s), \end{aligned}$$

where (24) is by the construction (23) with $\sigma \in \{0_v, 1_v\}$; (25) is by the induction hypothesis; (26) and (27) are by Equation (22). This proves that M is equivalent to $\mathcal{T}$. Finally, for satisfying the minimality of M, we take its equivalent minimal multi-source. ■

Remark 17 *It can be shown that a minimal decomposed source (resp., multi-source) is unique. Hence, Lemmas 15 and 16 imply that, for a given tree T, there is a one-to-one mapping between the minimal decomposed sources and multi-sources.*

Consider algorithm DECO applied with a tree T_{DECO}. The redundancy of the DECO algorithm on a sequence $\mathbf{x}_1^n$, with respect to any decomposed source $\mathcal{T} = (T, \{M_v\})$, is

$$R_{\text{DECO}}(\mathbf{x}_1^n, \mathcal{T}) = \log P_{\mathcal{T}}(\mathbf{x}_1^n) - \log P_{\text{DECO}}(\mathbf{x}_1^n).$$

We do not know how to express this redundancy directly in terms of the unknown source $\mathcal{T}$. However, we can express it in terms of an equivalent decomposed source $\mathcal{T}'$ that has the same tree as in the algorithm. This "translation" is done using an equivalent multi-source mediator that can be constructed according to Lemmas 15 and 16. To facilitate this discussion, we define, for a decomposed source $\mathcal{T} = (T, \{M_v\})$, its *$T'$-equivalent* source to be any equivalent decomposition source with tree T'. By Lemmas 15 and 16 this source exists.

Corollary 18 *For any decomposed source $\mathcal{T} = (T, \{M_v\})$ and a tree T' there exists a T'-*equivalent *source $\mathcal{T}' = (T', \{M_i'\})$.*

Theorem 19 *Let T_{DECO} be any tree and $\mathbf{x}_1^n$ a sequence. For every internal node $v \in T_{\text{DECO}}$, denote by CTW_v the corresponding* CTW *predictor of the* DECO *algorithm applied with T_{DECO}. Let $\mathcal{T} = (T, \{M_v\})$ be any decomposed source. Then, $R_{\text{DECO}}(\mathbf{x}_1^n, P_{\mathcal{T}}) \leq \sum_{i=1}^{k-1} R_i(\mathbf{x}_1^n)$, where i is an internal-node in T_{DECO}, and*

$$R_i(\mathbf{x}_1^n) = \begin{cases} \frac{|\mathcal{S}_i|}{2} \log \frac{n_i}{|\mathcal{S}_i|} + |\mathcal{S}_i| + \frac{k|\mathcal{S}_i|-1}{k-1} & , n_i \geq |\mathcal{S}_i|; \\ n_i + \frac{k|\mathcal{S}_i|-1}{k-1} & , 0 < n_i < |\mathcal{S}_i|; \\ 0 & , n_i = 0. \end{cases} \tag{28}$$

$\mathcal{S}_i$ is the suffix set of the ith (internal) node of the T'-equivalent source of $\mathcal{T}$, and n_i is the number of times this node is visited when predicting $\mathbf{x}_1^n$.

Proof Let $\mathcal{T}' = (T_{\text{DECO}}, \{M_{v'}\})$ be the T_{DECO}-equivalent decomposed source of $\mathcal{T}$. Fix any order on the internal nodes of T_{DECO}. We will refer to internal nodes both by their order's index and by the notation v. By the chain-rule, $P_v(\mathbf{x}_1^n) = \prod_{x_t \in \Sigma_v} P_v(x_t | \mathbf{x}_{1-D}^{t-1})$, where $P_v(x_t | \mathbf{x}_{1-D}^{t-1}) = P_v(x_t | s)$ and $s \in \mathcal{S}_v$ is a suffix of $\mathbf{x}_{1-D}^{t-1}$. Thus,

$$\begin{aligned} P_{\mathcal{T}}(\mathbf{x}_1^n) &= P_{\mathcal{T}'}(\mathbf{x}_1^n) && (29) \\ &= \prod_{t=1}^{n} P_{\mathcal{T}'}(x_t | \mathbf{x}_{1-D}^{t-1}) \\ &= \prod_{t=1}^{n} \prod_{v \in T_{\text{DECO}}, \text{ s.t., } x_t \in \Sigma_v} P_v(x_t | \mathbf{x}_{1-D}^{t-1}) \\ &= \prod_{v \in T_{\text{DECO}}} \prod_{x_t \in \Sigma_v} P_v(x_t | \mathbf{x}_{1-D}^{t-1}) = \prod_{v \in T_{\text{DECO}}} P_v(\mathbf{x}_1^n), && (30) \end{aligned}$$

where (29) follows from by Corollary 18.

We show that $R_{\text{DECO}}(\mathbf{x}_1^n, P_{\mathcal{T}}) \leq \sum_{i=1}^{k-1} R_i(\mathbf{x}_1^n)$.

$$\begin{aligned}
R_{\text{DECO}}(\mathbf{x}_1^n, P_{\mathcal{T}}) &= \log P_{\mathcal{T}}(\mathbf{x}_1^n) - \log P_{\text{DECO}}(\mathbf{x}_1^n) && (31)\\
&= \log P_{\mathcal{T}'}(\mathbf{x}_1^n) - \log P_{\text{DECO}}(\mathbf{x}_1^n) && (32)\\
&= \sum_{j=1}^{k-1} \log P_j(\mathbf{x}_1^n) - \sum_{i=1}^{k-1} \log P_{\text{CTW}_i}(\mathbf{x}_1^n) && (33)\\
&= \sum_{i=1}^{k-1} (\log P_i(\mathbf{x}_1^n) - \log P_{\text{CTW}_i}(\mathbf{x}_1^n)) && (34)\\
&\leq \sum_{i=1}^{k-1} R_i(\mathbf{x}_1^n),
\end{aligned}$$

where (31) follows from Corollary 18; in Equations (32) and (33) the probabilities P_j and P_i refer to internal nodes of $\mathcal{T}'$; in (32) we used Equation (30); and finally, equality (34) directly follows from the proof of Theorem 9. In that proof, we applied the bound (18) for the term (14 *i*) with $k = 2$, because the zero-order predictors, $z_s(\cdot)$, of CTW_v provide binary predictions. The bound on the term (14 *ii*) remains as is because CTW_v uses a k-ary tree. ■

The precise values of the model orders $|\mathcal{S}_i|$ in the above upper bound are unknown since the decomposed source is unknown. Nevertheless, for each i, $|\mathcal{S}_i| \leq k^D$. It follows that any DECO scheme is universal with respect to the class of D-bounded (multi) tree-sources. Specifically, given any multi-source, consider its T_{DECO}-equivalent decomposed source $\mathcal{T}$. For a sequence $\mathbf{x}_1^n$, by Theorem 19 the per-symbol redundancy is $\frac{1}{n} R_{\text{DECO}}(\mathbf{x}_1^n, P_{\mathcal{T}}) \leq \frac{1}{n} \sum_{i=1}^{k-1} R_i(\mathbf{x}_1^n)$, which vanishes with n since $n_i \leq n$ for every internal-node i.

Remark 20 *The dependency of the* DECO *algorithm on the maximal bound D and the initial context* $\mathbf{x}_{1-D}^0$ *can be eliminated by using the extensions for the* CTW *algorithm suggested by Willems (1998). Recall that Willems provided a point-wise redundancy bound for this case (see Remark 11). Thus, we can straightforwardly use this result to derive a corresponding bound for the* DECO *algorithm (the details are omitted).*

3.1 Huffman Decompositions

The general bound of Theorem 19 holds for any decomposition tree. However, it is expected that some trees will result in a tighter bound. Therefore, it is desirable to optimize the bound over all trees. Unfortunately, the sizes $|\mathcal{S}_i|$ are unknown. Even if the sizes $|\mathcal{S}_i|$ were known, it is an NP-hard problem even to decide on the optimal partition corresponding to the root. This hardness result can be obtained by a reduction from MAX-CUT (see, e.g., Papadimitriou, 1994, Chapter 9.3). Hence, we can only hope to approximate the optimal tree.

However, if we replace each $|\mathcal{S}_i|$ value with its maximal value k^D, we are able to show that the bound is optimized when the decomposition tree is the Huffman decoding tree (see, e.g., Cover and Thomas, 1991, Chapter 5.6) of the sequence $\mathbf{x}_1^n$.

For any decomposition tree T and a sequence $\mathbf{x}_1^n$, let n_i be the number of times that the internal node $i \in T$ is visited when predicting $\mathbf{x}_1^n$ using the DECO algorithm. These are precisely the n_i used in Theorem 19, Equation (28). We call these n_i "the counters of T".

Lemma 21 *Let $\mathbf{x}_1^n$ be a sequence and T a decomposition tree constructed using Huffman's procedure, which is based on the empirical distribution $\hat{P}(\sigma) = N_\sigma/n$. Let $\{n_i\}$ be the counters of T. Then, $\sum_{i=1}^{k-1} n_i$ and $\prod_{i=1}^{k-1} n_i$ are both minimal with respect to any other decomposition tree.*

Proof Any tree T induces the following prefix-code over Σ. The codeword of a symbol $\sigma \in \Sigma$ is the path from the root of T to the leaf σ. The length of this code for some T, with respect to $\mathbf{x}_1^n$, is $\ell(\mathbf{x}_1^n) = \sum_{t=1}^n \ell(x_t)$, where $\ell(x_t)$ is the codeword length of the symbol x_t. It is not hard to see that

$$\ell(\mathbf{x}_1^n) = \sum_\sigma N_\sigma \cdot \ell(\sigma) = \sum_{i=1}^{k-1} n_i. \tag{35}$$

If T is constructed using Huffman's algorithm, the average code length, $\frac{1}{n}\sum_\sigma N_\sigma \cdot \ell(\sigma)$, is the smallest possible. Therefore, T minimizes $\frac{1}{n}\sum_{i=1}^{k-1} n_i$.

To prove that Huffman's tree also minimizes $\prod_{i=1}^{k-1} n_i$, we define the following lexicographic order on the set of inner nodes of any tree. Given a tree, we let n_v be the counter corresponding to inner node v. We can order the inner nodes, first in ascending order of their counters n_v, and then (among nodes with equal counters), in ascending order of the heights of the sub-trees they root. Let T be a Huffman tree, and T' be any other tree. Let $\{n_v\}$ be the counters of T and let $\{n_{v'}\}$ be the counters of T'. We already know that $\sum_v n_v \le \sum_{v'} n_{v'}$. We can order (separately) both sets of counters according to the above lexicographic order such that $n_{v_1} \le \cdots \le n_{v_{k-1}}$ (and similarly, for v'_i). We prove, by induction on k, that $n_{v_i} \le n_{v'_i}$, for $i = 1, \ldots, k-1$. For $k = 2$ the statement trivially holds. Assume that for $i = 1, \ldots, k-1$, $n_{v_i} \le n_{v'_i}$. We examine now the case where $i = 1, \ldots, k$. According to the construction scheme of the Huffman tree (see, Cover and Thomas, 1991, Chapter 5.6), we have that $n_{v_1} \le n_{v'_1}$. Note that the children of v_1 and v'_1 are all leaves. Otherwise, the non-leaf child must have the same counter as its parent and is rooting a sub-tree with smaller height. Therefore, by our lexicographic order, the counter of this child must appear before the counter of its parent, which is a contradiction.

Replace v_1 (resp., v'_1) with a leaf. Note that every node v (resp., v') in the resulting trees keeps its original counter n_v (resp., $n_{v'}$). Hence, nodes can change their order only with nodes of equal counter. Thus, by applying the inductive hypothesis we concluded that $n_{v_i} \le n_{v'_i}$ for $i = 1, \ldots, k$. ■

Remark 22 *After establishing Lemma 21, we found that Glassey and Karp (1976) showed that if $f(\cdot)$ is an arbitrary concave function, then the Huffman tree minimizes $\sum_{i=1}^{k-1} f(n_i)$. This general result clearly implies Lemma 21.*

From Lemma 21 it follows that the tree constructed by Huffman's algorithm minimizes any linear function of either $\sum_i n_i$ or $\sum_i \log n_i$, which proves, using Theorem 19, the following corollary.

Corollary 23 *Let $\bar{R}_i$ be the R_i of Equation (28) with every $|\mathcal{S}_i|$ replaced by its maximal value, k^D. Then, $R_{\text{DECO}}(\mathbf{x}_1^n, P_T) \le \sum_i \bar{R}_i(\mathbf{x}_1^n)$ and the Huffman coding tree minimizes this bound. The resulting bound is given in Corollary 25.*

4. Mind the Gap

Here we compare our redundancy (upper) bound for DECO and the known bound for multi-CTW. Relying on Corollary 23, we focus on the case where DECO uses the Huffman tree.

A clear advantage of the DECO algorithm is that it "activates" only internal node (binary) predictors corresponding to observed symbols. This can be seen by the bound of Theorem 19, which decreases with the number of unobserved symbols. Since the multi-CTW bound is insensitive to alphabet sparsity, this suggests that DECO will outperform the multi-CTW when predicting sequences in which alphabet symbols are sparse.

In this section we prove that the redundancy bound of DECO is strictly better than the corresponding multi-CTW bound, for any sufficiently long sequence. For this purpose, we examine the difference between the two bounds using a worst-case expression of the DECO bound.

Let Σ be an alphabet with $|\Sigma| = k$ and $\mathbf{x}_1^n$ be a sequence over Σ. Fix some order D and let $\mathcal{S}$ be the topology corresponding to the D-bounded tree-source that maximizes the probability of $\mathbf{x}_1^n$ over $\mathcal{C}_D$. Denote by $\bar{R}_{\text{CTW}}$ the multi-CTW redundancy bound (see Theorem 9),

$$\bar{R}_{\text{CTW}}(\mathbf{x}_1^n) = \frac{(k-1)|\mathcal{S}|}{2}\log\frac{n}{|\mathcal{S}|} + |\mathcal{S}|\log k + \frac{k|\mathcal{S}|-1}{k-1}. \tag{36}$$

Similarly, let $\bar{R}_{\text{HUFF}}$ denote the redundancy of DECO applied with a Huffman-tree (see Theorem 19),

$$\bar{R}_{\text{HUFF}}(\mathbf{x}_1^n) = \sum_{i=1}^{k-1}\left(\frac{\Psi}{2}\log\frac{n_i}{\Psi} + \Psi + \frac{k\Psi-1}{k-1}\right), \tag{37}$$

where Ψ is an upper-bound on the model-sizes $|\mathcal{S}_i|$ (see Equation 28). We would like to bound below the gap $\bar{R}_{\text{CTW}} - \bar{R}_{\text{HUFF}}$ between these bounds.

The next lemma and corollary provide a worst case upper bound for $\bar{R}_{\text{HUFF}}$.

Lemma 24 *Let $\mathbf{x}_1^n$ be a sequence over Σ. Let T be the corresponding Huffman decomposition tree and $\{n_i\}_{i=1}^{k-1}$ its internal node counters. Then,*

$$\sum_{i=1}^{k-1}\log n_i < (k-1)\cdot(\log n + \log(1+\log k) - \log(k-1)) \tag{38}$$

Proof Recall that for every symbol $\sigma \in \Sigma$, N_σ denote the number of occurrences of σ in $\mathbf{x}_1^n$ and $\ell(\sigma)$ denotes the length of the path from the root of T to the leaf σ. Denote by $\hat{H}$ the empirical entropy, $\hat{H} = -\sum_{\sigma\in\Sigma}\frac{N_\sigma}{n}\log\frac{N_\sigma}{n}$.

$$\begin{aligned}
\sum_{i=1}^{k-1}\frac{1}{k-1}\log n_i &\leq \log\left(\sum_{i=1}^{k-1}\frac{1}{k-1}\log n_i\right) && (39)\\
&= \log\left(\sum_{i=1}^{k-1}\log n_i\right) - \log(k-1)\\
&= \log\left(\sum_{\sigma\in\Sigma}N_\sigma\ell(\sigma)\right) - \log(k-1) && (40)\\
&< \log\left(n\cdot(1+\hat{H})\right) - \log(k-1) && (41)\\
&\leq \log\left(n\cdot(1+\log k)\right) - \log(k-1) && (42)\\
&= \log n + \log(1+\log k) - \log(k-1). && (43)
\end{aligned}$$

In (39) we used Jensen's inequality; (40) is an application of Equation (35); T yields a Huffman code with an average code length of $\sum_{\sigma\in\Sigma}\frac{N_\sigma}{n}\ell(\sigma) < 1+\hat{H}$ (see, e.g., Cover and Thomas, 1991, Section 5.4 and 5.8), which implies (41); finally, (42) follows from the fact that $\hat{H} \leq \log k$ (see, e.g., Cover and Thomas, 1991, Theorem 2.6.4). We conclude by multiplying both sides by $k-1$. ■

Corollary 25

$$\bar{R}_{\text{HUFF}}(\mathbf{x}_1^n) < \frac{(k-1)\Psi}{2}\left(\log\frac{n}{\Psi}+\log(1+\log k)-\log(k-1)+2+\frac{2k}{k-1}\right).$$

Proof

$$\begin{aligned}
\bar{R}_{\text{HUFF}}(\mathbf{x}_1^n) &= \sum_{i=1}^{k-1}\left(\frac{\Psi}{2}\log\frac{n_i}{\Psi}+\Psi+\frac{k\Psi-1}{k-1}\right)\\
&= \sum_{i=1}^{k-1}\left(\frac{\Psi}{2}\log n_i\right)+\sum_{i=1}^{k-1}\left(-\frac{\Psi}{2}\log(\Psi)+\Psi+\frac{k\Psi-1}{k-1}\right)\\
&= \frac{\Psi}{2}\sum_{i=1}^{k-1}(\log n_i)+(k-1)\left(-\frac{\Psi}{2}\log(\Psi)+\Psi+\frac{k\Psi-1}{k-1}\right)\\
&< \frac{(k-1)\Psi}{2}(\log n+\log(1+\log k)-\log(k-1))+\\
&\quad (k-1)\left(-\frac{\Psi}{2}\log(\Psi)+\Psi+\frac{k\Psi-1}{k-1}\right) \qquad (44)\\
&= \frac{(k-1)\Psi}{2}\left(\log\frac{n}{\Psi}+\log(1+\log k)-\log(k-1)\right)+(k-1)\left(\Psi+\frac{k\Psi-1}{k-1}\right)\\
&< \frac{(k-1)\Psi}{2}\left(\log\frac{n}{\Psi}+\log(1+\log k)-\log(k-1)+2+\frac{2k}{k-1}\right). \qquad (45)
\end{aligned}$$

Here (44) follows by application of (38) and we obtained (45) using $\frac{k\Psi-1}{k-1} < \frac{k\Psi}{k-1}$. ■

The next theorem characterizes cases where the DECO algorithm has a strictly smaller redundancy bound than the multi-CTW bound.

Theorem 26 *Let Σ be an alphabet with $|\Sigma| = k \geq 118$ and $\mathbf{x}_1^n$ be a sequence over Σ generated by the (unknown) D-bounded multi-source $\mathcal{M} = (\mathcal{S}, P_{\mathcal{S}})$. Then, $R_{\text{CTW}}(\mathbf{x}_1^n) > \bar{R}_{\text{HUFF}}(\mathbf{x}_1^n)$.*

Proof We take the upper bound $\Psi = |\mathcal{S}|$. By the proof of Lemma 15, when translating $\mathcal{M}$ into its equivalent decomposed source, the internal node topologies are first initiated with $\mathcal{S}$ and then may

be pruned to achieve minimality. Hence, $|\mathcal{S}|$ is an upper bound on the sizes $|\mathcal{S}_i|$. Thus, we have

$$\begin{aligned}
\bar{R}_{\text{CTW}}(\mathbf{x}_1^n) - \bar{R}_{\text{HUFF}}(\mathbf{x}_1^n) &= \frac{(k-1)|\mathcal{S}|}{2}\log\frac{n}{|\mathcal{S}|} + |\mathcal{S}|\log k + \frac{k|\mathcal{S}|-1}{k-1} - \\
&\quad \sum_{i=1}^{k-1}\left(\frac{|\mathcal{S}|}{2}\log\frac{n_i}{|\mathcal{S}|} + |\mathcal{S}| + \frac{k|\mathcal{S}|-1}{k-1}\right) \\
&> \frac{(k-1)|\mathcal{S}|}{2}\log\frac{n}{|\mathcal{S}|} + |\mathcal{S}|\log k + \frac{k|\mathcal{S}|-1}{k-1} - \frac{(k-1)|\mathcal{S}|}{2} \times \\
&\quad \left(\log\frac{n}{|\mathcal{S}|} + \log(1+\log k) - \log(k-1) + 2 + \frac{2k}{k-1}\right) \qquad (46) \\
&= |\mathcal{S}|\log k + \frac{k|\mathcal{S}|-1}{k-1} - \\
&\quad \frac{(k-1)|\mathcal{S}|}{2}\left(\log(1+\log k) - \log(k-1) + 2 + \frac{2k}{k-1}\right), \qquad (47)
\end{aligned}$$

where (46) is by Corollary 25. Using straightforward analysis it is not hard to show that (47) grows with k and is positive for $k \geq 118$. This completes the proof. ■

The gap, between the CTW and DECO bounds, shown in Theorem 26 is relevant when the internal node redundancies of DECO are $R_i = \frac{|\mathcal{S}_i|}{2}\log\frac{n_i}{|\mathcal{S}_i|} + |\mathcal{S}_i| + \frac{k|\mathcal{S}_i|-1}{k-1}$. By a simple analysis of Equation (28) using the function $f(x) = \frac{x}{2}\log\frac{n}{x} + x + \frac{kx-1}{k-1}$, we can show that the gap is positive when $n_i \geq \max\{0.17 \cdot \Psi, \mathcal{S}_i\}$.

We conclude that the redundancy bound of DECO algorithm converges faster than the bound of the CTW algorithm for alphabet of size $k \geq 118$. Currently, the CTW algorithm is known to have the best convergence rate (see Table 5). Therefore, the current bound is the tightest one known for prediction (and lossless compression) in realistic settings.

Remark 27 *The result of Theorem 26 is obtained using a worst-case analysis for the* DECO *redundancy. This analysis considered a sequence that contains all alphabet symbols; each symbol appears sufficiently many times. However, in many practical applications (such as predictions of ASCII sequences) most of the symbols are expected to have small frequencies (e.g., by Zipf's Law). In this case, the* DECO *redundancy is even smaller than the worst case bound of Corollary 25 and the gap between the two bounds is larger.*

5. Examining Other Alphabet Decompositions

The bound $\bar{R}_{\text{HUFF}}$, given in Equation (37), is optimized using a Huffman decomposition tree (Corollary 23). However, replacing each $|\mathcal{S}_i|$ with its maximal value can affect the bound considerably. For example, if we manage to place a very easy (binary) prediction problem at the root, it could be the case that the "true" model order for this problem is very small. Such considerations are not explicitly treated by the Huffman tree optimization. Therefore, it is of major interest to consider other types of alphabet decomposition trees. Also, if our goal is to utilize the (successful) *binary* CTW in multi-alphabet problems, there is no apparent reason why we should restrict ourselves to *hierarchical* alphabet decompositions as discussed so far. The parallel study of "multi-category de-

compositions" in supervised learning suggests other approaches such *one-vs-all*, *all-pairs*, etc. (see, e.g., Allwein et al., 2001).

We empirically targeted two questions: (i) Are there better alphabet decomposition trees for the DECO algorithm? (ii) Can the "flat" decomposition techniques of supervised learning be effectively applied in our sequential prediction setting?

To answer the first question, we developed a simple heuristic procedure that attempts to increase log-likelihood performance of the DECO algorithm, starting from any decomposition tree. This procedure searches for a locally optimal tree using the actual performance of DECO on a given sequence. Starting from a given tree, this procedure attempts to swap an alphabet symbol from one subtree to the other while recursively "optimizing" the resulting subtrees. Each such swap is 'accepted' only if it improves the actual performance. We applied this procedure using a Huffman tree as the starting point and refer to the resulting algorithm as 'Improved'.

Sequence	Random	Improved	Huffman	Huffman Comb	Inverted Huffman-Comb
bib	1.91	**1.81**	1.83	2.04	2.16
news	2.47	**2.34**	2.36	2.65	2.75
book1	2.26	**2.20**	2.21	2.28	2.38
book2	1.99	**1.92**	1.94	2.06	2.14
paper1	2.40	**2.26**	2.27	2.58	2.69
paper2	2.31	**2.21**	2.23	2.41	2.53
paper3	2.60	**2.45**	2.47	2.74	2.87
paper4	2.95	**2.72**	2.75	3.20	3.34
paper5	3.12	**2.86**	2.89	3.42	3.56
paper6	2.50	**2.32**	2.36	2.67	2.84
trans	1.52	**1.40**	1.43	1.71	1.89
progc	2.51	**2.32**	2.35	2.76	2.87
progl	1.74	**1.64**	1.67	1.88	2.01
progp	1.78	**1.63**	1.66	1.92	2.09
Average	2.29	**2.15**	2.17	2.45	2.58

Table 1: Comparing average log-loss of DECO with different decomposition structures. The best results appear in boldface. Results for the random decomposition reflect an average on ten random trees.

We experimented with DECO, 'Improved,' and several others decomposition schemes. Following standard convention in the lossless compression community, we examined the algorithms over the 'Calgary Corpus.' This Corpus serves as a standard benchmark for testing log-loss prediction and lossless compression algorithms (Bell et al., 1990; Witten and Bell, 1991; Cleary and Teahan, 1995; Begleiter et al., 2004). The corpus consists of 18 files of nine different types. Most of the files are pure ASCII files and four are binary files. The ASCII files consist of English texts (books 1-2 and papers 1-6), a bibliography file (bib), a batch of unedited news articles (news), some source code of computer programs (prog c,l,p), and a transcript of a terminal session (trans). The longest file (book1) has 785kb symbols and the shortest (paper5) 12kb symbols.

In addition to the Huffman and 'Improved' decompositions, we include the performance of a random tree and two types of "Huffman Comb" trees. The random tree was constructed bottom-up in agglomerative random fashion where symbol cluster pairs to be merged were selected uniformly at random among all available nodes at each 'merge' step. Each of the two 'comb' trees is a full (binary) tree of height $k-1$. That is, such trees operate similarly to *decision lists*. The comb tree whose leaves (symbols) are ordered top-down according to their ascending frequencies in $\mathbf{x}_1^n$ is referred to as the "Huffman Comb," and the comb tree whose leaves are reversely ordered is called the "Inverted Huffman Comb." Obviously, it is expected at the outset that the inverted Huffman comb will give rise to inferior performance.

In all the experimental results below we analyzed the statistical significance of pairwise comparisons between algorithms using the Wilcoxon signed rank test (Wilcoxon, 1945)[5] with a confidence level of 95%.

Table 1 shows the average prediction performance of DECO compared to several tree structures over the text files of the Calgary Corpus. The slightly better but statistically significant performance of the improved-DECO indicates that there are more effective trees than Huffman's. It is also interesting to see that the random tree (based on an average of 10 random trees) is significantly better than both the Huffman Comb trees. The latter observation suggests that it is hard to construct very inefficient decomposition structures.

Sequence$*$10%	DECO	All-Pairs	One-vs-All
progc	**3.11**	4.28	4.04
progl	**1.66**	2.27	2.16
progp	**2.69**	3.53	3.50
paper1	**3.08**	3.82	3.67
paper2	**3.15**	3.66	3.62
paper3	**3.39**	4.10	4.00
paper4	**3.89**	4.62	4.54
paper5	**3.91**	4.82	5.02
paper6	**3.32**	4.11	4.00
Average	**3.13**	3.91	3.84

Table 2: Comparing three decomposition methods over a reduced version of the Calgary Corpus. The best results appear in boldface.

To investigate the second question, regarding other decomposition schemes, we implemented the 'one-vs-all' and 'all-pairs' schemes, straightforwardly adapted to our sequential setting. The reader is referred to Rifkin and Klautau (2004) for a discussion of these techniques in standard supervised learning. The prediction results, over a reduced version of the Calgary text files, appear in Table 2. In this reduced dataset we took 10% (from the start) of each original sequence. The reason for considering smaller texts (of shorter sequences) is the excessive memory requirements of the 'all-pairs' algorithm, which requires $\binom{k}{2} = 8128$ different binary predictors (compared to the

5. The Wilcoxon signed rank test is a nonparametric alternative to the paired t-test, which is similar to the Fisher sign test. This test assumes that there is information in the magnitudes of the differences between paired observations, as well as the signs.

$k-1$ and k binary predictors required by DECO and 'one-vs-all', respectively).[6] The results of Table 2 indicate that the hierarchical decomposition is better than the other two flat decomposition schemes. (Note that the advantage of 'one-vs-all' over 'all-pairs' is at 90% confidence.)

6. On Applying CTW with Other Zero-Order Estimators

Another interesting direction when attempting to improve the performance of the standard CTW on multi-alphabet sequences is to use other, perhaps stronger (in some sense), zero-order estimators instead of the KT estimator. In particular, it seems most appropriate to consider well-known estimators such as Good-Turing and the very recent ones proposed by Orlitsky et al. (2003), some of which have strong performance guarantees in a certain worst case sense.

To this end, we compared the prediction quality of multi-CTW and DECO each applied with four different sequential zero-order estimators: Good-Turing (denoted $\hat{z}^{\mathrm{GT}}$), "Improved add-one" (denoted $\hat{z}^{+\underline{1}}$), "improved Good-Turing" (denoted $\hat{z}^{\mathrm{GT}^*}$) and standard KT (denoted $\hat{z}^{\mathrm{KT}}$). The description of the first three estimators is provided in Appendix B.

Sequence	$\hat{z}^{\mathrm{KT}}$	$\hat{z}^{+\underline{1}}$	$\hat{z}^{\mathrm{GT}}$	$\hat{z}^{\mathrm{GT}^*}$
bib	2.47	2.35	**2.27**	2.29
news	2.92	2.82	**2.75**	**2.75**
book1	2.50	2.46	**2.42**	**2.42**
book2	2.32	2.24	**2.19**	2.20
paper1	2.98	2.83	**2.73**	2.75
paper2	2.77	2.68	**2.60**	2.61
paper3	3.16	3.08	3.00	**2.99**
paper4	3.57	3.50	3.41	**3.38**
paper5	3.76	3.66	3.57	**3.56**
paper6	3.10	2.95	**2.84**	2.85
trans	2.18	1.92	**1.76**	1.84
progc	3.04	2.89	**2.79**	2.82
progl	2.29	2.14	**2.05**	2.08
progp	2.26	2.11	**2.00**	2.04
Average	2.80	2.69	**2.60**	2.61

Table 3: Comparing the average log-loss of multi-CTW with different sequential zero-order estimators. The comparison is made with textual ($|\Sigma| = 128$) sequences taken from the Calgary Corpus, and with parameter $D = 5$. Each numerical value is the average log-loss (the loss per symbol). The best (minimal) result of each comparison is marked in boldface.

All four estimators have worst-case performance guarantees based on a maximal *likelihood ratio*, which is the ratio between the highest possible probability assigned by some distribution and the probability assigned by the estimators. The set of "all possible distributions" considered is referred to as the comparison class. Orlitsky et al. analyzed the performance of these estimators

6. With our two gigabyte RAM machine the runs with the entire corpus would take approximately two months.

Sequence	$\hat{2}^{\mathrm{KT}}$	$\hat{2}^{+\underline{1}}$	$\hat{2}^{\mathrm{GT}}$	$\hat{2}^{\mathrm{GT}*}$
bib	**1.84**	2.39	2.02	2.35
news	**2.36**	2.94	2.54	2.85
book1	**2.22**	2.39	2.23	2.38
book2	**1.94**	2.27	2.02	2.26
paper1	**2.28**	3.03	2.53	2.93
paper2	**2.23**	2.74	2.39	2.68
paper3	**2.47**	3.08	2.66	2.98
paper4	**2.75**	3.52	3.00	3.36
paper5	**2.90**	3.78	3.18	3.59
paper6	**2.36**	3.16	2.63	3.04
trans	**1.43**	2.43	1.83	2.35
progc	**2.35**	3.16	2.61	3.03
progl	**1.67**	2.33	1.90	2.26
progp	**1.66**	2.44	1.95	2.37
Average	**2.18**	2.83	2.39	2.74

Table 4: Comparing predictions of DECO with different sequential zero-order estimators. The comparison is made with textual ($|\Sigma| = 128$) sequences taken from the Calgary Corpus, and with parameter $D = 5$. Each numerical value resemble the average log-loss (the loss per-symbol). The best (minimal) result of each comparison is marked in boldface.

for *infinite* discrete alphabets and a comparison class consisting of *all* possible distributions over n-length sequences. They showed that the average *per-symbol* ratio is infinite for sequential add-constant estimators such as KT. The Good-Turing and Improved add-one estimators assign to each ('large') sequence a probability which is at most a factor of c^n (for some constant $c > 1$) smaller than the maximal possible probability; the improved Good-Turing estimator assigns to each sequence a probability that is within a sub-exponential factor of the maximal probability.

In addition to the above, the KT and Good-Turing estimators enjoy the following guarantees. In Theorem 7 we stated a *finite-sample* guarantee for the redundancy of the KT estimator. Recall that this guarantee refers to *finite* alphabets and a comparison class consisting of zero-order distributions. Moreover, within this setting, KT was shown to be (asymptotically) close, up to a constant, to the best possible ratio (Xie and Barron, 2000; Freund, 2003), and the constant is proportional to the alphabet size. Thus, when considering the per-symbol ratio, KT is *asymptotically* optimal. Along with the above worst-case guarantees, the Good-Turing estimator also has a convergence guarantee to the "true" missing mass probability (McAllester and Schapire, 2000), assuming the existence of a true underlying distribution that generated the sequence.

In Tables 3 and 4 we provide the respective per symbol log-loss obtained with these estimators for all the textual (ASCII) sequences from the Calgary Corpus (14 datasets). In all the experiments below we analyzed the statistical significance of the results using the Wilcoxon signed rank test at a confidence of 95%.

Table 3 presents the log-loss of the four zero-order estimators when used as the zero-order predictor within the multi-CTW scheme. The support set of the zero-order estimators is of size 128.

Observe that multi-CTW with $\hat{z}^{\mathrm{KT}}$ suffers the worst log-loss. On the other hand, when applying these estimators in DECO (thus, when solving binary prediction problems), as depicted in Table 4, the $\hat{z}^{\mathrm{KT}}$ outperforms all the other estimators. Also observe that the best multi-CTW result ($\hat{z}^{\mathrm{GT}}$ in Table 3) is worse than the best DECO result ($\hat{z}^{\mathrm{KT}}$ in Table 4).

In summarizing these results, we note that:

- For text sequences, the CTW algorithm can be significantly improved when applied with the Good-Turing estimator (instead of the KT estimator).
- The improved Good-Turing estimator proposed by Orlitsky et al. (2003) does not improve the Good-Turing.
- The Deco-Huffman algorithm achieves best performance with the original (binary) KT estimator.

7. Related Work

To the best of our knowledge, hierarchical alphabet decompositions in the log-loss prediction/compression setting were first considered by Tjalkens, Willems and Shtarkov (1994).[7] In this paper, the authors study a hierarchical decomposition where each internal node in the decomposition tree is associated with a (binary) KT estimator (instead of binary-CTW instances in DECO). In this setting the comparison class is the set of all zero order sources. The authors derived a redundancy bound of $k-1+\frac{1}{2}\sum_{n_i>0}\log n_i$ for this algorithm, where the n_i terms are the node counters as defined in Theorem 19. This result is similar to a special case of our bound, $2+\frac{1}{2}\sum_{n_i>0}\log n_i$, obtained using Theorem 19 for the special case $D=0$ (implying $|\mathcal{S}_i|=1$). In that paper Tjalkens et al. proposed the essence of the DECO algorithm as presented here; however, they did not provide the details. A thorough study of algorithm DECO and other CTW-based approaches for dealing with multi-alphabets are presented in Volf's Ph.D. thesis (Volf, 2002). In particular, an in-depth empirical study of DECO, over the Calgary and Canterbury Corpora, indicated that this algorithm achieves state-of-the-art performance in lossless compression. Thus, it matches the good performance of the *prediction by partial match* (PPM) family of heuristics.[8] Further empirical evidence that substantiated this observation appears in Sadakane et al. (2000); Shkarin (2002); Begleiter et al. (2004).

There are also many discrete prediction algorithms that are not CTW-based. We restrict the discussion here to some of the most popular algorithms that are known to be universal with respect to some comparison class. Probably the most famous (and the first) universal lossless compression algorithms were proposed by Ziv and Lempel (1977; 1978). For example, the well-known LZ78 algorithm is a fast dictionary method that avoids explicit statistical considerations. This algorithm is universal (with respect to the set of ergodic sources); however, in contrast to both conventional wisdom and the algorithm's phenomenal commercial success, it is not among the best lossless compressors (see, e.g., Bell et al., 1990).

Two more recent universal algorithms are the Burrows-Wheeler transform (BWT) (Burrows and Wheeler, 1994) and grammar-based compression (Yang and Kieffer, 2000). The public-domain

7. A similar paper by Tjalkens, Volf and Willems, proposing the same method and results, appeared a few years later (Tjalkens et al., 1997).
8. As far as we know, the best PPM performance over the Calgary Corpus is reported for the PPM-II variant, proposed by Shkarin (2002).

version of BWT, called BZIP, is considered to be a relatively strong compressor over the Calgary Corpus, which is fast but somewhat inferior to PPM and DECO. The grammar-based compression algorithm has the advantage of providing an "explanation" (grammar) for the way it compressed the target sequence.

The point-wise (worst case) redundancy of the prediction game was introduced by Shtarkov (1987). Given a comparison class $\mathcal{C}$ of target distributions P and some hypothesis class $\mathcal{P}$, from which the prediction algorithm selects one approximating distribution $\hat{P}$, the point-wise redundancy of this game is

$$R_n^*(\mathcal{C}) = \inf_{\hat{P}\in\mathcal{P}} \sup_{P\in\mathcal{C}} \max_{\mathbf{x}_1^n} \log \frac{P(\mathbf{x}_1^n)}{\hat{P}(\mathbf{x}_1^n)}.$$

Shtarkov also presented the first asymptotic lower bound on the redundancy for the case where both the hypothesis and comparison classes are the set D-order Markov sources. To date, the tightest *asymptotic* lower bound on the point-wise redundancy for D-gram Markov sources was recently given by Jacquet and Szpankowski (2004, Theorem 3). They showed that for large (but unspecified) n, the lower bound is $\frac{k^D(k-1)}{2}\log\frac{n}{2\pi} + \log A(D,k) + \log(1+O(\frac{1}{n}))$, where $A(D,k)$ is a constant depending on the order D and the alphabet size k. In Table 5 we present known upper-bounds (leading term) on the redundancy of the algorithms mentioned above. As can be seen, the CTW algorithm enjoys the tightest bound. Note that there exist sequential prediction algorithms that enjoy other types of performance guarantees. One example is the *probabilistic suffix trees* (PST) algorithm (Ron et al., 1996). The PST is a well-known algorithm that is mainly used in the bioinformatic community (see, e.g., Bejerano and Yona, 2001). The algorithm enjoys a PAC-like performance guarantee with respect to the class of VMMs (which is valid only if the predicted sequence was generated by a VMM).

Algorithm	Per Symbol Point-wise Redundancy	Comparison Class	Source
LZ78	$O(1/\log n)$	Markov Sources	Savari (1997); Kieffer and Yang (1999); Potapov (2004)
CTW	$\frac{\lvert\mathcal{S}\rvert(\lvert\Sigma\rvert-1)}{2n}\log n$	Markov sources	Willems et al. (1995); Willems (1998)
BWT	$\frac{\lvert\mathcal{S}\rvert(\lvert\Sigma\rvert+1)}{2n}\log n$ (average redundancy)	D-order Markov sources	Effros et al. (2002)
Grammar Based	$O(\log\log n/\log n)$	Ergodic sources	Yang and Kieffer (2000)
Asymptotic Lower Bound	$\frac{\lvert\mathcal{S}\rvert(\lvert\Sigma\rvert-1)}{2n}\log n$	D-order Markov sources	Shtarkov (1987)

Table 5: Point-wise redundancy (leading term) of several universal lossless compression (and prediction) algorithms. The predicted sequence is of length n. Note that the stated BWT redundancy matches the *average redundancy*; hence, it bounds the BWT point-wise redundancy from below.

8. Concluding Remarks

Our main result is the first redundancy bound for the DECO algorithm. Our bounding technique can be adapted to DECO-like decomposition schemes using any binary predictor that has a (binary) point-wise redundancy bound with respect to VMMs. To the best of our knowledge, our bound for the Huffmann decomposition algorithm (proposed by Volf) is the tightest known for prediction under the log-loss and therefore, for lossless compression. This result provides a compelling justification for the superior empirical performance of the DECO-Huffman predictor/compressor as indicated in several works (see, e.g., Volf, 2002; Sadakane et al., 2000).

Our experiments with random decomposition structures indicate that the DECO scheme is quite robust to the choice of the tree, and even a random tree is likely to outperform the multi-CTW. However, the excellent performance of the Huffman decomposition clearly motivates attempts to optimize it. Our local optimization procedure is able to generate better trees than Huffman's, suggesting that better prediction can be obtained with better optimization of the tree structure. Similar observations were also reported in Volf (2002). Since finding the best decomposition is an NP-hard problem, a very interesting research question is whether one could optimize the DECO redundancy bound over the possible decompositions.

Interestingly, our numerical examples strongly indicate that hierarchical decompositions are better suited to sequential prediction than the standard 'flat' approaches ('one-vs-all' and 'all-pairs') commonly used in supervised learning. This result may further motivate the consideration of hierarchical decompositions in supervised learning (e.g., as suggested by Huo et al., 2002; Cheong et al., 2004; El-Yaniv and Etzion-Rosenberg, 2004).

The fact that the other zero-order estimators can improve the multi-CTW performance (with larger alphabets) motivates further research along these lines. First, it would be interesting to try combining CTW with other zero-order estimators. Second, it would be interesting to analyze the combined algorithm(s), possibly by relying on the worst case results of Orlitsky et al. (2003).

But perhaps the most important research target at this time is the development of a lower bound on the redundancy of predictors for finite (and short) sequences. While the Jacquet and Szpankowski (2004) lower bound is indicative on the asymptotical achievable rates, it is meaningless in the finite (and small) sample context. For example, our bounds, and even the multi-CTW bounds known today, are smaller than the Jacquet and Szpankowski *lower* bound.

9. Acknowledgments

We thank Paul A. Volf and Roee Engelberg for the helpful discussions and Tjalling J. Tjalkens for providing relevant bibliography.

Appendix A. On the KT Estimator - Proof of Theorem 7

We provide a proof for the (worst-case) performance guarantee of the KT estimator as stated in Theorem 7. This proof is based on lecture notes by Catoni (2004).[9]

9. Krichevsky and Trofimov (1981) proved an asymptotic version of Theorem 7 for the average redundancy; Willems et al. (1995) provided a proof for binary alphabets.

Lemma 28 *Consider the case where* KT *counts all the symbols of the sequence* $\mathbf{x}_1^n$ *(i.e.,* $s = \varepsilon$*). Then,*

$$\hat{z}^{\mathrm{KT}}(\mathbf{x}_1^n) = \frac{\Gamma(\frac{k}{2})\prod_{\sigma\in\Sigma}\Gamma(N_\sigma+\frac{1}{2})}{\Gamma(\frac{1}{2})^k\Gamma(\sum_{\sigma\in\Sigma}N_\sigma+\frac{k}{2})}, \tag{48}$$

where $\Gamma(x) = \int_{\mathbb{R}^+} t^{x-1}\exp(-t)dt$ *is the gamma function.*[10]

Proof

The proof is based on the identity $\Gamma(x+1) = x\Gamma(x)$ and on a rewriting of Definition 6,

$$\begin{aligned}
\hat{z}^{\mathrm{KT}}(\mathbf{x}_1^n) &= \hat{z}^{\mathrm{KT}}(x_1^{n-1})\frac{N_{x_n}+1/2}{\sum_{\sigma\in\Sigma}N_\sigma+k/2} \\
&= \frac{\prod_{\sigma\in\Sigma}((1/2)\cdot(1+1/2)\cdot(2+1/2)\cdots(N_\sigma+1/2))}{(k/2)\cdot(1+k/2)\cdot(2+k/2)\cdots(\sum_{\sigma\in\Sigma}N_\sigma+k/2)} \\
&= \frac{\prod_{\sigma\in\Sigma}\left(\frac{1}{\Gamma(\frac{1}{2})^k}\Gamma(N_\sigma+\frac{1}{2})\right)}{\frac{1}{\Gamma(\frac{k}{2})}\Gamma(\sum_{\sigma\in\Sigma}N_\sigma+\frac{k}{2})},
\end{aligned}$$

and (48) is obtained by rearranging the terms. ■

We now provide a proof for Theorem 7. Recall that this theorem states an upper bound of $\frac{k-1}{2}\log n + \log k$ for the worst-case redundancy of $\hat{z}^{\mathrm{KT}}(\mathbf{x}_1^n)$.

Proof It is sufficient to prove that

$$\frac{\hat{z}^{\mathrm{KT}}(\mathbf{x}_1^n)}{\sup_{\mathbf{z}\in\mathcal{Z}}\mathbf{z}(\mathbf{x}_1^n)}n^{\frac{k-1}{2}} \geq \frac{1}{k}. \tag{49}$$

Let $\mathbf{a} = (a_i)_{i=1}^k \in \mathbb{N}^k$ be a vector of arbitrary symbol counts for some sequence $\mathbf{x}_1^n$.[11] For these counts, by Lemma 28, KT would assign the probability, $\frac{\Gamma(\frac{k}{2})}{\Gamma(\frac{1}{2})^k}\frac{\prod_{i=1}^k\Gamma(a_i+\frac{1}{2})}{\Gamma(\sum_{i=1}^k a_i+\frac{k}{2})}$. Let $\mathbf{z}$ be the corresponding empirical distribution: $\mathbf{z}(\mathbf{x}_1^n) = \prod_{i=1}^k\left(\frac{a_i}{\sum_{i=1}^k a_i}\right)^{a_i}$, where $n = \sum_{i=1}^k a_i$. It is well known that, given the counts $\mathbf{a}$, the distribution that maximizes the probability of $\mathbf{x}_1^n$ is $\mathbf{z}$, the maximum likelihood distribution (see, e.g., Cover and Thomas, 1991, Theorem 12.1.2). Thus, taking $\mathbf{z} = \arg\max_{z'\in\mathcal{Z}}z'(\mathbf{x}_1^n)$, inequality (49) becomes

$$\Delta(\mathbf{a}) = \frac{\Gamma(\frac{k}{2})}{\Gamma(\frac{1}{2})^k}\frac{\prod_{i=1}^k\Gamma(a_i+\frac{1}{2})}{\Gamma(\sum_{i=1}^k a_i+\frac{k}{2})}\frac{(\sum_{i=1}^k a_i)^{\sum_i a_i+\frac{k-1}{2}}}{\prod_{i=1}^k a_i^{a_i}} \geq \frac{1}{k}.$$

We have to show that for any $\mathbf{a} \neq (0,0,\ldots,0)$, $\Delta(\mathbf{a}) \geq \frac{1}{k}$.

Observe that $\Delta(\mathbf{a})$ is invariant under any permutation of the coordinations of $\mathbf{a}$. Also note that, by the identity, $\Gamma(x+1) = x\Gamma(x)$,

$$\Delta((1,0,\ldots,0)) = \frac{\Gamma(\frac{k}{2})\Gamma(1+\frac{1}{2})}{\Gamma(\frac{1}{2})\Gamma(1+\frac{k}{2})} = \frac{1}{k}.$$

10. It can be shown that $\Gamma(1) = 1$ and that $\Gamma(x+1) = x\Gamma(x)$. Therefore, if $n \geq 1$ is an integer, $\Gamma(n+1) = n!$. For further information see, e.g., Courant and John (1989).
11. In information theory $\mathbf{a}$ is called a *type*. See, e.g., Cover and Thomas (1991, Chapter 12).

It is sufficient to prove that for any $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ with $a_1 > 1$, $\Delta(\mathbf{a}) \geq \Delta((a_1 - 1, a_2, \ldots, a_k))$. Observe that

$$\Delta(\mathbf{a}) = \Delta(a_1 - 1, a_2, \ldots, a_k) \frac{(a_1 - \frac{1}{2})(a_1 - 1)^{a_1 - 1}}{a_1^{a_1}} \frac{n^{n + \frac{k-1}{2}}}{(n + \frac{k}{2} - 1)(n-1)^{n-1+\frac{k-1}{2}}}.$$

Thus, it is enough to show that

$$\frac{(a_1 - \frac{1}{2})(a_1 - 1)^{a_1 - 1}}{a_1^{a_1}} \frac{n^{n + \frac{k-1}{2}}}{(n + \frac{k}{2} - 1)(n-1)^{n-1+\frac{k-1}{2}}} \geq 1, \qquad \text{where } a_1 \geq 1, n \geq 2.$$

This can be done by showing that

$$\begin{aligned} f(t) &= \log\left(\frac{(t - \frac{1}{2})(t-1)^{t-1}}{t^t}\right) \geq -1; \\ g(q) &= \log\left(\frac{q^{q + \frac{k-1}{2}}}{(q + \frac{k}{2} - 1)(q-1)^{q-1+\frac{k-1}{2}}}\right) \geq 1. \end{aligned}$$

Recall that $\lim_{x \to +\infty} (1 + \frac{y}{x})^x = e^y$ and observe that,

$$\begin{aligned} \lim_{t \to +\infty} f(t) &= \lim_{t \to +\infty} \log\left(1 - \frac{1}{2t}\right) + (t-1)\log\left(1 - \frac{1}{t}\right) = -1; \\ \lim_{q \to +\infty} g(q) &= \lim_{q \to +\infty} -\left(q - 1 + \frac{k-1}{2}\right)\log\left(1 - \frac{1}{q}\right) - \log\left(1 + \frac{k-2}{2q}\right) = 1. \end{aligned}$$

We conclude by showing that both functions decreasing monotone to their limits, hence, $f'(t) \leq 0$ and $g'(q) \leq 0$. For $f'(t)$ we next show that it is a a non-decreasing function (i.e., $f''(t) \geq 0$) that is bounded from above by zero.

$$\begin{aligned} f'(t) &= \frac{1}{t - \frac{1}{2}} + \log\left(\frac{t-1}{t}\right); \\ \lim_{t \to +\infty} f'(t) &= 0; \\ f''(t) &= \frac{-1}{(t - \frac{1}{2})^2} + \frac{1}{t(t-1)}; \\ &= \frac{-1}{(t - \frac{1}{2})^2} + \frac{1}{(t - \frac{1}{2})^2 - \frac{1}{4}} \geq 0. \end{aligned}$$

Therefore, $f'(t) \geq 0$ for any $t > 1$. In a similar manner,

$$\begin{aligned} g'(q) &= \log\left(\frac{q}{1-q}\right) + \frac{k-1}{2}\left(\frac{1}{q} - \frac{1}{q-1}\right) - \frac{1}{q + \frac{k}{2} - 1}; \\ \lim_{q \to +\infty} g'(q) &= 0; \\ g''(q) &\geq 0. \end{aligned}$$

■

Appendix B. Zero Order Estimators

In this appendix we describe the sequential zero-order estimators: Good-Turing, "Improved add-one", "improved Good-Turing" by their "next-symbol" probability assignment. These estimators are compared along with KT in Section 6.

Recall that, by the chain-rule, $\hat{2}(\mathbf{x}_1^n) = \prod_{t=0}^{n-1} \hat{2}(x_{t+1}|\mathbf{x}_1^t)$, where $\mathbf{x}_1^0$ is the empty sequence. Hence, it is sufficient to define the next-symbol prediction, $\hat{2}(x_{t+1}|\mathbf{x}_1^t)$, which is based on the symbol counts N_σ in $\mathbf{x}_1^t$. We require the following definition. Let $\mathbf{x}_1^t$ be a sequence. Define a_m to be the number of symbols that appear exactly m times in $\mathbf{x}_1^t$, i.e., $a_m = |\{\sigma \in \Sigma : N_\sigma = m\}|$. We denote the "improved add-one" estimator by $\hat{2}^{+\underline{1}}$ and the "improved GT" by $\hat{2}^{\mathrm{GT}^*}$.[12]

The Good-Turing (GT) estimator (Good, 1953) is well known and most commonly used in language modeling for speech recognition (see, e.g., Chen and Goodman, 1996).[13] The next-symbol probability generated by GT is

$$\hat{2}^{\mathrm{GT}}(\sigma|\mathbf{x}_1^t) \quad = \quad \frac{1}{S_{t+1}} \times \begin{cases} \frac{a'_1}{t \times a_0}, & \text{if } N_\sigma = 0; \\ \frac{N_\sigma+1}{t} \frac{a'_{N_\sigma+1}}{a'_{N_\sigma}}, & \text{otherwise,} \end{cases} \tag{50}$$

where a'_m is a smoothed version of a_m and S_{t+1} is a normalization factor.[14] In the following experiments we used the simple smoothing suggested by Orlitsky et al. (2003) where $a'_m = \max(a_m, 1)$.

Denote by m the number of distinct symbols in $\mathbf{x}_1^t$ (i.e., $m = \sum_{i=1}^t a_i$). The next-symbol probability of the improved add-one estimator is

$$\hat{2}^{+\underline{1}}(\sigma|\mathbf{x}_1^t) \quad = \quad \frac{1}{S_{t+1}} \times \begin{cases} \frac{m+1}{a_0}, & \text{if } N_\sigma = 0; \\ (t-m+1)\frac{N_\sigma}{t}, & N_\sigma > 0, \end{cases} \tag{51}$$

where S_{t+1} is a normalization factor.

For any natural number c, define the function $f_c(a) = \max(a, c)$. Also define the integer-sequence $c_n = \lceil n^{1/3} \rceil$. The next-symbol probability assigned by the improved GT estimator is

$$\hat{2}^{\mathrm{GT}^*}(\sigma|\mathbf{x}_1^t) \quad = \quad \frac{1}{S_{t+1}} \times \begin{cases} \frac{f_{c_{t+1}}(a_1+1)}{a_0}, & \text{if } N_\sigma = 0; \\ (N_\sigma+1)\frac{f_{c_{t+1}}(a_{N_\sigma+1})}{f_{c_{t+1}}(a_{N_\sigma})}, & \text{otherwise,} \end{cases} \tag{52}$$

where S_{t+1} is a normalization factor. The improved GT estimator ($\hat{2}^{\mathrm{GT}^*}$) is optimal with respect to the worst-case criterion of Orlitsky et al. (2003).

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

12. In Orlitsky et al. (2003) the $\hat{2}^{+\underline{1}}$ estimator is denoted by $q_{+1'}$ and $\hat{2}^{\mathrm{GT}^*}$ by $q_{1/3}$.
13. I. J. Good and A. M. Turing used this estimator to break the Enigma Cipher (Hodges, 2000) during World War II.
14. Orlitsky et al. mention that Turing had an intuitive motivation for this estimator. Unfortunately, this explanation was never published.

G. Bejerano and G. Yona. Variations on probabilistic suffix trees: Statistical modeling and the prediction of protein families. *Bioinformatics*, 17(1):23–43, 2001.

T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice-Hall, Inc., 1990.

M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipement Corporation, 1994.

O. Catoni. Statistical learning theory and stochastic optimization. *Lecture Notes in Mathematics*, 1851, 2004.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

S. Cheong, S. H. Oh, and S. Lee. Support vector machines with binary tree architecture for multi-class classification. *Neural Information Processing - Letters and Reviews*, 2(3):47–51, March 2004.

J. G. Cleary and W. J. Teahan. Experiments on the zero frequency problem. In *DCC '95: Proceedings of the Conference on Data Compression*, page 480, Washington, DC, USA, 1995. IEEE Computer Society.

R. Courant and F. John. *Introduction to Calculus and Analysis*. Springer-Verlag, 1989.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

M. Effros, K. Visweswariah, S. R. Kulkarni, and S. Verdu. Universal lossless source coding with the Burrows Wheeler transform. *IEEE Transactions on Information Theory*, 48(5):1061–1081, 2002.

R. El-Yaniv and N. Etzion-Rosenberg. Hierarchical multiclass decompositions with application to authorship determination. Technical Report CS-2004-15, Technion - Israel Institute of Technology, March 2004.

Y. Freund. Predicting a binary sequence almost as well as the optimal biased coin. *Information and Computation*, 182(2):73–94, 2003.

C. R. Glassey and R. M. Karp. On the optimality of Huffman trees. *SIAM Journal on Applied Mathematics*, 31(2):368–378, September 1976.

I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 1953.

D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

D. P. Helmbold and R. E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997.

A. Hodges. *Alan Turing: the enigma*. Walker and Co., 2000.

X. Huo, J. Chen, S. Wang, and K. L. Tsui. Support vector trees: Simultaneously realizing the principles of maximal margin and maximal purity. Technical report, The Logistics Institute, Georgia Tech, and Asia Pacific, National University of Singapore, 2002.

P. Jacquet and W. Szpankowski. Markov types and minimax redundancy for Markov sources. *IEEE Transactions on Information Theory*, 50:1393–1402, 2004.

J. C. Kieffer and E. H. Yang. A simple technique for bounding the pointwise redundancy of the 1978 Lempel-Ziv algorithm. In *DCC '99: Proceedings of the Conference on Data Compression*, page 434. IEEE Computer Society, 1999.

R. Krichevsky and V. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27:199–207, 1981.

P. Laplace. Philosophical essays on probabilities. *Springer-Verlag*, 1995. Translated by A. Dale from the 5th (1825) edition.

D. McAllester and R. Schapire. On the convergence rate of Good-Turing estimators. In *In Proceedings of the Thirteenth annual conference on computational learning theory*, 2000.

N. Merhav and M. Feder. Universal prediction. *IEEE Transactions on Information Theory*, 44(6): 2124–2147, 1998.

A. Orlitsky, N. P. Santhanam, and J. Zhang. Always Good Turing: Asymptotically optimal probability estimation. *Science*, 302(5644):427–431, October 2003.

C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

V. N. Potapov. Redundancy estimates for the Lempel-Ziv algorithm of data compression. *Discrete Applied Mathematics*, 135(1-3):245–254, 2004. ISSN 0166-218X.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2–3):117–149, 1996.

K. Sadakane, T. Okazaki, and H. Imai. Implementing the context tree weighting method for text compression. In *Data Compression Conference*, pages 123–132, 2000.

S. A. Savari. Redundancy of the Lempel-Ziv incremental parsing rule. *IEEE Transactions on Information Theory*, 43:9–21, 1997.

D. Shkarin. PPM: One step to practicality. In *Data Compression Conference*, pages 202–212, 2002.

Y. Shtarkov. Universal sequential coding of single messages. *Problems in Information Transmission*, 23:175–186, 1987.

N. J. A. Sloane and S. Plouffe. *The Encyclopedia of Integer Sequences*. Academic Press, 1995.

T. J. Tjalkens, Y. Shtarkov, and F. M. J. Willems. Context tree weighting: Multi-alphabet sources. In *Proc. 14th Symp. on Info. Theory, Benelux*, pages 128–135, 1993.

T. J. Tjalkens, P. A. Volf, and F. M. J. Willems. A context-tree weighting method for text generating sources. In *Data Compression Conference*, page 472, 1997.

T. J. Tjalkens, F. M. J. Willems, and Y. Shtarkov. Multi-alphabet universal coding using a binary decomposition context tree weighting algorithm. In *Proc. 15th Symp.on Info. Theory, Benelux*, pages 259–265, 1994.

P. A. Volf. *Weighting Techniques in Data Compression Theory and Algorithms*. PhD thesis, Technische Universiteit Eindhoven, 2002.

V. Vovk. Aggregating strategies. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 371–383, 1990.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

F. M. J. Willems. The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44(2):792–798, March 1998.

F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, pages 653–664, 1995.

I. H. Witten and T. C. Bell. The zero-frequency problem: estimating the probabilities of novelevents in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

Q. Xie and A. R. Barron. Asymptotic minimax regret for data compression, gambling, and prediction. *IEEE Transactions on Information Theory*, 46(2):431–445, 2000.

E. H. Yang and J. C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. part one: Without context models. *IEEE Transactions on Information Theory*, 46(3):755–777, 2000.

J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337–343, May 1977.

J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.

Journal of Machine Learning Research 7 (2006) 413–427 Submitted 5/05; Revised 1/06; Published 2/06

Geometric Variance Reduction in Markov Chains: Application to Value Function and Gradient Estimation

Rémi Munos REMI.MUNOS@POLYTECHNIQUE.FR
Centre de Mathématiques Appliquées
Ecole Polytechnique
91128 Palaiseau, France

Editor: Shie Mannor

Abstract

We study a variance reduction technique for Monte Carlo estimation of functionals in Markov chains. The method is based on designing *sequential control variates* using successive approximations of the function of interest V. Regular Monte Carlo estimates have a variance of $O(1/N)$, where N is the number of sample trajectories of the Markov chain. Here, we obtain a geometric variance reduction $O(\rho^N)$ (with $\rho < 1$) up to a threshold that depends on the approximation error $V - \mathcal{A}V$, where $\mathcal{A}$ is an *approximation operator* linear in the values. Thus, if V belongs to the right approximation space (i.e. $\mathcal{A}V = V$), the variance decreases geometrically to zero.

An immediate application is value function estimation in Markov chains, which may be used for policy evaluation in a policy iteration algorithm for solving Markov Decision Processes.

Another important domain, for which variance reduction is highly needed, is gradient estimation, that is computing the sensitivity $\partial_\alpha V$ of the performance measure V with respect to some parameter α of the transition probabilities. For example, in policy parametric optimization, computing an estimate of the policy gradient is required to perform a gradient optimization method.

We show that, using two approximations for the *value function* and the *gradient*, a geometric variance reduction is also achieved, up to a threshold that depends on the approximation errors of both of those representations.

1. Introduction

We consider a Markov chain over a finite state space $\mathcal{X}$ defined by the transition matrix P. Note that although we consider a finite state space for simplicity, this work can be easily extended to the case of infinite state spaces (countable or continuous). Write $X(x)$ a trajectory $(x_t)_{t\geq 0}$ starting at a state $x_0 = x$. Let $\Psi(r, X(x))$ be a functional that depends on some function $r : \mathcal{X} \to \mathbb{R}$ and the trajectory $X(x)$, and write $V(x)$ the expectation of the functional that we wish to evaluate:

$$V(x) = \mathbb{E}[\Psi(r, X(x))]. \tag{1}$$

Here, the quantity of interest V is expressed in terms of a **probabilistic representation**, as an expectation of a functional that depends on trajectories. We will consider a functional $\Psi(r,\cdot)$ that is linear in r, and such that its expectation V may equivalently be expressed in terms of a **solution to a linear system**

$$\mathcal{L}V = r, \tag{2}$$

(where r and V are considered as column vectors) with $\mathcal{L}$ an invertible linear operator (matrix).

Such an example of Ψ is the sum of discounted rewards r received along the trajectory:

$$\Psi(r, X(x)) = \sum_{t \geq 0} \gamma^t r(x_t). \tag{3}$$

with $0 < \gamma < 1$ being a discount factor. In that case, V is the solution to the Bellman equation (2) with $\mathcal{L} = I - \gamma P$. Indeed, using matrix notations, $V = \sum_{t \geq 0} \gamma^t P^t r = (I - \gamma P)^{-1} r$.

A regular Monte-Carlo (MC) method would estimate $V(x)$ by sampling N independent trajectories $\{X^n(x)\}_{1 \leq n \leq N}$ starting from x and calculate the average $\frac{1}{N} \sum_{n=1}^{N} \Psi(r, X^n(x))$. The variance of such an estimator is of order $1/N$. Variance reduction is crucial since the numerical approximation error of the quantity of interest is directly related to the variance of its estimate.

Variance reduction techniques include importance sampling, correlated sampling, control variates, antithetic variates and stratified sampling, see e.g. (Hammersley and Handscomb, 1964; Halton, 1970). Geometric variance reduction rates have been obtained by processing these variance reduction methods iteratively, the so-called *sequential* (or *recursive*) *Monte-Carlo*. Examples include adaptive importance sampling (Kollman et al., 1999) and what Halton called the "Third Sequential Method" (Halton, 1994) based on sequential correlated sampling and control variates. This approach has been recently developed in (Maire, 2003) for numerical integration and, more related to our work, applied to (continuous time) Markov processes in (Gobet and Maire, 2005).

The idea is to replace the expectation of $\Psi(r, \cdot)$ by the expectation of $\Psi(r - \mathcal{L}W, \cdot)$ for some function W close to V. From the linearity of Ψ and the equivalence between the representations (1) and (2), for any W, one has

$$V(x) = W(x) + \mathbb{E}[\Psi(r - \mathcal{L}W, X(x))].$$

Thus, if W is a good approximation of V, the residual $r - \mathcal{L}W$ is small, and the variance is low.

In the sequential method described in this paper, we use successive approximations V_n of V to estimate by Monte Carlo a correction E_n using the residual $r - \mathcal{L}V_n$ in Ψ, which is used to process a new approximation V_{n+1}. We consider an approximation operator $\mathcal{A}$ that is *linear in the values*. We show that (for enough sample trajectories at each iteration) the variance of the estimator has a geometric rate ρ^N (with $\rho < 1$, and N the total number of sampled trajectories) until some threshold is reached, whose value is related to the approximation error $\mathcal{A}V - V$.

An interesting extension of this idea concerns the estimation of the gradient $\partial_\alpha V$ of V with respect to (w.r.t.) some parameter α of the transition matrix P. A useful application of such sensitivity analysis appears in policy gradient estimation. An optimal control problem may be approximated by a parametric optimization problem in a given space of parameterized policies. Thus, the transition matrix P depends on some (possible multidimensional) policy parameter α. In order to apply gradient methods to search for a local maximum of the performance in the parameter space, one wishes to estimate the policy gradient, i.e. the sensitivity $Z = \partial_\alpha V$ of the performance measure with respect to α. The gradient may be expressed as an expectation $Z(x) = \mathbb{E}[\Phi(r, X(x))]$, using the so-called *likelihood ratio* or *score method* (Reiman and Weiss, 1986; Glynn, 1987; Williams, 1992; Baxter and Bartlett, 2001; Marbach and Tsitsiklis, 2003). The gradient Z is also the solution to a linear system

$$\mathcal{L}Z = -\partial_\alpha \mathcal{L}\, \mathcal{L}^{-1} r = -\partial_\alpha \mathcal{L} V. \tag{4}$$

(note that the derivative operator ∂_α only applies to $\mathcal{L}$). Indeed, since V solves $V = \mathcal{L}^{-1} r$, we have $Z = \partial_\alpha V = -\mathcal{L}^{-1} \partial_\alpha \mathcal{L}\, \mathcal{L}^{-1} r$. For example, in the infinite horizon, discounted case (3), we have

$\mathcal{L} = I - \gamma P$, thus $\partial_\alpha \mathcal{L} = -\gamma \partial_\alpha P$ and

$$Z = \gamma (I - \gamma P)^{-1} \partial_\alpha P \, (I - \gamma P)^{-1} r = \sum_{t \geq 0} \gamma^{t+1} P^t \partial_\alpha P \sum_{s \geq 0} \gamma^s P^s r.$$

The functional Φ may thus be defined as

$$\Phi(r, X(x)) \quad = \quad \sum_{t \geq 0} \gamma^{t+1} \frac{\partial_\alpha P(x_t, x_{t+1})}{P(x_t, x_{t+1})} \sum_{s \geq 0} \gamma^s r(x_{s+t+1}), \tag{5}$$

which may be rewritten as

$$\Phi(r, X(x)) = \sum_{t \geq 0} \gamma^t r(x_t) \sum_{s=0}^{t-1} \frac{\partial_\alpha P(x_s, x_{s+1})}{P(x_s, x_{s+1})}.$$

We show that, using two approximations V_n and Z_n of the *value function* and the *gradient*, a geometric variance reduction is also achieved, up to a threshold that depends on the approximation errors of both of those representations.

Numerical experiments on a simple Gambler's ruin problem illustrate the approach.

2. Value Function Estimation

We first describe the approximation operator *linear in the values* considered here, then describe the algorithm, and state the main result on geometric variance reduction.

2.1 Approximation Operator $\mathcal{A}$

We consider a fixed set of J *representative states* $\mathcal{X}_J := \{x_j \in \mathcal{X}\}_{1 \leq j \leq J}$ and *basis functions* $\{\phi_j : \mathcal{X} \to \mathbb{R}\}_{1 \leq j \leq J}$. The linear approximation operator $\mathcal{A}$ maps any function $W : \mathcal{X}_J \to \mathbb{R}$ to the function $\mathcal{A}W : \mathcal{X} \to \mathbb{R}$, according to

$$\mathcal{A}W(x) = \sum_{j=1}^{J} W(x_j) \phi_j(x). \tag{6}$$

With a slight abuse of notation, for any function $W : \mathcal{X} \to \mathbb{R}$, we define $\mathcal{A}W : \mathcal{X} \to \mathbb{R}$ similarly from the values of W at $\mathcal{X}_J$. This kind of function approximation includes:

- **Linear approximation**, for example with *Spline*, *Polynomial*, *Radial Basis*, *Fourier* or *Wavelet* decomposition. $\mathcal{A}W$ is the projection of a function W onto the space spanned by a set of functions $\{\psi_k : \mathcal{X} \to \mathbb{R}\}_{1 \leq k \leq K}$, i.e. the function minimizing some norm (induced by a discrete inner product $\langle f, g \rangle := \sum_{j=1}^{J} \mu_j f(x_j) g(x_j)$, for some distribution μ over $\mathcal{X}_J$):

  $$\min_{\alpha \in \mathbb{R}^K} \left|\left| \sum_{k=1}^{K} \alpha_k \psi_k - W \right|\right|^2.$$

  The solution α solves the linear system $A\alpha = b$ with A an $K \times K$-matrix of elements $A_{kl} = \langle \psi_k, \psi_l \rangle$ and b a K-vector of components $b_k = \langle W, \psi_k \rangle$. Thus $\alpha_k = \sum_{l=1}^{K} A_{kl}^{-1} \sum_{j=1}^{J} \mu_j \psi_l(x_j) W(x_j)$ and the best fit $\sum_{k=1}^{K} \alpha_k \psi_k$ is thus of type (6) with

  $$\phi_j(x) = \mu_j \sum_{k=1}^{K} \sum_{l=1}^{K} A_{kl}^{-1} \psi_l(x_j) \psi_k(x). \tag{7}$$

- **Non-parametric approximation**, such as *k-nearest neighbors* (where $\phi_j(x) = \frac{1}{k}$ if x has x_j as one of its $k-$nearest neighbors, and $\phi_j(x) = 0$ otherwise), *locally weighted learning* and *Kernel regression* (Atkeson et al., 1997; Hastie et al., 2001), where functions similar to (7) may be derived (with the matrix A being dependent on x through the kernel), and *Support Vector Regression* (when using a quadratic loss function) (Vapnik et al., 1997; Vapnik, 1998).

2.2 The Algorithm

We assume the equivalence between the probabilistic interpretation (1) and the representation as solution to the linear system (2), i.e. for any function $f : \mathcal{X} \to \mathbb{R}$,

$$f(x) = \mathbb{E}[\Psi(\mathcal{L}f, X(x))]. \tag{8}$$

We consider successive approximations $V_n \in \mathbb{R}^J$ of V defined at the states $\mathcal{X}_J = (x_j)_{1\leq j\leq J}$ recursively:

- We initialize $V_0(x_j) = 0$.
- At stage n, we use the values $V_n(x_j)$ to provide a new estimate of $V(x_j)$. Let $E_n(x_j) := V(x_j) - \mathcal{A}V_n(x_j)$ be the approximation error at the states $(x_j)_{1\leq j\leq J}$. From the equivalence property (8), we have: $\mathcal{A}V_n(x) = \mathbb{E}[\Psi(\mathcal{L}\mathcal{A}V_n, X(x))]$. Thus, by linearity of Ψ w.r.t. its first variable,
$$E_n(x_j) = \mathbb{E}[\Psi(r - \mathcal{L}\mathcal{A}V_n, X(x_j))].$$
Now, we use a Monte Carlo technique to estimate $E_n(x_j)$ at each representative state x_j, using M trajectories $(X^{n,m}(x_j))_{1\leq m\leq M}$: we calculate the average
$$\widehat{E_n}(x_j) := \frac{1}{M}\sum_{m=1}^{M} \Psi(r - \mathcal{L}\mathcal{A}V_n, X^{n,m}(x_j)),$$
and define the new approximation at the states $\mathcal{X}_J$:
$$V_{n+1}(x_j) := \mathcal{A}V_n(x_j) + \widehat{E_n}(x_j). \tag{9}$$

Remark 1 *Notice that there is a slight difference between this algorithm and that of (Gobet and Maire, 2005), which may be written*

$$V_{n+1}(x_j) = V_n(x_j) + \mathcal{A}\Big[\frac{1}{M}\sum_{m=1}^{M} \Psi(r - \mathcal{L}V_n, X^{n,m}(x_j))\Big].$$

Our formulation enables us to avoid the assumption of the idempotent property for $\mathcal{A}$ (i.e. that $\mathcal{A}^2 = \mathcal{A}$) which does not hold in general in non-parametric approximation (e.g. in k-nearest neighbors, for $k \geq 2$) and guarantees that V_n is an unbiased estimate of V, for all n, as showed in the next paragraph.

2.3 Properties of the Estimates V_n

We write the conditional expectations and variances:

$$\mathbb{E}^n[Y] = \mathbb{E}[Y|X^{p,m}(x_j), 0 \leq p < n, 1 \leq m \leq M, 1 \leq j \leq J]$$

and $\mathrm{Var}^n[Y] = \mathbb{E}^n[Y^2] - (\mathbb{E}^n[Y])^2$. We have the following properties on the estimates:

Expectation of V_n. From the definition (9),

$$\mathbb{E}^n[V_{n+1}(x_j)] = \mathcal{A}V_n(x_j) + E_n(x_j) = V(x_j).$$

Thus $\mathbb{E}[V_n(x_j)] = V(x_j)$ for all $n \geq 1$: the approximation $V_n(x_j)$ is an unbiased estimate of $V(x_j)$.

Variance of V_n. Write $v_n = \sup_{1\leq j\leq J} \mathrm{Var}\, V_n(x_j)$. The following result (whose proof is provided in Appendix A) expresses that for large enough values of M, the variance decreases geometrically with n.

Theorem 2 *We have*

$$v_{n+1} \leq \rho_M v_n + \frac{2}{M}\mathcal{V}_\Psi(V - \mathcal{A}V) \tag{10}$$

with $\rho_M = \frac{2}{M}\left(\sum_{j=1}^J \sqrt{\mathcal{V}_\Psi(\phi_j)}\right)^2$, *using the notation*

$$\mathcal{V}_\Psi(f) := \sup_{1\leq j\leq J} \mathit{Var}\, \Psi(\mathcal{L}f, X(x_j)).$$

Thus, for large enough values of M, (i.e. whenever $\rho_M < 1$), $(v_n)_n$ decreases geometrically at rate ρ_M, up to the threshold

$$\limsup_{n\to\infty} v_n \leq \frac{1}{1-\rho_M}\frac{2}{M}\mathcal{V}_\Psi(V - \mathcal{A}V).$$

If V belongs to the space of functions that are representable by $\mathcal{A}$, i.e. $\mathcal{A}V = V$, then the variance geometrically decreases to 0 at rate ρ^N with $\rho := \rho_M^{1/M}$ and N being the total number of sample trajectories per state x_j (i.e. N is the product of the number n of iterations by the number M of trajectories per iteration and state x_j).

Notice that the threshold depends on the variance of Ψ for the function $\mathcal{L}(V - \mathcal{A}V) = r - \mathcal{L}\mathcal{A}V$, the residual of the representation (by $\mathcal{A}$) of V. Notice also that this threshold depends on $V - \mathcal{A}V$ only at states reached by the trajectories $\{X(x_j)\}_{x_j\in\mathcal{X}_J}$: a uniform (over the whole domain) representation of V is not required.

Of course, once the threshold is reached, a further convergence of $O(1/N)$ can be obtained thereafter, using regular Monte Carlo.

2.4 Example: The Infinite Horizon, Discounted Case

Let us illustrate the sequential control variates algorithm to value function estimation in Markov chains in the infinite horizon, discounted case (3). The value function

$$V(x) = \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r(x_t)\Big]$$

solves Bellman's equation: $V = r + \gamma PV$, which may be written as the linear system (2) with $\mathcal{L} = I - \gamma P$.

In the previous algorithm, at stage n, the approximation error $E_n(x_j) = V(x_j) - \mathcal{A}V_n(x_j)$ is therefore the expectation

$$E_n(x_j) = \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t [r(x_t) - \mathcal{A}V_n(x_t) + \gamma P\mathcal{A}V_n(x_t)] | x_0 = x_j\Big]. \tag{11}$$

We notice that the term $r - \mathcal{A}V_n + \gamma P\mathcal{A}V_n$ is the *Bellman residual* of the approximation $\mathcal{A}V_n$. The estimate thus has zero variance if this approximation happens to be the value function. Following the algorithm, the next approximation V_{n+1} is defined by (9) with $\widehat{E_n}(x_j)$ being a Monte Carlo estimate of (11).

Remark 3 *Note that the expectation operator P may not be easy to process. In model-free learning, it would be interesting to replace the term $P\mathcal{A}V_n(x_t)$ by $\mathcal{A}V_n(x_{t+1})$ in (11) leaving the expectation unchanged. However, this would introduce some additional variance that annihilates the benefit of the technique.*

Nevertheless, the term $P\mathcal{A}V_n$ may actually be computed as $\mathcal{A}'V_n$, where $\mathcal{A}'$ is an approximation operator defined by another set of basis functions $\{\phi'_j := P\phi_j\}_{1\leq j\leq J}$ (i.e. $\phi'_j(x) := \sum_{y\in\mathcal{X}} P(x,y)\phi_j(y)$, $1 \leq j \leq J$). Indeed, for any $W : \mathcal{X}_J \to \mathbb{R}$,

$$P\mathcal{A}W(x) = \sum_{y\in\mathcal{X}} P(x,y) \sum_{j=1}^{J} W(x_j)\phi_j(y) = \sum_{j=1}^{J} W(x_j)\phi'_j(x) = \mathcal{A}'W(x).$$

These functions $\{\phi'_j := P\phi_j\}_{1\leq j\leq J}$ may be precomputed before simulations, or approximated on-line with function approximation techniques.

2.5 Other Examples

Other possible settings include the finite-horizon time, the infinite horizon stochastic shortest path, and average reward problems, briefly described now.

In a *finite-time horizon problem*, the value $V(t,x)$ is time-dependent. So let $X(t,x) = \{x_s\}_{t\leq s\leq T}$ be a trajectory starting from $x \in \mathcal{X}$ at time $t \in \{0,\dots T\}$. Write $\Psi(r,X(t,x)) := \sum_{s=t}^{T} r(x_s)$. The value function $V(t,x) = \mathbb{E}[\Psi(r,X(t,x))]$ solves Bellman's equation

$$V(t,x) = r(x) + \sum_{y\in\mathcal{X}} P(x,y)V(t+1,y), \text{ for } 0 \leq t < T$$

and $V(T,x) = r(x)$. A similar variance reduction method holds in the product space $\{0,\dots,T\} \times \mathcal{X}$. Approximate functions $\mathcal{A}W$ are defined on a grid $\{(t_j,x_j)\}_{1\leq j\leq J}$ over the product space, as a linear combination of basis functions $\{\phi_j(t,x)\}$: for any function W defined on the product space, $\mathcal{A}W(t,x) := \sum_{j=1}^{J} W(t_j,x_j)\phi_j(t,x)$. The variance reduction result of the previous section applies immediately to this case.

In *infinite horizon stochastic shortest path problems*, we usually assume that the reward function is non-negative (or non-positive if it represents a cost function) and that there exists an absorbing state (with a zero reward) that is reached, from any initial state, in finite time with probability 1. The functional is $\Psi(r,X(x)) := \sum_{t\geq 0} r(x_t)$ and the value function V solves Bellman's equation $(I-P)V = r$ with $(I-P)$ being invertible.

The case of *average reward problems* is more subtle and would deserve deeper treatment. We simply provide the idea of the possible application to this case. The functional is $\Psi(r,X(x)) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} r(x_t)$. In aperiodic, recurrent, unichain Markov chains, the average expected gain ρ

$$\rho(x) := \Big(\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} P^t r \Big)(x)$$

is independent from the start state $\rho(x) = \rho$, and satisfies $\rho = \pi r$, where π is the stationary distribution of the chain (π is considered as a row vector), i.e. $\pi P = \pi$. The relative value function $V(x) := \mathbb{E}[\Psi(r-\rho, X(x))]$ solves the equation $(I-P)V = r-\rho$. This equation has several solutions but a unique one V such that $P^{\pi}V = 0$, with P^{π} being the matrix with all rows equal to π.

In this setting, a possible extension of the variance reduction technique would process two approximations ρ_n and V_n of the average reward ρ and the relative value function V, respectively.

2.6 Numerical Experiment

We consider the *Gambler's ruin problem* described in (Kollman et al., 1999): a gambler with i dollars bets repeatedly against the house, whose initial capital is $L-i$. Each bet is one dollar and the gambler has probability p of winning. The state space is $\mathcal{X} = \{0, \dots, L\}$ and the transition matrix P is defined, for $i, j \in \mathcal{X}$, by

$$P_{ij} = \begin{cases} p, & \text{if } j-i = 1 \text{ and } 0 < i < L, \\ 1-p, & \text{if } i-j = 1 \text{ and } 0 < i < L, \\ 0, & \text{otherwise.} \end{cases}$$

Betting continues until either the gambler is ruined ($i = 0$) or he has "broken the bank" ($i = L$) (thus 0 and L are terminal states). This is an infinite-horizon time stochastic shortest path problem. We are interested in computing the probability of the gambler's eventual ruin $V(i)$ when starting from initial fortune i. We thus define the function $r(0) = 1$ and $r(i \neq 0) = 0$. The value function V solves the Bellman equation $(I-P)V = r$, and its value is

$$V(i) = \frac{\lambda^i - \lambda^L}{1-\lambda^L}, \text{ for } i \in \mathcal{X}, \tag{12}$$

with $\lambda := \frac{1-p}{p}$ when $p \neq 0.5$, and $V(i) = 1 - i/L$ for $p = 0.5$. The representative states are $X_J = \{1, 7, 13, 19\}$ (here $L = 20$). We consider two linear function approximations $\mathcal{A}_1$ and $\mathcal{A}_2$ that are projection operators (minimizing the L_2 norm at the states X_J) onto the space spanned by a set of functions $\{\psi_k : \mathcal{X} \to \mathbb{R}\}_{1 \leq k \leq K}$. $\mathcal{A}_1$ uses $K = 2$ functions $\psi_1(i) = 1, \psi_2(i) = \lambda^i, i \in \mathcal{X}$, whereas $\mathcal{A}_2$ uses $K = 4$ functions $\psi_1(i) = 1, \psi_2(i) = i, \psi_3(i) = i^2, \psi_4(i) = i^3$, $i \in \mathcal{X}$. Notice that V is representable by $\mathcal{A}_1$ (i.e. $\mathcal{A}_1 V = V$) but not by $\mathcal{A}_2$. We chose $p = 0.51$.

We ran the algorithm with $\mathcal{L} = I - P$ (which is an invertible matrix). At each iteration, we used $M = 100$ simulations per state. Figure 1 shows the L_∞ approximation error ($\max_{j \in X_J} |V(j) - V_n(j)|$) in logarithmic scale, as a function of the iteration number $1 \leq n \leq 10$. This approximation error (which is the true quantity of interest) is directly related to the variance of the estimates V_n.

For the approximation $\mathcal{A}_1$, we observe the geometric convergence to 0, as predicted in Theorem 2. It takes less than 10×100 simulations per state to reach an error of 10^{-15}. Using $\mathcal{A}_2$, the error does not decrease below some threshold $\simeq 2.10^{-5}$ due to the approximation error $V - \mathcal{A}_2 V$. This threshold is reached using about 5×100 simulations per state. For comparison, usual MC reaches an error of 10^{-4} with 10^8 simulations per state.

The variance reduction obtained when using such sequential control variates is thus considerable.

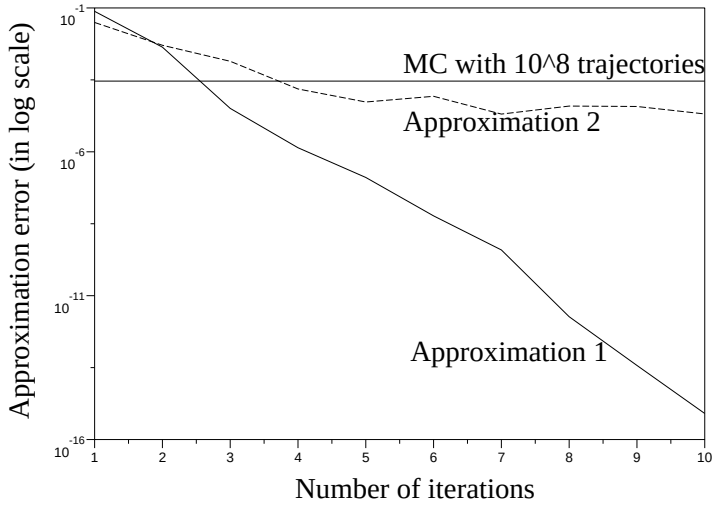


Figure 1: Approximation error for regular MC and sequential control variate algorithm using two approximations $\mathcal{A}_1$ and $\mathcal{A}_2$, as a function of the number of iterations.

3. Gradient Estimation

Here, we assume that the transition matrix P depends on some parameter α, and that we wish to estimate the sensitivity of $V(x) = \mathbb{E}[\Psi(r, X(x))]$ with respect to α, which we write $Z(x) := \partial_\alpha V(x)$.

An example of interest consists in solving approximately a Markov Decision Problem by searching for a feedback control law in a given class of parameterized stochastic policies. The optimal control problem is replaced by a parametric optimization problem, which may be solved (at least in order to find a local optimum) using gradient methods. Thus we are interested in estimating the gradient of the performance measure w.r.t. the parameter of the policy. In this example, the transition matrix P would be the transition matrix of the MDP combined with the parameterized stochastic policy.

As mentioned in the introduction, the gradient may be expressed as an expectation $Z(x) = \mathbb{E}[\Phi(r, X(x))]$ (using the so-called *likelihood ratio* or *score method* (Reiman and Weiss, 1986; Glynn, 1987; Williams, 1992; Baxter and Bartlett, 2001; Marbach and Tsitsiklis, 2003)) where $\Phi(r, X(x))$ is also a functional that depends on the trajectory $X(x)$, and that is linear in its first variable. For example, in the discounted case (3), the functional Φ is given by (5). The variance is usually high, thus variance reduction techniques are highly needed (Greensmith et al., 2005).

The gradient Z is also the solution to the linear system (4). Unfortunately, this linear expression is not of the form (2) since $\partial_\alpha \mathcal{L}$ is not invertible, which prevents us from using directly the method of the previous section.

However, the linear equation (4) provides us with another representation for Z in terms of a probabilistic representation:

$$Z(x) = \mathbb{E}[\Phi(r, X(x))] = \mathbb{E}[\Psi(-\partial_\alpha \mathcal{L} V, X(x))]. \tag{13}$$

We may extend the previous algorithm to the estimation of Z by using two representations: V_n and Z_n. The approximation V_n of V is updated from Monte-Carlo estimation of the residual $r - \mathcal{L}V_n$, and Z_n, which approximates Z, is updated from the gradient residual $-\partial_\alpha \mathcal{L}V_n - \mathcal{L}Z_n$ built from the current V_n. This approach may be related to the so-called *Actor-Critic algorithms* (Konda and Borkar, 1999; Sutton et al., 2000), which use the representation (13) with an approximation of the value function.

A geometric variance reduction is also achieved, up to a threshold that depends on the approximation errors of both of those representations.

Finally, we present a variance reduction technique that only makes use of the gradient representation Z_n (which may be useful for Partially Observable MDPs) but at the cost of a variance increase.

3.1 The Algorithm

Although the approximation operators for V and Z may be different in practice (they may use different sets of representative states and basis functions), in this section, we will use the same approximation operator $\mathcal{A}$ for simplicity.

From (13) and the equivalence property (8), we obtain the following representation for Z:

$$\begin{aligned} Z(x) &= \mathcal{A}Z_n(x) + \mathbb{E}\big[\Psi(-\partial_\alpha \mathcal{L}V - \mathcal{L}\mathcal{A}Z_n, X(x))\big] \\ &= \mathcal{A}Z_n(x) + \mathbb{E}\big[\Psi(-\partial_\alpha \mathcal{L}(V - \mathcal{A}V_n), X(x)) - \Psi(\partial_\alpha \mathcal{L}\mathcal{A}V_n + \mathcal{L}\mathcal{A}Z_n, X(x))\big] \\ &= \mathcal{A}Z_n(x) + \mathbb{E}\big[\Phi(r - \mathcal{L}\mathcal{A}V_n, X(x)) - \Psi(\partial_\alpha \mathcal{L}\mathcal{A}V_n + \mathcal{L}\mathcal{A}Z_n, X(x))\big]. \end{aligned} \tag{14}$$

from which the algorithm is deduced. We consider successive approximations $V_n \in \mathbb{R}^J$ of V and $Z_n \in \mathbb{R}^J$ of Z defined at the states $\mathcal{X}_J = (x_j)_{1 \le j \le J}$.

- We initialize $V_0(x_j) = 0$, $Z_0(x_j) = 0$.
- At stage n, we simulate by Monte Carlo M trajectories $(X^{n,m}(x_j))_{1 \le m \le M}$ and define the new approximations V_{n+1} and Z_{n+1} at the states $\mathcal{X}_J$:

$$\begin{aligned} V_{n+1}(x_j) &= \mathcal{A}V_n(x_j) + \frac{1}{M}\sum_{m=1}^{M} \Psi(r - \mathcal{L}\mathcal{A}V_n, X^{n,m}(x_j)) \\ Z_{n+1}(x_j) &= \mathcal{A}Z_n(x_j) + \frac{1}{M}\sum_{m=1}^{M} \Big[\Phi(r - \mathcal{L}\mathcal{A}V_n, X^{n,m}(x_j)) \\ &\qquad - \Psi(\partial_\alpha \mathcal{L}\mathcal{A}V_n + \mathcal{L}\mathcal{A}Z_n, X^{n,m}(x_j))\Big]. \end{aligned}$$

3.2 Properties of the Estimates V_n and Z_n

Expectation of V_n and Z_n. We have already seen that $\mathbb{E}[V_n] = V$ for all $n > 0$. Now, (14) implies that $\mathbb{E}^n[Z_{n+1}] = Z$, thus $\mathbb{E}[Z_n] = Z$ for all $n > 0$.

Variance of V_n and Z_n. We write $v_n = \sup_{1 \le j \le J} \operatorname{Var} V_n(x_j)$ and $z_n = \sup_{1 \le j \le J} \operatorname{Var} Z_n(x_j)$. The next theorem (proved in Appendix B) states the geometric variance reduction for large enough values of M.

Theorem 4 *We have*

$$\begin{aligned} v_{n+1} &\leq \rho_M v_n + \frac{2}{M}\mathcal{V}_\Psi(V - \mathcal{A}V) \\ z_{n+1} &\leq \rho_M z_n + \frac{2}{M}[c_1(V - \mathcal{A}V, Z - \mathcal{A}Z) + c_2 v_n] \end{aligned}$$

with $\rho_M = \frac{2}{M}\left(\sum_{j=1}^J \sqrt{\mathcal{V}_\Psi(\phi_j)}\right)^2$, *and the coefficients*

$$\begin{aligned} c_1(f,g) &= \left(\sqrt{\mathcal{V}_\Phi(f)} + \sqrt{\mathcal{V}_\Psi(\mathcal{L}^{-1}\partial_\alpha \mathcal{L} f)} + \sqrt{\mathcal{V}_\Psi(g)}\right)^2 \\ c_2 &= \Big[\sum_{j=1}^J \sqrt{\mathcal{V}_\Phi(\phi_j)} + \sqrt{\mathcal{V}_\Psi(\mathcal{L}^{-1}\partial_\alpha \mathcal{L}\phi_j)}\Big]^2, \end{aligned}$$

using the notations $\mathcal{V}_\Psi(f) := \sup_{1\leq j\leq J} Var\ \Psi(\mathcal{L}f, X(x_j))$ *and* $\mathcal{V}_\Phi(f) := \sup_{1\leq j\leq J} Var\ \Phi(\mathcal{L}f, X(x_j))$. *Thus, for large enough values of M, (i.e. whenever* $\rho_M < 1$*), the convergence of* $(v_n)_n$ *and* $(z_n)_n$ *is geometric at rate* ρ_M*, up to the thresholds*

$$\begin{aligned} \limsup_{n\to\infty} v_n &\leq \frac{1}{1-\rho_M}\frac{2}{M}\mathcal{V}_\Psi(V - \mathcal{A}V) \\ \limsup_{n\to\infty} z_n &\leq \frac{1}{1-\rho_M}\frac{2}{M}\Big[c_1(V - \mathcal{A}V, Z - \mathcal{A}Z) + c_2\frac{1}{1-\rho_M}\frac{2}{M}\mathcal{V}_\Psi(V - \mathcal{A}V)\Big]. \end{aligned}$$

Here also, if V and Z are representable by $\mathcal{A}$, then the variance converges geometrically to 0.

3.3 Numerical Experiment

Again we consider the *Gambler's ruin problem* described previously. The transition matrix is parameterized by $\alpha = p$, the probability of winning. The gradient $Z(i) = \partial_\alpha V(i)$ may be derived from (12):

$$Z(i) = \frac{L(1-\lambda^i)\lambda^{L-1} - i(1-\lambda^L)\lambda^{i-1}}{(1-\lambda^L)^2\alpha^2} \text{ for } i \in \mathcal{X},$$

(for $\alpha \neq 0.5$), and $Z(i) = 0$ for $\alpha = 0.5$. Again we use the representative states $X_J = \{1, 7, 13, 19\}$. Here, we consider two possible approximators $\mathcal{A}_1$ and $\mathcal{A}_2$ for the value function representations V_n (as defined previously), and two approximators $\mathcal{A}_2$ and $\mathcal{A}_3$ for the gradient representations Z_n, where $\mathcal{A}_3$ is the projection that uses $K = 3$ functions $\psi_1(i) = 1, \psi_2(i) = \lambda^i, \psi_3(i) = i\lambda^i, i \in \mathcal{X}$. Notice that Z is representable by $\mathcal{A}_3$ but not by $\mathcal{A}_2$. We choose $p = 0.51$ and $M = 1000$.

Figure 2 shows the L_∞ approximation error of Z ($\max_{j\in X_J} |Z(j) - Z_n(j)|$) in logarithmic scale, for the different possible approximations of V and Z.

When both V and Z may be perfectly approximated (i.e. $\mathcal{A}_1$ for V and $\mathcal{A}_3$ for Z) we observe the geometric convergence to 0, as predicted in Theorem 4. The error is around 10^{-14} using a total of 10^4 simulations. When either the value function or the gradient is not representable in the approximation spaces, the error does not decrease below some threshold ($\simeq 3.10^{-3}$ when Z is not representable) reached in 2.10^3 simulations. The threshold is lower ($\simeq 2.10^{-4}$) when Z is representable. For comparison, usual MC reaches an error (for Z) of 3.10^{-3} with 10^8 simulations per state.

The variance reduction of this sequential method compared to regular MC is thus also considerable.

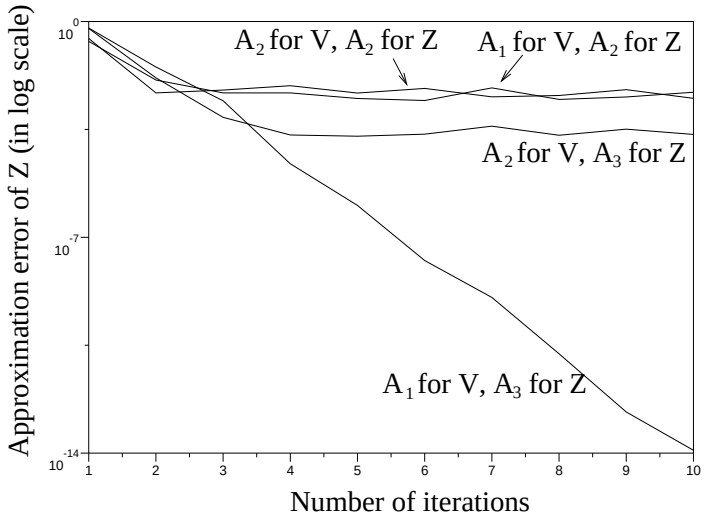


Figure 2: Approximation error of the gradient $Z = \partial_\alpha V$ using approximators $\mathcal{A}_1$ and $\mathcal{A}_2$ for the value function, and $\mathcal{A}_2$ and $\mathcal{A}_3$ for the gradient.

3.4 Variance Reduction with Only Z Representation

It would be desirable to design a similar variance reduction method using the gradient approximation only. However, as seen previously, the linear system (4) does not enable to recover r from the gradient (since $\partial_\alpha \mathcal{L}$ is not invertible), which prevents us from directly using the method of Section 2.

Nevertheless, from (13), we have the representation for Z:

$$Z(x) \quad = \quad \mathcal{A}Z_n(x) + \mathbb{E}\big[\Phi(r,X(x)) - \Psi(\mathcal{L}\mathcal{A}Z_n, X(x))\big],$$

from which we deduce the following algorithm: at stage n, simulate M trajectories $X^{n,m}$ per state (x_j) and update the approximation Z_n according to

$$Z_{n+1}(x_j) \quad = \quad \mathcal{A}Z_n(x_j) + \frac{1}{M}\sum_{m=1}^{M}\Big[\Phi(r,X^{n,m}(x_j)) - \Psi(\mathcal{L}\mathcal{A}Z_n, X^{n,m}(x_j))\Big].$$

Unfortunately, we may not expect this algorithm to exhibit a variance reduction to 0 in the case of perfect approximation of the gradient (i.e. $\mathcal{A}Z = Z$). Indeed, there is an incompressible variance term that comes from the estimation of $\Phi(r,X(x))$ instead of $\Psi(\mathcal{L}Z,X(x)) = \Psi(-\partial_\alpha \mathcal{L}\,\mathcal{L}^{-1}r, X(x))$.

To illustrate, in the infinite-horizon, discounted case (5), this incompressible variance term appears in the estimation of

$$\Phi(r,X(x)) - \Psi(\mathcal{L}Z,X(x)) \quad = \quad \sum_{t\geq 0}\gamma^t\Big[\frac{\partial_\alpha P(x_t,x_{t+1})}{P(x_t,x_{t+1})}\sum_{s\geq 0}\gamma^{s+1}r(x_{s+t+1}) - (I-\gamma P)Z(x_t)\Big].$$

However this variance (which can be related to the variance of the value function $V(x_{t+1})$ estimation by the sum of future rewards $\sum_{s\geq 0}\gamma^s r(x_{s+t+1})$ and a bound on the likelihood ratios $\frac{\partial_\alpha P(x_t,x_{t+1})}{P(x_t,x_{t+1})}$) is much lower (especially when γ is close to 1) than the initial variance of the direct estimation of $\mathbb{E}[\Phi(r,X(x))]$.

Thus, this algorithm would provide a geometric variance reduction, up to a threshold that depends on $\mathcal{V}_\Psi(Z-\mathcal{A}Z)$ plus this incompressible variance term (the proof is a simple extension of that of Theorem 2 taking into account the additional variance term). This algorithm may be interesting in Partially Observable MDPs, and provide an alternative technique compared to other variance reduction techniques developed in this setting (Greensmith et al., 2005).

4. Conclusion

We described a sequential control variates method for estimating the expectation of functionals in Markov chains, using linear approximation (in the values). We illustrate the method on value function and gradient estimates. We proved geometric variance reduction up to a threshold that depends on the approximation error of the functions of interest.

There are several possible directions for future research, among which:

- Estimate the number of sample trajectories M per state that enables the method to exhibit a geometric variance reduction (i.e. whenever $\rho_M < 1$).
- For a total budget of N trajectories per state, define what is the best trade-off between the number of iterations n and the number of trajectories M per iteration (such that $N = nM$).
- Define a stopping criterion (i.e. whenever there is no more variance decrease) from which we should continue (if needed) with a regular Monte Carlo method.
- Consider the case where the initial states are drawn according to some distribution over $\mathcal{X}$ instead of using the set of representative states $\mathcal{X}_J$.
- Consider non-linear function approximation.
- Extend this work to a model-free, on-line learning framework.

Appendix A. Proof of Theorem 2

From the decomposition

$$V - \mathcal{A}V_n = V - \mathcal{A}V + \sum_{i=1}^{J}(V-V_n)(x_i)\phi_i, \tag{15}$$

we have

$$\begin{aligned} V_{n+1}(x_j) &= \mathcal{A}V_n(x_j) + \frac{1}{M}\sum_{m=1}^{M}\Big[\Psi(\mathcal{L}(V-\mathcal{A}V), X^{n,m}(x_j)) \\ &\quad + \sum_{i=1}^{J}(V-V_n)(x_i)\Psi(\mathcal{L}\phi_i, X^{n,m}(x_j))\Big]. \end{aligned}$$

Thus

$$\begin{aligned}\mathrm{Var}^n V_{n+1}(x_j) &= \frac{1}{M}\mathrm{Var}^n\big[\Psi(\mathcal{L}(V-\mathcal{A}V),X(x_j)) \\ &\quad +\sum_{i=1}^{J}(V-V_n)(x_i)\Psi(\mathcal{L}\phi_i,X(x_j))\big].\end{aligned}$$

We use the general bound

$$\begin{aligned}\mathrm{Var}\,\big[\sum_i \alpha_i Y_i\big] &= \sum_{i_1,i_2}\alpha_{i_1}\alpha_{i_2}\mathrm{Cov}(Y_{i_1},Y_{i_2}) \\ &\le \sum_{i_1,i_2}|\alpha_{i_1}||\alpha_{i_2}|\sqrt{\mathrm{Var}\,[Y_{i_1}]}\sqrt{\mathrm{Var}\,[Y_{i_2}]} \le \big[\sum_i|\alpha_i|\sqrt{\mathrm{Var}\,[Y_i]}\big]^2, \qquad (16)\end{aligned}$$

for any real numbers $(\alpha_i)_i$ and square integrable real random variables $(Y_i)_i$, to deduce that

$$\mathrm{Var}^n V_{n+1}(x_j) \le \frac{1}{M}\Big[\sqrt{\mathcal{V}_\Psi(V-\mathcal{A}V)}+\sum_{i=1}^{J}|V-V_n|(x_i)\sqrt{\mathcal{V}_\Psi(\phi_i)}\Big]^2, \qquad (17)$$

with $\mathcal{V}_\Psi(f) := \sup_{1\le j\le J}\mathrm{Var}\,\Psi(\mathcal{L}f,X(x_j))$. Now, we use the variance decomposition

$$\begin{aligned}\mathrm{Var}\,V_{n+1}(x_j) &= \mathrm{Var}\,[\mathbb{E}^n[V_{n+1}(x_j)]]+\mathbb{E}[\mathrm{Var}^n[V_{n+1}(x_j)]] \\ &= \mathbb{E}[\mathrm{Var}^n[V_{n+1}(x_j)]],\end{aligned}$$

and the general bound (deduced similarly to (16))

$$\mathbb{E}\big[(\alpha_0+\sum_{i=1}^{J}\alpha_i Y_i)^2\big] \le 2\alpha_0^2+2\Big(\sum_{i=1}^{J}|\alpha_i|\sqrt{\mathbb{E}[Y_i^2]}\Big)^2, \qquad (18)$$

to deduce from (17) that

$$\nu_{n+1} \le \frac{2}{M}\big[\mathcal{V}_\Psi(V-\mathcal{A}V)+\big(\sum_{i=1}^{J}\sqrt{\mathcal{V}_\Psi(\phi_i)}\big)^2\nu_n\big],$$

which gives (10). Now, if M is such that $\rho_M := \frac{2}{M}\big(\sum_{i=1}^{J}\sqrt{\mathcal{V}_\Psi(\phi_i)}\big)^2 < 1$, then taking the upper limit finishes the proof of Theorem 2.

Appendix B. Proof of Theorem 4

Using (4) and (6), we have the decomposition

$$\begin{aligned}-\partial_\alpha\mathcal{L}\mathcal{A}V_n-\mathcal{L}\mathcal{A}Z_n &= -\partial_\alpha\mathcal{L}\mathcal{A}(V_n-V)-\partial_\alpha\mathcal{L}(\mathcal{A}V-V) \\ &\quad +\mathcal{L}(Z-\mathcal{A}Z)+\mathcal{L}\mathcal{A}(Z-Z_n) \\ &= \sum_{i=1}^{J}(V-V_n)(x_i)\partial_\alpha\mathcal{L}\phi_i-\partial_\alpha\mathcal{L}(\mathcal{A}V-V) \\ &\quad +\mathcal{L}(Z-\mathcal{A}Z)+\sum_{i=1}^{J}(Z-Z_n)(x_i)\mathcal{L}\phi_i.\end{aligned}$$

Now, using (15), the variance may be written

$$\begin{aligned}\mathrm{Var}^n Z_{n+1}(x_j) &= \frac{1}{M}\mathrm{Var}^n\Big[\Phi(\mathcal{L}(V-\mathcal{A}V),X(x_j))\\ &+\sum_{i=1}^{J}(V-V_n)(x_i)\Phi(\mathcal{L}\phi_i,X(x_j))-\Psi(\partial_\alpha\mathcal{L}(\mathcal{A}V-V),X(x_j))\\ &+\sum_{i=1}^{J}(V-V_n)(x_i)\Psi(\partial_\alpha\mathcal{L}\phi_i,X(x_j))+\Psi(\mathcal{L}(Z-\mathcal{A}Z),X(x_j))\\ &+\sum_{i=1}^{J}(Z-Z_n)(x_i)\Psi(\mathcal{L}\phi_i,X(x_j))\Big].\end{aligned}$$

We use (16) to deduce the bound

$$\begin{aligned}\mathrm{Var}^n Z_{n+1}(x_j) &\le \frac{1}{M}\Big[\sqrt{\mathcal{V}_\Phi(V-\mathcal{A}V)}+\sqrt{\mathcal{V}_\Psi(\mathcal{L}^{-1}\partial_\alpha\mathcal{L}(\mathcal{A}V-V))}\\ &+\sum_{i=1}^{J}|V-V_n|(x_i)\big(\sqrt{\mathcal{V}_\Phi(\phi_i)}+\sqrt{\mathcal{V}_\Psi(\mathcal{L}^{-1}\partial_\alpha\mathcal{L}\phi_i)}\big)\\ &+\sqrt{\mathcal{V}_\Psi(Z-\mathcal{A}Z)}+\sum_{i=1}^{J}|Z-Z_n|(x_i)\sqrt{\mathcal{V}_\Psi(\phi_i)}\Big]^2,\end{aligned}$$

Now, we use (18) to deduce that

$$\begin{aligned}z_{n+1} &\le \frac{2}{M}\Big\{\big(\sqrt{\mathcal{V}_\Phi(V-\mathcal{A}V)}+\sqrt{\mathcal{V}_\Psi(\mathcal{L}^{-1}\partial_\alpha\mathcal{L}(\mathcal{A}V-V))}\\ &+\Big[\sum_{i=1}^{J}\sqrt{\mathcal{V}_\Phi(\phi_i)}+\sqrt{\mathcal{V}_\Psi(\mathcal{L}^{-1}\partial_\alpha\mathcal{L}\phi_i)}\Big]^2 v_n\\ &+\sqrt{\mathcal{V}_\Psi(Z-\mathcal{A}Z)}+\Big[\sum_{i=1}^{J}\sqrt{\mathcal{V}_\Psi(\phi_i)}\Big]^2 z_n\Big\},\end{aligned}$$

and Theorem 4 follows.

References

C. G. Atkeson, S. A. Schaal, and Andrew W. Moore. Locally weighted learning. *AI Review*, 11, 1997.

J. Baxter and P. L. Bartlett. Infinite-horizon gradient-based policy search. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

P. W. Glynn. Likelihood ratio gradient estimation: an overview. In A. Thesen, H. Grant, and W. D. Kelton, editors, *Proceedings of the 1987 Winter Simulation Conference*, pages 366–375, 1987.

E. Gobet and S. Maire. Sequential control variates for functionals of Markov processes. *SIAM Journal on Numerical Analysis*, 43(3):1256–1275, 2005.

E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2005.

J. H. Halton. A retrospective and prospective survey of the Monte-Carlo method. *SIAM Review*, 12 (1):1–63, 1970.

J. H. Halton. Sequential Monte-Carlo techniques for the solution of linear systems. *Journal of Scientific Computing*, 9:213–257, 1994.

J. M. Hammersley and D. C. Handscomb. *Monte-Carlo Methods*. Chapman and Hall, 1964.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.

C. Kollman, K. Baggerly, D. Cox, and R. Picard. Adaptive importance sampling on discrete Markov chains. *The Annals of Applied Probability*, 9(2):391–412, 1999.

V. R. Konda and V. S. Borkar. Actor-critic-type learning algorithms for Markov decision processes. *SIAM Journal of Control and Optimization*, 38:1:94–123, 1999.

S. Maire. An iterative computation of approximations on Korobov-like spaces. *J. Comput. Appl. Math.*, 54(6):261–281, 2003.

P. Marbach and J. N. Tsitsiklis. Approximate gradient methods in policy-space optimization of Markov reward processes. *Journal of Discrete Event Dynamical Systems*, 13:111–148, 2003.

M. I. Reiman and A. Weiss. Sensitivity analysis via likelihood ratios. In J. Wilson, J. Henriksen, and S. Roberts, editors, *Proceedings of the 1986 Winter Simulation Conference*, pages 285–289, 1986.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems. MIT Press*, pages 1057–1063, 2000.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

V. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation and signal processing. *In Advances in Neural Information Processing Systems*, pages 281–287, 1997.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Journal of Machine Learning Research 7 (2006) 429–454 Submitted 7/05; Revised 1/06; Published 2/06

Inductive Synthesis of Functional Programs: An Explanation Based Generalization Approach

Emanuel Kitzelmann EMANUEL.KITZELMANN@WIAI.UNI-BAMBERG.DE
Ute Schmid UTE.SCHMID@WIAI.UNI-BAMBERG.DE
Department of Information Systems and Applied Computer Science
Otto-Friedrich-University
Bamberg, Germany

Editors: Roland Olsson and Leslie Pack Kaelbling

Abstract

We describe an approach to the inductive synthesis of recursive equations from input/output-examples which is based on the classical two-step approach to induction of functional Lisp programs of Summers (1977). In a first step, I/O-examples are rewritten to traces which explain the outputs given the respective inputs based on a datatype theory. These traces can be integrated into one conditional expression which represents a non-recursive program. In a second step, this initial program term is generalized into recursive equations by searching for syntactical regularities in the term. Our approach extends the classical work in several aspects. The most important extensions are that we are able to induce a *set* of recursive equations in one synthesizing step, the equations may contain more than one recursive call, and additionally needed parameters are automatically introduced.

Keywords: inductive program synthesis, inductive functional programming, explanation based generalization, recursive program schemes

1. Introduction

Automatic induction of recursive programs from input/output-examples (I/O-examples) is an active area of research since the sixties and of interest for AI research as well as for software engineering (Lowry and McCarthy, 1991; Flener and Partridge, 2001). In the seventies and eighties, there were several approaches to the synthesis of Lisp programs from examples or traces (see Biermann et al. 1984 for an overview). The most influential approach was developed by Summers (1977), who put inductive synthesis on a firm theoretical foundation.

Summers' early approach is an explanation based generalization (EBG) approach, thus it widely relies on algorithmic processes and only partially on search: In a first step, traces—steps of computations executed from a program to yield an output from a particular input—and predicates for distinguishing the inputs are calculated for each I/O-pair. Construction of traces, which are *terms* in the classical functional approaches, relies on knowledge of the inductive datatype of the inputs and outputs. That is, traces *explain* the outputs based on a theory of the used datatype given the respective inputs. The classical approaches for synthesizing Lisp-programs used the general Lisp datatype *S-expression*. By integrating traces and predicates into a conditional expression a non-recursive program explaining all I/O-examples is constructed as a result of the first synthesis step. In a second step, regular-

ities are searched for between the traces and predicates respectively. Found regularities are then inductively generalized and expressed in the form of the resulting recursive program.

The programs synthesized by Summers' system contain exactly one recursive function, possibly along with one constant term calling the recursive function. Furthermore, all synthesizable functions make use of a small fixed set of Lisp-primitives, particularly of exactly one predicate function, *atom*, which tests whether its argument is an atom, e.g., the empty list. The latter implies two things: First, that Summers' system is restricted to induce programs for *structural* problems on S-expressions. That means, that execution of induced programs depends only on the structure of the input S-expression, but never on the semantics of the atoms contained in it. For example, *reversing* a list is a structural problem, yet not *sorting* a list. The second implication is, that calculation of the traces is a deterministic and algorithmic process, i.e., does not rely on search and heuristics.

Due to only limited progress regarding the class of programs which could be inferred by functional synthesis, interest decreased in the mid-eighties. There was a renewed interest of inductive program synthesis in the field of inductive logic programming (ILP) (Flener and Yilmaz, 1999; Muggleton and De Raedt, 1994), in genetic programming and other forms of evolutionary computation (Olsson, 1995) which rely heavily on search.

We here present an EBG approach which is based on the methodologies proposed by Summers (1977). We regard the functional two-step approach as worthwhile for the following reasons: First, algebraic datatypes provide guidance in expressing the outputs in terms of the inputs as the first synthesis step. Second, it enables a seperate and thereby specialized handling of a knowledge dependent part and a purely syntactic driven part of program synthesis. Third, using both algebraic datatypes and seperating a knowledge-dependent from a syntactic driven part enables a more accurate use of search than in ILP or evolutionary programming. Fourth, the two-step approach using algebraic datatypes provides a systematic way to introduce auxiliary recursive equations if necessary.

Our approach extends Summers in several important aspects, such that we overcome fundamental restrictions of the classical approaches to induction of Lisp programs: First, we are able to induce a *set* of recursive equations in one synthesizing step, second, the equations may contain more than one recursive call, and third, additionally needed parameters are automatically introduced. Furthermore, our generalization step is domain-independent, in particular independent from a certain programming language. It takes as input a first-order term over an arbitrary signature and generalizes it to a recursive program scheme, that is, a set of recursive equations over that signature. Hence it can be used as a learning component in all domains which can represent their objects as recursive program schemes and provide a system for solving the first synthesis step. For example, we use the generalization algorithm for learning recursive control rules for AI planning problems (cp. Schmid and Wysotzki 2000; Wysotzki and Schmid 2001).

2. Overview Over the Approach

The three central objects dealt with by our system are (1) sets of *I/O-examples* specifying the algorithm to be induced, (2) *initial (program) terms* explaining the I/O-examples, and (3) *recursive program schemes (RPSs)* representing the induced algorithms. Their functional role in our two-step synthesis approach is shown in Figure 1.

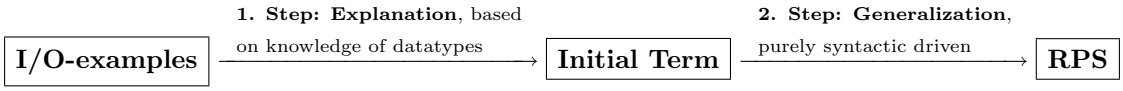


Figure 1: Two synthesis steps

2.1 First Synthesis Step: From I/O-examples to an Initial Term

An example for I/O-examples is given in Table 1. The examples specify the *lasts* function which takes a list of lists as input and yields a list of the last elements of the lists as output. In the first synthesis step, an initial term is constructed from these examples. An

$$\begin{aligned} [] &\mapsto [], \\ [[a]] &\mapsto [a], \\ [[a,b]] &\mapsto [b], \\ [[a,b,c],[d]] &\mapsto [c,d], \\ [[a,b,c,d],[e,f]] &\mapsto [d,f], \\ [[a],[b],[c]] &\mapsto [a,b,c] \end{aligned}$$

Table 1: I/O-examples for *lasts*

initial term is a term respecting an arbitrary first-order signature extended by the special constant symbol Ω, meaning the *undefined* value and directing generalization in the second synthesis step. Suitably interpreted, an initial term evaluates to the specified output when its variable is instantiated with a particular input of the example set and to *undefined* for all other inputs.

Table 2 gives an example of an initial term. It shows the result of applying the first synthesis step to the I/O-examples for the *lasts* function as shown in Table 1. *if* means the 3ary non-strict function which returns the value of its second parameter if its first parameter evaluates to *true* and otherwise returns the value of its third parameter; *empty* is a predicate which tests, whether its argument is the empty list; *head* and *tail* yield the first element and the rest of a list respectively; *cons* constructs a list from one element and a list; and [] denotes the empty list.

Calculation of initial terms relies on knowledge of the datatypes of the example inputs and outputs. For our exemplary *lasts* program inputs and outputs are lists. Lists are uniquely constructed by means of the empty list [] and the constructor *cons*. Furthermore, they are uniquely decomposed by the functions *head* and *tail*. That allows to calculate a unique term which expresses an example output in terms of the input. For example, consider the fourth I/O-example from Table 1: If x denotes the input $[[a,b,c],[d]]$, then the term $cons(head(tail(tail(head(x)))), head(tail(x)))$ expresses the specified output $[c,d]$ in terms of the input. Such traces are constructed for each I/O-pair. The overall concept for integrating the resulting traces into one initial term is to go through all traces in parallel position by position. If the same function symbol is contained at the current position in all traces, then it is introduced to the initial term at this position. If at least two traces differ at the current position, then an *if*-expression is introduced. Therefore a predicate function is calculated to discriminate the inputs according to the different traces. Construction of the initial term proceeds from the discriminated inputs and traces for the second and

```
if(empty(x), [],
 cons(
  head(
   if(empty(tail(head(x))), head(x),
    if(empty(tail(tail(head(x)))), tail(head(x)),
     if(empty(tail(tail(tail(head(x))))), tail(tail(head(x))),
      Ω)))),
  if(empty(tail(x)), [],
   cons(
    head(
     if(empty(tail(head(tail(x)))), head(tail(x)),
      Ω)),
    if(empty(tail(tail(x))), [],
     Ω)))))))
```

Table 2: Initial term for *lasts*

third branch of the *if*-tree respectively. We describe the calculation of initial terms from I/O-examples, i.e., the first synthesis step, in Section 4.

2.2 Second Synthesis Step: From Initial Terms to Recursive Equations

In the second synthesis step, initial ground terms are generalized to a recursive program scheme. Initial terms are considered as *(incomplete) unfoldings* of an RPS which is to be induced by generalization. An RPS is a set of recursive equations whose left-hand-sides consist of the names of the equations followed by their parameter lists and whose right-hand-sides consist of terms over the signature from the initial terms, the set of the equation names, and the parameters of the equations. One equation is distinguished to be the main one. An example is given in Table 3. This RPS, suitably interpreted, computes the *lasts* function as described above and specified by the examples in Table 1. It results from

$$lasts(x) = \text{if}(\text{empty}(x), [], \text{cons}(\text{head}(last(\text{head}(x))), lasts(\text{tail}(x))))$$

$$last(x) = \text{if}(\text{empty}(\text{tail}(x)), x, last(\text{tail}(x)))$$

Table 3: Recursive Program Scheme for *lasts*

applying the second synthesis step to the initial term shown in Table 2. Note that it is a *generalization* from the initial term in that it not merely computes the *lasts* function for the example inputs but for input-lists of arbitrary length containing lists of arbitrary length.

The second synthesis step does not depend on domain knowledge. The meaning of the function symbols is irrelevant, because the generalization is completely driven by detecting syntactical regularities in the initial terms. To understand the link between initial terms and RPSs induced from them, we consider the process of incrementally unfolding an RPS.

Unfolding of an RPS is a (non-deterministic and possibly infinite) rewriting process which starts with the instantiated head of the main equation of an RPS and which repeatedly rewrites a term by substituting any instantiated head of an equation in the term with either the equally instantiated body or with the special symbol Ω. Unfolding stops, when all heads of recursive equations in the term are rewritten to Ω, i.e., the term contains no rewritable head any more. Consider the *last* equation from the RPS shown in Table 3 and the initial instantiation $\{x \mapsto [a, b, c]\}$. We start with the instantiated head $last([a, b, c])$ and rewrite it to the term:

$$\text{if}(\text{empty}(\text{tail}([a, b, c])), [a, b, c], \text{last}(\text{tail}([a, b, c])))$$

This term contains the head of the *last* equation instantiated with $\{x \mapsto tail([a, b, c])\}$. When we rewrite this head again with the equally instantiated body we obtain:

$$\begin{array}{l}\text{if}(\text{empty}(\text{tail}([a, b, c])), [a, b, c],\\ \quad \text{if}(\text{empty}(\text{tail}(\text{tail}([a, b, c]))), \text{tail}([a, b, c]),\\ \quad\quad \text{last}(\text{tail}(\text{tail}([a, b, c])))\end{array}$$

This term now contains the head of the equation instantiated with $\{x \mapsto tail(tail([a, b, c]))\}$. We rewrite it once again with the instantiated body and then replace the head in the resulting term with Ω and obtain:

$$\begin{array}{l}\text{if}(\text{empty}(\text{tail}([a, b, c])), [a, b, c],\\ \quad \text{if}(\text{empty}(\text{tail}(\text{tail}([a, b, c]))), \text{tail}([a, b, c]),\\ \quad\quad \text{if}(\text{empty}(\text{tail}(\text{tail}(\text{tail}([a, b, c])))), \text{tail}(\text{tail}([a, b, c])), \Omega)))\end{array}$$

The resulting *finite* term of a *finite* unfolding process is also called *unfolding*. Unfoldings of RPSs contain regularities if the heads of the recursive equations are more than once rewritten with its bodies before they are rewritten with Ωs. The second synthesis step is based on detecting such regularities in the initial terms.

We describe the generalization of initial terms to RPSs in Section 3. The reason why we first describe the second synthesis step and only afterwards the first synthesis step is, that the latter is governed by the goal of constructing a term which can be generalized in the second step. Therefore, for understanding the first step, it is necessary to know the connection between initial terms and RPSs as established in the second step.

2.3 Characteristics and Limitations of the Approach

The overall objective of our approach is automatical induction of recursive functional programs from I/O-examples which are correct with respect to the functional behaviour desired by the user. Since the approach is based on finding differences between traces, i.e., analyzing one example in relation to the following example, the examples have to be the first k examples according to an ordering of the underlying data-type with the first example beeing the least complex instance for which the target recursive program is defined. This is in contrast to learning from a randomly chosen set of training data (according to some distribution) which is the common setting in most learning approaches, e.g., in all PAC-learning (Valiant, 1984) algorithms. Another implication of this generalization methodology is that very few

examples are sufficient. This is again in contrast to most learning settings, especially in contrast to the identification-in-the-limit setting (Gold, 1967). A third implication is that the examples have to be correct, i.e., the desired function has to be consistent with the examples. This is not a limitation in our view since we assume that the examples are given by some (end-user) programmer who knows the functional behaviour of the target program and thus can provide a few correct examples. A fourth implication of the example-driven approach is, that termination is assured for the induced programs. This is an important characteristic since in general, termination is not decidable. Our algorithms output a set of recursive equations which are consistent with the I/O-examples, i.e., which compute any specified example-output from the respective example-input.

There are restrictions regarding the programs which can be synthesized. The first step (see Section 2.1 for an overview and Section 4 for details) is restricted to structural problems, i.e., functions on lists may only depend on the list structure but not on the meaning of the items in the lists. The induced recursive equations stand in some call-relation. Due to the second synthesis step (see Section 2.2 for an overview and Section 3 for details), this relation is restricted to be *flat*, that is, recursive calls cannot be nested. Furthermore, the relation is non-mutual, i.e., if one equation calls a second one, then the second one cannot call the first one. Since the induction process relies on two successive synthesis steps, the overall restrictions are the sum of the restrictions of the first step and the restrictions of the second step. On the other hand, the two-step approach provides some modularity. Since the limitation to structural problems is a restriction of *only* the first step, it would be sufficient to only extend or substitute the first step to extend our approach to be capable of dealing with non-structural problems, e.g., sorting problems.

For experimental results, a discussion of the approach, and a comparison to other inductive programming systems see Sections 5 and 6.

3. Generalizing an Initial Term to an RPS

Since our generalization algorithm exploits the relation between an RPS and its unfoldings, in the following we will first introduce the basic terminology for terms, substitutions, and term rewriting as for example presented in Dershowitz and Jouanaud (1990). Then we will present definitions for RPSs and the relation between RPSs and their unfoldings. The set of all possible RPSs constitutes the hypothesis language for our induction algorithm. Some restrictions on this general hypothesis language are introduced and finally, the components of the generalization algorithm are described.

3.1 Preliminaries

We denote the set of natural numbers starting with 0 by $\mathbb{N}$ and the natural numbers greater than 0 by $\mathbb{N}_+$. A signature Σ is a set of (function) symbols with $\alpha : \Sigma \to \mathbb{N}$ giving the arity of a symbol. We write T_Σ for the set of ground terms, i.e., terms without variables, over Σ and $T_\Sigma(X)$ for the set of terms over Σ and a set of variables X. We write $T_{\Sigma,\Omega}$ for the set of ground terms—called partial ground terms—constructed over $\Sigma \cup \{\Omega\}$, where Ω is a special constant symbol denoting the *undefined* value. Furthermore, we write $T_{\Sigma,\Omega}(X)$ for the set of partial terms constructed over $\Sigma \cup \{\Omega\}$ and variables X. With $T^\infty_{\Sigma,\Omega}(X)$ we denote the set of inifinite partial terms over Σ and variables X. Over the sets $T_{\Sigma,\Omega}$, $T_{\Sigma,\Omega}(X)$ and $T^\infty_{\Sigma,\Omega}(X)$

a complete partial order (CPO) $\leq$ is defined by: a) $\Omega \leq t$ for all $t \in T_{\Sigma,\Omega}, T_{\Sigma,\Omega}(X), T^{\infty}_{\Sigma,\Omega}(X)$ and b) $f(t_1,\ldots,t_n) \leq f(t'_1,\ldots,t'_n)$ iff $t_i \leq t'_i$ for all $i \in [1;n]$.

Terms can uniquely be expressed as labeled trees: If a term is a constant symbol or a variable, then the corresponding tree consists of only one node labeled by the constant symbol or variable. If a term has the form $f(t_1,\ldots,t_n)$, then the root node of the corresponding tree is labeled with f and contains from left to right the subtrees corresponding to $t_1,\ldots,t_n$. We use the terms *tree* and *term* as synonyms. A *position* of a term/tree is a sequence of positive natural numbers, i.e., an element from $\mathbb{N}_+^*$. The set of positions of a term t, denoted $\mathbf{pos}(t)$, contains the empty sequence ϵ and the position iu, if the term has the form $t = f(t_1,\ldots,t_n)$ and u is a position from $\mathbf{pos}(t_i), i \in [1;n]$. Each position of a term uniquely denotes one subterm. We write $t|_u$ for denoting that subterm which is determined as follows: (a) $t|_\epsilon = t$, (b) if $t = f(t_1,\ldots,t_n)$ and u is a position in t_i, then $t|_{iu} = t_i|_u$, $i \in [1;n]$. E.g., for the term $f(x,y,g(h(a,p(s,t),b),z))$ the position 312 denotes the subterm $p(s,t)$ because it is the second subterm of the first subterm of the third subterm of the original term. We say that position u is smaller than position u', $u \leq u'$, if u is a prefix of u'. If u is a position of term t and $u' \leq u$, then u' is a position of t. For a term t and a position u, $\mathbf{node}(t,u)$ denotes the fixed symbol $f \in \Sigma$, if $t|_u = f(t_1,\ldots,t_n)$ or $t|_u = f$ respectively. The set of all positions at which a fixed symbol f appears in a term is denoted by $\mathbf{pos}(t,f)$. The replacement of a subterm $t|_u$ by a term s in a term t at position u is written as $t[u \leftarrow s]$. Let U denote a set of positions in a term t. Then $t[U \leftarrow s]$ denotes the replacement of all subterms $t|_u$ with $u \in U$ by s in t.

A *substitution* σ is a mapping from variables to terms. Substitutions are naturally continued to mappings from terms to terms by $\sigma(f(t_1,\ldots,t_n)) = f(\sigma(t_1),\ldots,\sigma(t_n))$. Substitutions are written in postfix notation, i.e., we write $t\sigma$ instead of $\sigma(t)$. Substitutions $\beta : X \to T_\Sigma$ from variables to *ground* terms are called (variable) *instantiations*. A term p is called *pattern* of a term t, iff $t = p\sigma$ for a substitution σ. A pattern p of a term t is called *trivial*, iff p is a variable and *non-trivial* otherwise. We write $t \leq_s p$ iff p is a pattern of t and $t <_s p$ iff additionally holds, that p and t can not be unified by variable renaming only.

A *term rewriting system (TRS)* over Σ and X is a set of pairs of terms $\mathcal{R} \subseteq T_\Sigma(X) \times T_\Sigma(X)$. The elements (l,r) of $\mathcal{R}$ are called rewrite rules and are written $l \to r$. A term t' can be derived in one rewrite step from a term t using $\mathcal{R}$ ($t \to_\mathcal{R} t'$), if there exists a position u in t, a rule $l \to r \in \mathcal{R}$, and a substitution $\sigma : X \to T_\Sigma(X)$, such that (a) $t|_u = l\sigma$ and (b) $t' = t[u \leftarrow r\sigma]$. $\mathcal{R}$ implies a rewrite relation $\to_\mathcal{R} \subseteq T_\Sigma(X) \times T_\Sigma(X)$ with $(t,t') \in \to_\mathcal{R}$ if $t \to_\mathcal{R} t'$.

3.2 Recursive Program Schemes

Definition 1 (Recursive Program Scheme) *Given a signature Σ, a set of* function variables $\Phi = \{G_1,\ldots,G_n\}$ *for a natural number $n > 0$ with $\Sigma \cap \Phi = \emptyset$ and arity $\alpha(G_i) > 0$ for all $i \in [1;n]$, a natural number $m \in [1;n]$, and a set of equations*

$$\mathcal{G} = \begin{array}{c} \{\ G_1(x_1,\ldots,x_{\alpha(G_1)}) = t_1, \\ \vdots \\ G_n(x_1,\ldots,x_{\alpha(G_n)}) = t_n\ \} \end{array}$$

where the t_i are terms with respect to the signature $\Sigma \cup \Phi$ and the variables $x_1, \ldots, x_{\alpha(G_i)}$, $\mathcal{S} = (\mathcal{G}, m)$ is an RPS. $G_m(x_1, \ldots, x_{\alpha(G_m)}) = t_m$ is called the main equation of $\mathcal{S}$.

The function variables in Φ are called *names* of the equations, the left-hand-sides are called *heads*, the right-hand-sides *bodies* of the equations. For the *lasts* RPS shown in Table 3 holds: $\Sigma = \{if, empty, cons, head, tail, []\}$, $\Phi = \{G_1, G_2\}$ with $G_1 = lasts$ and $G_2 = last$, and $m = 1$. $\mathcal{G}$ is the set of the two equations.

We can identify a TRS with an RPS $\mathcal{S} = (\mathcal{G}, m)$:

Definition 2 (TRS implied by an RPS) *Let $\mathcal{S} = (\mathcal{G}, m)$ be an RPS over Σ, Φ and X, and Ω the bottom symbol in $T_{\Sigma,\Omega}(X)$. The equations in $\mathcal{G}$ constitute rules $\mathcal{R}_\mathcal{S} = \{G_i(x_1, \ldots, x_{\alpha(G_i)}) \rightarrow t_i \mid i \in [1; n]\}$ of a term rewriting system. The system additionally contains rules $\mathcal{R}_\Omega = \{G_i(x_1, \ldots, x_{\alpha(G_i)}) \rightarrow \Omega \mid i \in [1; n],\ G_i$ is* recursive*$\}$.*

The standard interpretation of an RPS, called *free interpretation*, is defined as the supremum in $T^\infty_{\Sigma,\Omega}(X)$ of the set of all terms in $T_{\Sigma,\Omega}(X)$ which can be derived by the implied TRS from the head of the main equation. Two RPSs are called *equivalent*, iff they have the same free interpretation, i.e., if they compute the same function for every interpretation of the symbols in Σ. Terms in $T_{\Sigma,\Omega}$ which can be derived by the *instantiated* head of the main equation regarding some instantiation $\beta : X \rightarrow T_\Sigma$ are called *unfoldings* of an RPS relative to β. Note, that terms derived from RPSs are partial and do not contain function variables, i.e., all heads of the equations are eventually rewritten by Ωs.

The goal of the generalization step is to find an RPS which *explains* a set of initial terms, i.e., to find an RPS such that the initial terms are unfoldings of that RPS. We denote initial terms by $\bar{t}$ and a set of initial terms by $\mathcal{I}$. We liberalize $\mathcal{I}$ such that it may include *incomplete* unfoldings. Incomplete unfoldings are unfoldings, where some subtrees containing Ωs are replaced by Ωs.

We need to define four further concepts, namely *recursion positions* which are positions in the equation bodies where recursive calls appear, *substitution terms* which are the argument terms in recursive calls, *unfolding positions* which are positions in unfoldings at which the heads of the equations are rewritten with their bodies, and finally *parameter instantiations in unfoldings* which are subterms of unfoldings resulting from the initial parameter instantiation and the substitution terms:

Definition 3 (Recursion Positions and Substitution Terms) *Let $G(x_1, \ldots, x_{\alpha(G)}) = t$ with parameters $X = \{x_1, \ldots, x_{\alpha(G)}\}$ be a recursive equation. The set of* recursion positions *of G is given by $R = \mathbf{pos}(t, G)$. Each recursive call of G at position $r \in R$ in t implies substitutions $\sigma_r : X \rightarrow T_\Sigma(X) : x_j \mapsto t|_{rj}$ for all $j \in [1; \alpha(G)]$ for the parameters in X. We call the terms $t|_{rj}$* substitution terms *of G.*

For equation *lasts* of the *lasts* RPS (Table 3) holds $R = \{32\}$ and $x\,\sigma_{32} = tail(x)$. For equation *last* holds $R = \{3\}$ and $x\,\sigma_3 = tail(x)$.

Now consider an unfolding process of a recursive equation and the positions at which rewrite steps are applied in the intermediate terms. The first rewriting is applied at root-position ϵ, since we start with the instantiated head of the equation which is completely rewritten with the instantiated body. In the instantiated body, rewrites occur at recursion

positions R. Assume that on recursion position $r \in R$ the instance of the head is rewritten with an instance of the body. Then, relative to the resulting *subtree* at position r, rewrites occur again at recursion positions, e.g., at position $r' \in R$. Relative to the entire term these latter rewrites occur therefore at compositions of position r and recursion positions, e.g., at position rr' and so on. We call the infinite set of positions at which rewrites can occur in the intermediate terms within an unfolding of a recursive equation *unfolding positions*. They are determined by the recursion positions as follows:

Definition 4 (Unfolding Positions) *Let R be the recursion positions of a recursive equation G. The set of* unfolding positions *U of G is defined as the smallest set of positions which contain the position ϵ and, if $u \in U$ and $r \in R$, the position ur.*

The unfolding positions of equation *lasts* of the *lasts* RPS are $\{32, 3232, 323232, \ldots\}$.

Now we look at the variable instantiations occuring during unfolding a recursive equation. Recall the unfolding process of the *last* equation (see Table 3) described at the end of Section 2.2. The initial instantiation was $\beta_\epsilon = \beta = \{x \mapsto [a, b, c]\}$, thus in the body of the equation (replaced for the instantiated head as result of the first rewrite step), its variable is instantiated with this initial instantiation. Due to the substitution term $tail(x)$, the variable of the head in this body is instantiated with $\beta_3 = \sigma_3\,\beta_\epsilon = \{x \mapsto tail([a, b, c])\}$, i.e., the variable in the body replaced for this instantiated head is instantiated with $\sigma_3\,\beta_\epsilon$. A further rewriting step implies the instantiation $\beta_{33} = \sigma_3\,\sigma_3\,\beta_\epsilon = \sigma_3\,\beta_3 = \{x \mapsto tail(tail([a, b, c]))\}$ and so on. We index the instantiations occuring during unfolding with the unfolding positions at which the particular instantiated heads were placed. They are determined by the substitutions implied by recursive calls and an initial instantiation as follows:

Definition 5 (Instantiations in Unfoldings) *Let $G(x_1, \ldots, x_{\alpha(G)}) = t$ be a recursive equation with parameters $X = \{x_1, \ldots, x_{\alpha(G)}\}$, R and U the recursion positions and unfolding positions of G resp., σ_r the substitutions implied by the recursive call of G at position $r \in R$, and $\beta : X \to T_\Sigma$ an initial instantiation. Then a family of instantiations indexed over U is defined as $\beta_\epsilon = \beta$ and $\beta_{ur} = \sigma_r\,\beta_u$ for $u \in U, r \in R$.*

3.3 Restrictions and the Generalization Problem

An RPS which can be induced from initial terms is restricted in the following way: First, it contains no mutual recursive equations, second, there are no calls of recursive equations within calls of recursive equations (no nested recursive calls). The first restriction is not a semantical restriction, since each mutual recursive program can be transformed to an equivalent (regarding a particular algebra) non-mutual recursive program. Yet it is a *syntactical* restriction, since unfoldings of mutual RPSs can not be generalized using our approach. A restriction similar to the second one was stated by Rao (2004). He names TRSs complying with such a restriction *flat* TRSs.

Inferred RPSs conform to the following syntactical characteristics: First, all equations, potentially except of the main equation, are recursive. The main equation may be recursive as well, but, it is the only equation not required to be recursive. Second, inferred RPSs are minimal, in that (i) each equation is directly or indirectly (by means of other equations) called from the main equation, and (ii) no parameter of any equation can be omitted

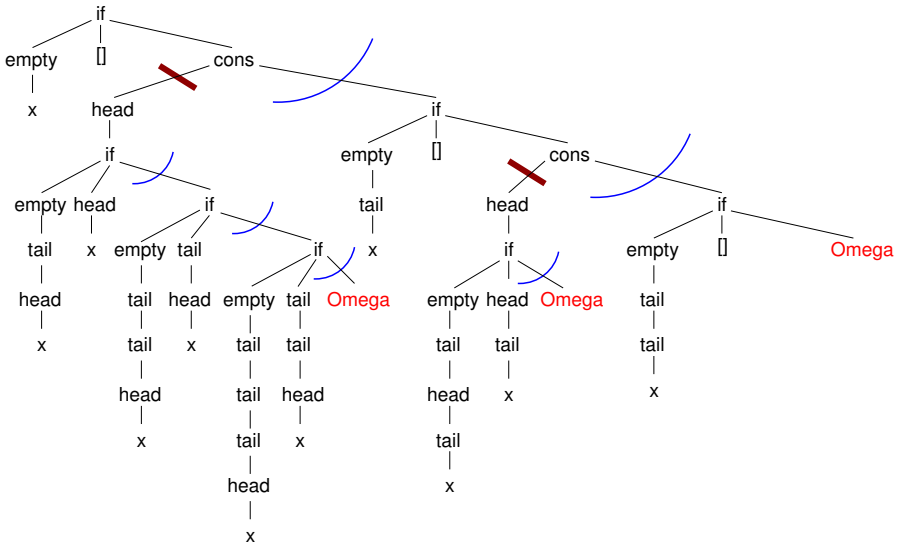


Figure 2: Initial Tree for *lasts*

without changing the free interpretation. RPSs complying with the stated restrictions and characteristics are called *minimal, non-mutual, flat recursive program schemes*.

There might be several RPSs which explain an initial term $\bar{t}$, but have different free interpretations. For example, Ω is an unfolding of *every* RPS with a recursive main equation. Therefore, an important question is which RPS will be induced. Summers (1977) required that recurrence relations hold at least over three succeeding traces and predicates to justify a generalization. A similar requirement would be that induced RPSs explain the initial terms *recurrently*, meaning that $\mathcal{I}$ contains at least one term $\bar{t}$ which can be derived from an unfolding process, in which each recursive equation had to be rewritten at least three times with its body. We use a slightly different requirement: One characteristic of minimal RPSs is, that if at least one substitution term is replaced by another, then the resulting RPS has a different free interpretation. We call this characteristic *substitution uniqueness*. Thus, it is sensible to require that induced RPSs are substitution unique regarding the initial terms, i.e., that *if* some substitution term is changed, then the resulting RPS *no longer* explains the initial terms. It holds, that a minimal RPS explains a set of initial trees recurrently, if it explains it substitution uniquely.

Thus the problem of generalizing a set of initial terms $\mathcal{I}$ to an RPS is to find an RPS which explains $\mathcal{I}$ and which is substitution unique regarding $\mathcal{I}$.

3.4 Solving the Generalization Problem

We will not state the generalization algorithm in detail in this section but we will describe the underlying concepts and the algorithm in a more informal manner. For this section and its subsections we use the term *body* of an equation for terms which are strictly speaking *incomplete* bodies: They contain only the name of the equation instead of complete recursive calls including substitution terms at recursion positions. For example, we refer to the term $\mathit{if}(\mathit{empty}(x), [], \mathit{cons}(\mathit{head}(\mathit{last}(\mathit{head}(x))), \mathit{lasts}))$ as the body for equation *lasts* of the *lasts*

RPS (see Table 3). The reason is, that we infer the complete body in two steps: First the term which we name body in this context, second the substitution terms for the recursive calls.

Generalization of a set of initial terms to an RPS is done in three successive steps, namely *segmentation of the terms*, *construction of equation bodies* and *calculation of substitution terms*. These three generalization steps are organized in a divide-and-conquer algorithm, where backtracking can occur to the divide-phase. *Segmentation* constitutes the divide-phase which proceeds top-down through the initial terms. Within this phase recursion positions (see Definition 3) and positions indicating further recursive equations are searched for each induced equation. The latter set of positions is called *subscheme positions* (see Definition 6 below). Found recursion positions imply unfolding positions (see Definition 4). As a result of the divide-phase the initial terms are divided into several parts by the subscheme positions, such that—roughly speaking—each particular part is assumed to be an unfolding of *one* recursive equation. Furthermore, the particular parts are segmented by the unfolding positions, such that—roughly speaking—each segment is assumed to be the result of *one* unfolding step of the respective recursive equation.

Consider the initial tree in Figure 2, it represents the initial term for *lasts*, shown in Table 2. The curved lines on the path to the rightmost Ω divide the tree into three segments which correspond to unfolding steps of the main equation, i.e., equation *lasts*. Note, that the rightmost segment is incomplete. The short broad lines denote two subtrees which are—except of their root *head*—unfoldings of the *last* equation. The curved lines within these subtrees divide each subtree into segments, such that each segment corresponds to one unfolding step of the *last* equation.

When the initial trees are segmented, calculation of equation bodies and of substitution terms follows within the conquer-phase. These two steps proceed bottom-up through the divided initial trees and reduce the trees during this process. The effect is, that bodies and substitution terms for each equation are calculated from trees which are unfoldings of *only* the currently induced equation and hence, each segment in these trees is an instantiation of the body of the currently induced equation. For example, for the *lasts* tree shown in Figure 2, a body and substitution terms are first calculated from the two subtrees, i.e., for the *last* equation. Since there are no further recursive equations called by the *last* equation—i.e., the segments of the two subtrees contain themselves no subtrees which are unfoldings of further equations—each segment is an instantiation of the body of the *last* equation. When this equation is completely inferred, the two subtrees are replaced by suitable instantiations of the head of the inferred *last* equation. The resulting reduced tree is an unfolding of merely *one* recursive equation, the *lasts* equation. The three segments in this reduced tree—indicated by the curved lines on the path to the rightmost Ω—are instantiations of the body of the searched for *lasts* equation. From this reduced tree, body and substitution terms for the *lasts* equation are induced and the RPS is completely induced.

Segmentations are searched for, whereas calculation of bodies and substitution terms are algorithmic. Construction of bodies always succeeds, whereas calculation of substitution terms—such that the inferred RPS explains the initial terms—may fail. Thus, an inferred RPS can be seen as the result of a search through a hypothesis space where the hypotheses are segmentations (divide-phase), and a *constructive* goal test, including construction of bodies and calculation of substitution terms (conquer-phase), which tests, whether the

completely inferred RPS explains the initial terms (and is substitution unique regarding them). In the following we describe each step in more detail:

3.4.1 Segmentation

When induction of an RPS from a set of initial trees $\mathcal{I}$ starts, the hypothesis is, that there exists an RPS with a recursive main equation which explains $\mathcal{I}$. First, recursion and subscheme positions for the hypothetical main equation G_m are searched for.

Definition 6 (Subscheme Positions) *Subscheme positions are all smallest positions in the body of a recursive equation G which denote subterms, in which calls of* further *recursive equations from the RPS appear, but* no recursive *call of equation G.*

E.g., the only subscheme position of equation *lasts* of the *lasts* RPS (Table 3) is $u = 31$. A priori, only particular positions from the initial trees come into question as recursion and subscheme positions, namely those which belong to a path leading from the root to an Ω. The reason is, that eventually *each* head of a recursive equation at any unfolding position in an intermediate term while unfolding this equation is rewritten with an Ω:

Lemma 7 (Recursion and Subscheme Positions imply Ωs) *Let $\bar{t} \in T_{\Sigma,\Omega}$ be an (incomplete) unfolding of an RPS $\mathcal{S} = (\mathcal{G}, m)$ with a recursive main equation G_m. Let R, U and S be the sets of recursion, unfolding and subscheme positions of G_m respectively. Then for all $u \in U \cap \mathbf{pos}(\bar{t})$ holds:*

1. $\mathbf{pos}(\bar{t}|_u, \Omega) \neq \emptyset$
2. $\forall s \in S$: *if* $us \in \mathbf{pos}(\bar{t})$ *then* $\mathbf{pos}(\bar{t}|_{us}, \Omega) \neq \emptyset$

It is not very difficult to see that this lemma holds. For a lack of space we do not give the proof here. It can be found in (Kitzelmann, 2003) where Lemma 7 and Lemma 9 are proven as one lemma. Knowing Lemma 7, before search starts, the initial trees can be reduced to their *skeletons* which are terms resulting from replacing subtrees without Ωs with variables.

Definition 8 (Skeleton) *The skeleton of a term $t \in T_{\Sigma,\Omega}(X)$, written* $\mathbf{skeleton}(t)$ *is the minimal pattern of t for which holds* $\mathbf{pos}(t, \Omega) = \mathbf{pos}(\mathbf{skeleton}(t), \Omega)$.

For example, consider the subtree indicated by the leftmost short broad line of the tree in Figure 2. Omitting the root *head*, it is an unfolding of the *last* equation of the *lasts* RPS shown in Table 3. Its skeleton is the *substantially* reduced term:

$$\mathit{if}(x_1, x_2, \mathit{if}(x_3, x_4, \mathit{if}(x_5, x_6, \Omega)))$$

Search for recursion and subscheme positions is done on the skeletons of the original initial trees. Thereby the hypothesis space is substantially narrowed without restricting the hypothesis language, since only those hypotheses are ruled out which are a priori known to fail the goal test.

Ωs are not only implied by recursion and subscheme positions, but also imply Ωs recursion and subscheme positions since Ωs in unfoldings result *only* from rewriting an instantiated head of a recursive equation in a term with an Ω:

Lemma 9 (Ωs imply recursion and subscheme positions) *Let $\bar{t} \in T_{\Sigma,\Omega}$ be an (incomplete) unfolding of an RPS $\mathcal{S} = (\mathcal{G}, m)$ with a recursive main equation G_m. Let R, U and S be the sets of recursion, unfolding and subscheme positions of G_m respectively. Then for all $v \in \mathbf{pos}(\bar{t}, \Omega)$ hold*

- *It exists an $u \in U \cap \mathbf{pos}(\bar{t}), r \in R$ with $u \leq v < ur$ or*
- *it exists an $u \in U \cap \mathbf{pos}(\bar{t}), s \in S$ with $us \leq v$.*

Proof: in (Kitzelmann, 2003).

From the definition of subscheme positions and the previous lemma follows, that subscheme positions are determined, if a set of recursion positions has been fixed. Lemma 7 restricts the set of positions which come into question as recursion and subscheme positions. Lemma 9 together with characteristics from subscheme positions suggests to organize the search as a search for recursion positions with a depending parallel calculation of subscheme positions. When hypothetical recursion, unfolding, and subscheme positions are determined they are checked regarding the labels in the initial trees on pathes leading to Ωs. The nodes between one unfolding position and its successors in unfoldings result from the same body (with different instantiations). Since variable instantiations only occur in subtrees at positions *not* belonging to pathes leading to Ωs, for each unfolding position the nodes between it and its successors *are necessarily equal*:

Lemma 10 (Valid Segmentation) *Let $\bar{t} \in T_{\Sigma,\Omega}$ be an unfolding of an RPS $\mathcal{S} = (\mathcal{G}, m)$ with a recursive main equation G_m. Then there exists a term $\check{t}_G \in T_{\Sigma,\Omega}(X)$ with $\mathbf{pos}(\check{t}_G, \Omega) = R \cup S$ such that for all $u \in U \cap \mathbf{pos}(\bar{t})$ hold: $\check{t}_G \leq_\Omega \bar{t}|_u$ where $\leq_\Omega$ is defined as (a) $\Omega \leq_\Omega t$ if $\mathbf{pos}(t, \Omega) \neq \emptyset$, (b) $x \leq_\Omega t$ if $x \in X$ and $\mathbf{pos}(t, \Omega) = \emptyset$, and (c) $f(t_1, \ldots, t_n) \leq_\Omega f(t'_1, \ldots, t'_n)$ if $t_i \leq_\Omega t'_i$ for all $i \in [1; n]$.*

Proof: in (Kitzelmann, 2003).

This lemma has to be slightly extended, if one allows for initial trees which are incomplete unfoldings. Lemma 10 states the requirements to assumed recursion and subscheme positions which can be assured at segmentation time. They are necessary for an RPS which explains the initial terms, yet not sufficient to assure, that an RPS complying with them exists which explains the initial trees. That is, later a backtrack can occur to search for other sets of recursion and subscheme positions. If found recursion and subscheme positions R and S comply with the stated requirements, we call the pair (R, S) a *valid segmentation*.

In our implemented system the search for recursion positions is organized as a greedy search through the space of sets of positions in the skeletons of the initial trees. When a valid segmentation has been found, compositions of unfolding and subscheme positions denote subtrees in the initial trees assumed to be unfoldings of further recursive equations. Segmentation proceeds recursively on each set of (sub)trees denoted by compositions of unfolding positions and one subscheme position $s \in S$. We denote such a set of initial (sub)trees $\mathcal{I}_s$.

3.4.2 Construction of Equation Bodies

Construction of each equation body starts with a set of initial trees $\mathcal{I}$ for which at segmentation time a valid segmentation (R, S) has been found, and an already inferred RPS for

each subscheme position $s \in S$ which explains the subtrees $\mathcal{I}_s$. These subtrees of the trees in $\mathcal{I}$ are replaced by the suitably instantiated heads or respectively bodies of the main equations of the already inferred RPSs. For example, consider the initial tree for *lasts* shown in Figure 2. When calculation of a body for the main equation *lasts* starts from this tree, an RPS containing only the *last* equation which explains all three subtrees indicated by the short broad lines has already been inferred. The initial tree is reduced by replacing these three subtrees by suitable instantiations of the head of the *last* equation. We denote the set of reduced initial trees also with $\mathcal{I}$ and its elements also with $\bar{t}$. By reducing the initial trees based on already inferred recursive equations, the problem of inducing a set of recursive equations is reduced to the problem of inducing merely *one* recursive equation (where the recursion positions are already known from segmentation).

An equation body is induced from the segments of an initial tree which is assumed to be an unfolding of one recursive equation.

Definition 11 (Segments) *Let $\bar{t}$ be an initial tree, R a set of (hypothetical) recursion positions and U the corresponding set of unfolding positions. The set of* complete segments *of $\bar{t}$ is defined as: $\{\bar{t}|_u[R \leftarrow G] \mid u \in U \cap \mathbf{pos}(\bar{t}), R \subset \bar{t}|_u\}$*

For example, consider the subtree indicated by the leftmost short broad line of the initial tree in Figure 2 without its root *head*. It is an unfolding of the *last* equation as stated in Table 3. When the only recursion position 3 has been found it can be splitted into three segments, indicated by the curved lines:

1. $\text{if}(\text{empty}(\text{tail}(\text{head}(x))), \text{head}(x), G)$
2. $\text{if}(\text{empty}(\text{tail}(\text{tail}(\text{head}(x)))), \text{tail}(\text{head}(x)), G)$
3. $\text{if}(\text{empty}(\text{tail}(\text{tail}(\text{tail}(\text{head}(x))))), \text{tail}(\text{tail}(\text{head}(x))), G)$

Expressed according to segments, the fact of a repetitive pattern between unfolding positions (see Lemma 10) becomes the fact, that the sequences of nodes between the root and each G are equal for each segment. Each segment is an instantiation of the body of the currently induced equation. In general, the body of an equation contains other nodes among those between its root and the recursive calls. These further nodes are also equal in each segment. Differences in segments of unfoldings of a recursive equation can *only* result from different instantiations of the variables of the body. Thus, for inducing the body of an equation from segments, we assume each position in the segments which is equally labeled in *all* segments as belonging to the body of the assumed equation, but each position which is variably labeled in at least two segments as belonging to the instantiation of a variable. This assumption can be seen as an inductive bias since it might occur, that also positions which are equal over all segments belong to a variable instantiation. Nevertheless it holds, that if an RPS exists which explains a set of initial trees, then there also exists an RPS which explains the initial trees and is constructed based on the stated assumption. Based on the stated assumption, the body of the equation to be induced is determined by the segments and defined as follows:

Definition 12 (Valid Body) *Given a set of reduced initial trees, the most specific maximal pattern of all segments of all the trees is called* valid body *and denoted $\hat{t}_G$.*

The maximal pattern of a set of terms can be calculated by first order anti-unification (Plotkin, 1969).

Calculating a valid body regarding the three segments enumerated above results in the term $\mathit{if}(\mathit{empty}(\mathit{tail}(x)), x, G)$. The *different* subterms of the segments are assumed to be instantiations of the parameters in the calculated valid body. Since each segment corresponds to one unique unfolding position, instantiations of parameters in unfoldings as defined in Definition 5 are now given. For example, from the three segments enumerated above we obtain:

1. $\beta_\epsilon(x) = \mathrm{head}(x)$
2. $\beta_3(x) = \mathrm{tail}(\mathrm{head}(x))$
3. $\beta_{33}(x) = \mathrm{tail}(\mathrm{tail}(\mathrm{head}(x)))$

3.4.3 Inducing Substitution Terms

Induction of substitution terms for a recursive equation starts on a set of reduced initial trees which are assumed to be unfoldings of one recursive equation, an already inferred (incomplete) equation body which contains only a G at recursion positions, and variable instantiations in unfoldings according to Definition 5. The goal is to complete each occurence of G to a recursive call including substitution terms for the parameters of the recursive equation.

The following lemma follows from Definition 5 and states characteristics of parameter instantiations in unfoldings more detailed. It characterizes the instantiations in unfoldings against the substitution terms of a recursive equation considering each single position in them.

Lemma 13 (Instantiations in Unfoldings) *Let $G(x_1, \ldots, x_{\alpha(G)}) = t$ be a recursive equation with parameters $X = \{x_1, \ldots, x_{\alpha(G)}\}$, recursion positions R and unfolding positions U, $\beta : X \to T_\Sigma$ an instantiation, σ_r substitution terms for each $r \in R$ and β_u instantiations as defined in Definition 5 for each $u \in U$. Then for all $i, j \in [1; \alpha(G)]$ and positions v hold:*

1. *If $(x_i\,\sigma_r)|_v = x_j$ then for all $u \in U$ hold $(x_i\beta_{ur})|_v = x_j\beta_u$.*
2. *If $(x_i\,\sigma_r)|_v = f((x_i\,\sigma_r)|_{v1}, \ldots, (x_i\,\sigma_r)|_{vn}), f \in \Sigma, \alpha(f) = n$ then for all $u \in U$ hold $\mathbf{node}(x_i\beta_{ur}, v) = f$.*

We can read the implications stated in the lemma in the inverted direction and thus we get almost immediately an algorithm to calculate the substitution terms of the searched for equation from the known instantiations in unfoldings.

One interesting case is the following: Suppose a recursive equation, in which at least one of its parameters *only* occurs within a recursive call in its body, for example the equation $G(x, y, z) = \mathit{if}(\mathit{zerop}(x), y, +(x, G(\mathit{prev}(x), z, \mathit{succ}(y))))$ in which this is the case for parameter z.[1] For such a variable no instantiations in unfoldings are given when induction of substitution terms starts. Also such variables are not contained in the (incomplete) valid

1. A practical example is the *tower-of-hanoi*-problem.

equation body. Our generalizer introduces them each time, when none of the both implications of Lemma 13 hold. Then it is assumed, that the currently induced substitution term contains such a "hidden" variable at the current position. Based on this assumption the instantiations in unfoldings of the hidden variable can be calculated and the inference of subtitution terms for it proceeds as described for the other parameters.

When substitution terms have been found, it has to be checked, whether they are substitution unique with regard to the reduced initial terms. This can be done for each substitution term that was found separately.

3.4.4 Inducing an RPS

We have to consider two further points: The first point is that segmentation presupposes the initial trees to be explainable by an RPS with a *recursive* main equation. Yet in Section 3.3 we characterized the inferable RPSs as liberal in this point, i.e., that also RPSs with a non-recursive main equation are inferable. In such a case, the initial trees contain a constant (not repetitive) part at the root such that no recursion positions can be found for these trees (as for example the three subtrees indicated by the short broad lines in Figure 2 which contain the constant root *head*). In this case, the root node of the trees is assumed to belong to the body of a non-recursive main equation and induction of RPSs recursively proceeds at each subtree of the root nodes.

The second point is that RPSs explaining the subtrees which are assumed to be unfoldings of further recursive equations at segmentation time are already inferred. Based on these already inferred RPSs, the initial trees are reduced and then a body and substitution terms are induced. Calculation of a body always succeeds, whereas calculation of substitution terms may fail. To deal with induction of RPSs explaining the subtrees as an independent problem requires, that *if* there exists a set of RPSs explaining the subtrees such that substitution terms can be calculated then substitution terms can be calculated for *any* set of RPSs explaining the subtrees.

Fortunately we could prove, that this requirement holds provided the main equation is constructed according to the "maximal-body" principle (see Definition 12). A proof sketch is as follows: Assume there are two different RPSs explaining the subtrees of a fixed subscheme position. Provided the main equations of the two RPSs are constructed according to the "maximal-body" principle, one can prove that the main equations of both RPSs have the same number of parameters with the same instantiations for explaining the subtrees (see Schmid, 2003, page 203, Theorem 7.3.3). Though the main equations of the RPSs might be different in their non-parameter positions, it is then assured that induction of the current equation will succeed for either both of the two different RPSs or for none of them but not for only one. The reason is that the possibly different non-parameter positions only affect the calculation of the body which always succeeds and that the critical point of inferring substitution terms is only affected by the parameters of the main equations of the RPSs and their instantiations.

4. Generating an Initial Term

Our theory and prototypical implementation for the first synthesis step uses the datatype *List*, defined as follows: The empty list [] is an (α-)list and if a is in element of type α and

l is an α-list, then $cons(a, l)$ is an α-list. Lists may contain lists, i.e., α may be of type *List* α'.

4.1 Characterization of the Approach

The constructed initial terms are composed from the list constructor functions $[], cons$, the functions for decomposing lists $head, tail$, the predicate *empty* testing for the empty list, one variable x, the 3ary (non-strict) conditional function *if* as control structure, and the bottom constant Ω meaning *undefined*. Similar to Summers (1977), the set of functions used in our term construction approach implies the restriction of induced programs to solve *structural* list programs. An extension to Summers is that we allow the example inputs to be *partially* ordered instead of only totally ordered. This is related to the extension of inducing *sets* of recursive equations as described in Section 3 instead of only one recursive equation.

We say that an initial term *explains* I/O-examples, if it evaluates to the specified output when applied to the respective input or to *undefined*. The goal of the first synthesis step is to construct an initial term which explains a set of I/O-examples and which can be explained by an RPS.

4.2 Basic Concepts

Definition 14 (Subexpressions) *The set of subexpressions of a list l is defined to be the smallest set which includes l itself and, if l has the form $cons(a, l')$, all subexpressions of a and of l'. If a is an atom, then a itself is its only subexpression.*

Since *head* and *tail*—which are defined by $head(cons(a, l)) = a$ and $tail(cons(a, l)) = l$—decompose lists uniquely, each subexpression can be associated with the unique term which computes the subexpression from the original list. E.g., consider the list $[[a], [b]]$. The set of all subexpressions together with their associated terms is: $\{x = [[a], [b]],\ head(x) = [a],\ tail(x) = [[b]],\ head(head(x)) = a,\ tail(head(x)) = [],\ head(tail(x)) = [b],\ tail(tail(x)) = [],\ head(head(tail(x))) = b,\ tail(head(tail(x))) = []\}$.

Since lists are uniquely constructed by the constructor functions $[]$ and *cons*, traces which compute the specified output can uniquely be constructed from the terms for the subexpressions of the respective input:

Definition 15 (Construction of Traces) *Let $i \mapsto o$ be an I/O-pair (i is a list). If o is a subexpression of i, then the trace is defined to be the term associated with o. Otherwise o has the form $cons(a, l)$. Let t and t' be the traces for the I/O-pairs $i \mapsto a$ and $i \mapsto l$ respectively. The trace for $i \mapsto o$ is defined to be the term $cons(t, t')$.*

For example, the trace for computing (the example-output) $[a, b]$ from (the example-input) $[[a], [b]]$ is the term $cons(head(head(x)), head(tail(x)))$.

Similar to Summers, we discriminate the inputs with respect to their structure, more precisely with regard to a partial order over them implied by their structural complexity. As stated above, we allow for arbitrarily nested lists as inputs. A partial order over such lists is given by: $[] \leq l$ for all lists l and $cons(a, l) \leq cons(a', l')$, iff $l \leq l'$ and, if a and a' are again lists, $a \leq a'$.

Consider any unfolding of an RPS. Generally it holds, that greater positions on a path leading to an Ω result from more rewritings of a head of a recursive equation with its body compared to some smaller position. In other words, the computation represented by a node at a greater position is one on a deeper recursion level than a computation represented by a smaller position. Since we use only the complexity of an input list as criterion whether the recursion stops or whether another call appears with the input decomposed in some way, deeper recursions result from more complex inputs in the induced programs.

4.3 Solving the Term Construction Problem

The overall concept of constructing the initial tree is to introduce the nodes from the traces position by position to the initial tree *as long as the traces are equal* and to introduce an *if*-expression as soon as at least two (sub)traces differ. The predicate in the *if*-expression divides the inputs into two sets. The "then"-subtree is recursively constructed from the input/trace-pairs whose inputs evaluate to *true* with the predicate and the "else"-subtree is recursively constructed from the other input/trace-pairs. Eventually only one single input/trace-pair remains when an *if*-expression is introduced. In this case an Ω indicating a recursive call on this path is introduced as leaf at the current position in the initial term and (this subtree of) the initial tree is finished. The reason for introducing an Ω in this case is, that we assume, that *if* the input/trace-set would contain a pair with a more complex input, than the respective trace would at some position differ from the remaining trace and thus it would imply an *if*-expression, i.e., a recursive call at some deeper position. Since we do not know the position at which this difference would occur, we can not use this single trace, but have to indicate a recursive call on this path by an Ω. Thus, for principal reasons, the constructed initial terms are undefined for the most complex inputs of the example set. There are two consequences of this particular loss of information in the initial terms compared to the I/O-examples. Since the following generalization step is based on the initial terms (1) the neccessary number of examples increases and (2) *if* the generalized program is incorrect it could especially be incorrect for the most complex examples. Thus consistence of the induced programs with respect to the I/O-examples is generally only assured for all examples except of the most complex ones.

We now consider the both cases that all roots of the traces are equal and that they differ respectively more detailed.

4.3.1 Equal Roots

Suppose all generated traces have the same root symbol. In this case, this symbol constitutes the root of the initial tree. Subsequently the sub(initial)trees are calculated through a recursive call to the algorithm. Suppose the initial tree has to explain the I/O-examples $\{[a] \mapsto a,\ [a,b] \mapsto b,\ [a,b,c] \mapsto c\}$. Calculating the traces and replacing them for the outputs yields the input/trace-set $\{[a] \mapsto head(x),\ [a,b] \mapsto head(tail(x)),\ [a,b,c] \mapsto head(tail(tail(x)))\}$. All three traces have the same root *head*, thus we construct the root of the initial tree with this symbol. The algorithm for constructing the initial subterm of the constructed root *head* now starts recusively on the set of input/trace-pairs where the traces are the subterms of the roots *head* from the three original traces, i.e., on the set $\{[a] \mapsto x,\ [a,b] \mapsto tail(x),\ [a,b,c] \mapsto tail(tail(x))\}$.

The traces from these new input/trace-set have different roots, that is, an *if*-expression is introduced as subtree of the constructed initial tree.

4.3.2 Introducing Control Structure

Suppose the traces (at least two of them) have different roots, as for example the traces of the second input/trace-set in the previous subsection. That means that the initial term has to apply different computations to the inputs corresponding to the different traces. This is done by introducing the conditional function *if*, i.e., the root of the initial term becomes the function symbol *if* and contains from left to right three subtrees: First, a predicate term with the predicate *empty* as root to distinguish between the inputs which have to be computed differently with regard to their complexity; second, a tree explaining all I/O-pairs whose inputs are evaluated to *true* from the predicate term; third, a tree explaining the remaining I/O-examples. It is presupposed, that all traces corresponding to inputs evaluating to *true* with the predicate are equal. These equal subtraces become the second subtree of the *if*-expression, i.e., they are evaluated, if an input evaluates to *true* with the predicate. That means that never an Ω occurs in a "then"-subtree of a constructed initial tree, i.e., that recursive calls in the induced RPSs may only occur in the "else"-subtrees. For the "else"-subtree the algorithm is recursively processed on all remaining input/trace-pairs.

For the predicate must hold that it evaluates to *true* for the least complex inputs because the "then"-subtree represents the termination of recursion whereas the "else"-subtree represents a further recursive call (for more complex inputs) of the induced program. An algorithm for calculating predicates evaluating to *true* for a particular expression and to *false* for any more complex expression can be found in (Smith, 1984, page 310). If, for example, the two input lists $[a, b]$ and $[a, b, c]$ shall be distinguished then the predicate is $empty(tail(tail(x)))$. For more complex data types as for example trees, or for nested lists, calculation of predicates might not be unique. Then a strategy for chosing a predicate has to be applied.

5. Experimental Results

We have implemented prototypes (without any thoughts about efficiency) for both described steps, construction of the initial tree and generalization to an RPS. The implementations are in Common-Lisp. In Table 4 we have listed experimental results for a few sample problems. Due to the restrictions of the first synthesis step all these induced programs deal with structural problems and are composed of only the primitive functions stated in Section 4.1. Many interesting programs, as for example *quicksort* or *towers-of-hanoi*, do not meet these restrictions and are not regarded. Due to the restriction of the second synthesis step all these programs contain no nested recursive calls. The first column lists the names for the induced functions, the second column lists the number of given I/O-pairs, the third column lists the total number of induced equations and in parentheses the number of induced *recursive* equations, and the fourth column lists the times consumed by the first step, the second step, and the total time respectively. The experiments were performed on a Pentium 4 with Linux and the program runs are interpreted with the *clisp* interpreter.

All induced programs compute the intended function. The number of given examples is in each case the minimal one. When given one example less, the system does *not* produce

function	#expl	#eqs(#rec)	times in sec
last	4	2(1)	.003 / .001 / .004
unpack	4	1(1)	.003 / .002 / .005
init	4	1(1)	.004 / .002 / .006
evenpos	7	2(1)	.01 / .004 / .014
switch	6	1(1)	.012 / .004 / .016
lasts	6	2(2)	.014 / .015 / .029
shift	6	3(2)	.015 / .033 / .048
mult-lasts	6	3(3)	.023 / .21 / .233
reverse	6	4(3)	.031 / .422 / .453
multi	12	5(5)	.114 / 6.96 / 7.074

Table 4: Some inferred functions

an *unintended* program, but produces *no* program. Indeed, an initial term is produced in such a case which is consistent with the example set, but no RPS is generalized, because it exists no RPS which explains the initial term *and is substitution unique with regard to it* (see Section 3.3).

last computes the last element of a list. The main equation is not recursive and only applies a *head* to the result of the induced recursive equation which computes a one element list containing the last element of the input list. *unpack* produces an output list, in which each element from the input list is encapsulated in a one element list, e.g., $unpack([a, b, c]) = [[a], [b], [c]]$. *unpack* is the classical example in (Summers, 1977). *init* returns the input list without the last element. *evenpos* computes a list containing each second element of the input list. The main equation is not recursive and only deals with the empty input list as special case. *switch* returns a list, in which each two successive elements of the input list are on switched positions, e.g., $switch([a, b, c, d, e]) = [b, a, d, c, e]$. *lasts* is the program described in Section 2. The given I/O-examples are those from Table 1. *shift* moves the last element of the input list to the front of the list. The main equation is not recursive and only deals with the empty list and a one-element-list as special cases. The two induced recursive equations compute the last element and the *init* of the input list respectively and are combined to compute the *shift* function. *mult-lasts* takes a list of lists as input just like *lasts*. It returns a list of the same structure as the input list where each inner list contains repeatedly the last element of the corresponding inner list from the input. For example, $mult\text{-}lasts([[a, b], [c, d, e], [f]]) = [[b, b], [e, e, e], [f]]$. All three induced equations are recursive. The third equation computes a one element list containing the last element of an input list. The second equation calls the third equation and returns a list of the same structure as a given input list where the elements of the input list are replaced by the last element. The first equation calls the second equation to compute the inner lists. *reverse* reverses a list. The induced program has an unusual form, nevertheless it is correct. Finally *multi* is a combination of *mult-lasts*, *unpack*, and *switch*. It takes a list of lists as input and applies *mult-lasts* to the first list, *unpack* to the second list, *switch* to the third list, and then again *mult-lasts* to the fourth list, *unpack* to the fifth list, *switch* to the sixth list and so on. *multi* is in one run induced from the examples shown in Table 5. The induced program is

$$
\begin{array}{rcl}
[] & \mapsto & [], \\
[[a]] & \mapsto & [[a]], \\
[[a,b]] & \mapsto & [[b,b]], \\
[[a,b,c],[d]] & \mapsto & [[c,c,c],[[d]]], \\
[[a,b,c,d],[e,f],[g]] & \mapsto & [[d,d,d,d],[[e],[f]],[g]], \\
[[a,b,c,d,e],[f,g,h],[i,j]] & \mapsto & [[e,e,e,e,e],[[f],[g],[h]],[j,i]], \\
[[a],[b,c,d,e],[f,g,h],[i]] & \mapsto & [[a],[[b],[c],[d],[e]],[g,f,h],[i]], \\
[[a],[b],[c,d,e,f],[g,h]] & \mapsto & [[a],[[b]],[d,c,f,e],[h]], \\
[[a],[b],[c,d,e,f,g],[h],[i]] & \mapsto & [[a],[[b]],[d,c,f,e,g],[h],[[i]]], \\
[[a],[b],[c,d,e,f,g,h],[i],[j,k],[l]] & \mapsto & [[a],[[b]],[d,c,f,e,h,g],[i],[[j],[k]],[l]], \\
[[a],[b],[c],[d],[e],[f,g]] & \mapsto & [[a],[[b]],[c],[d],[[e]],[g,f]], \\
[[a],[b],[c],[d],[e],[f],[g]] & \mapsto & [[a],[[b]],[c],[d],[[e]],[f],[g]]
\end{array}
$$

Table 5: I/O-examples for *multi*

shown in Table 6. Note that the names *multi, switch, unpack* etc. of the equations of the induced RPS are ex post introduced from us; the system introduces names $G_1, G_2, \ldots$.

$$
\begin{array}{rl}
multi(x) = & \mathrm{if}(\mathrm{empty}(x), [], \mathrm{cons}(multlasts(\mathrm{head}(x)), \\
& \quad \mathrm{if}(\mathrm{empty}(\mathrm{tail}(x)), [], \mathrm{cons}(unpack(\mathrm{head}(\mathrm{tail}(x))), \\
& \quad\quad \mathrm{if}(\mathrm{empty}(\mathrm{tail}(\mathrm{tail}(x))), [], \mathrm{cons}(switch(\mathrm{head}(\mathrm{tail}(\mathrm{tail}(x)))), \\
& \quad\quad\quad multi(\mathrm{tail}(\mathrm{tail}(\mathrm{tail}(x))))))))))
\end{array}
$$

$$
\begin{array}{rl}
switch(x) = & \mathrm{if}(\mathrm{empty}(\mathrm{tail}(x)), x, \mathrm{cons}(\mathrm{head}(\mathrm{tail}(x)), \mathrm{cons}(\mathrm{head}(x), \\
& \quad \mathrm{if}(\mathrm{empty}(\mathrm{tail}(\mathrm{tail}(x))), [], switch(\mathrm{tail}(\mathrm{tail}(x)))))))
\end{array}
$$

$$
unpack(x) = \mathrm{cons}(\mathrm{cons}(\mathrm{head}(x), []), \mathrm{if}(\mathrm{empty}(\mathrm{tail}(x)), [], unpack(\mathrm{tail}(x))))
$$

$$
multlasts(x) = \mathrm{if}(\mathrm{empty}(\mathrm{tail}(x)), x, \mathrm{cons}(\mathrm{head}(last(x)), multlasts(\mathrm{tail}(x))))
$$

$$
last(x) = \mathrm{if}(\mathrm{empty}(\mathrm{tail}(\mathrm{tail}(x))), \mathrm{tail}(x), last(\mathrm{tail}(x)))
$$

Table 6: Recursive Program Scheme for *multi*

Considering the times taken by the first and second synthesis step for the problems listed in Table 4 one finds (1) that they depend on the number of examples for the first step and on the number of recursive equations for the second step and (2) that the times taken from the second step increase faster than the times taken from the first step. A detailed analysis of the complexities of the two synthesis steps has still to be done. For some results regarding the second step see (Schmid, 2003, Section 7.4.1).

6. Comparison with Other Inductive Programming Systems

Inductive learning of programs is in general primarily known from the field of inductive logic programming (ILP), where the target language is relational descriptions in form of logic programs, e.g., Prolog programs. However the usual goal of ILP is learning *concepts* in form of a single, non-recursive predicate but not learning *recursive algorithms* with multiple interdependent predicates. Nevertheless there are ILP systems that have reasonable behaviour on inducing recursive logic programs, GOLEM (Muggleton and Feng, 1990) as an example. One interactive ILP system *specializing* in synthesizing recursive programs is DIALOGS (Flener, 1997). For a comparison of different ILP systems specializing in learning recursive predicates see (Flener and Yilmaz, 1999). More recent approaches to learn recursive logic programs are the approach of Rao and Sattar (2001) and the system ATRE (Malerba, 2003; Berardi et al., 2004). Two non-ILP systems for inducing recursive programs are the evolutionary computation system ADATE (Automatic Design of Algorithms Through Evolution) (Olsson, 1995) which induces functional programs in Standard ML and the Optimal Ordered Problem Solver (OOPS) (Schmidhuber, 2004). All these systems and approaches differ in their induction strategy, in the training data (many examples vs. few examples, only positive vs. both positive and negative examples, I/O-examples vs. example-inputs together with an evaluation function), in whether the induction relies on background knowledge, and in the limitations regarding inducable programs.

Our approach is different from most of the other approaches in that it is mostly analytical instead of search-based. In the following, we discuss this difference considering our system, ADATE, the well known ILP system FOIL (Quinlan, 1990) which was extended with concepts to learn recursive clauses (Cameron-Jones and Quinlan, 1993), and the Optimal Ordered Problem Solver. FOIL as well as ADATE and our system are capable of inducing more than one recursive function/clause in one run. FOIL needs a specification for *every* clause it shall induce, whereas ADATE and our system are capable of automatically introduce auxiliary recursive functions and thereby auxiliary parameters. E.g., one can give a specification of reversing a list to our system in terms of the I/O-examples $\{[] \mapsto [],\ [a] \mapsto [a],\ [a,b] \mapsto [b,a],\ [a,b,c] \mapsto [c,b,a],\ [a,b,c,d] \mapsto [d,c,b,a]\}$ and it automatically introduces an auxiliary function containing the second accumulating variable. When ADATE or our system outputs more than one recursive function these functions clearly are interdependent. In contrast, when different predicates to learn in one run are specified in FOIL, they are mostly learned independently one after another though foremost learned predicates can be used as background knowledge for the remaining predicates. FOIL has no knowledge of structured datatypes, e.g. lists, on its own and actually can handle only atoms. Thus lists have to be simulated with constants and one has to specify procedures for "composing" and "decomposing" such simulated lists as background knowldedge.

FOIL and ADATE directly search through a hypothesis space, whereas our system deterministically constructs an explanation of the I/O-examples in a first step and only then searches a hypothesis space for a generalization of the explanation. The main effect regarding this difference is that FOIL and ADATE can be given any background knowledge in terms of additional predicate specifications in the case of FOIL and predefined SML functions in the case of ADATE respectively. These predicates or functions respectively are then used in the synthesized programs. Since the branching factor in the search spaces

grows as this background knowledge increases, increasing background knowledge supposably tends to result in increasing run times. In contrast—though our generalization component is domain independent—, our system on the whole is restricted to background knowledge that admits an almost deterministic explanation of the I/O-examples. Therefore it cannot be given any predefined functions to be used in a synthesized program. Until now, synthesized programs can only be composed of the predefined functions stated in Section 4.1. Since the particular knowledge of datatypes admits deterministic explanations, it is used to *restrict* the hypothesis search space.

It would be interesting to compare the run times of FOIL, ADATE, and our system. However, since the systems have different restrictions, it is not trivial to find adequate and significant problems and specifications for a comparison. The restrictions of FOIL—no handling with structured datatypes and no automatic introduction of auxiliary predicates and variables—could be dealt with by simulating lists and by specifying all needed predicates. On the other side, the restrictions of our system—only particular primitive functions can be used in the synthesized programs—cannot be bypassed at present. For problems which need only few predicates/functions as background knowledge and contain only one recursive predicate/function as for example *last* or *member*, FOIL as well as our system take less than one second on a Pentium 4 with Linux. We have not measured the run times of the ADATE system for these simple problems, but on the web pages of the ADATE system[2] Roland Olsson reports on 570 seconds on a 200MHz PentiumPro for reversing a list.

Like FOIL and ADATE, the Optimal Ordered Problem Solver is based on a "generate-and-test" method. In (Schmidhuber, 2004), inducing a recursive program for *towers-of-hanoi* is reported. The induction takes a few days on a personal computer.

It is theoretically plausible as well as empirically evident that higher generality of inducable programs leads to higher computational effort of the program synthesizer. ADATE as well as OOPS are highly general program synthesizers with high run times. On the other extreme is our system with strong restrictions regarding synthesizable programs but much faster program inductions. An interesting question is, whether it could be possible to combine both approaches. We think that one approach to combine both methods could be to generate the traces and predicates with some "generate-and-test" method and then generalizing the integrated initial terms with our generalization algorithm. This would overcome the restrictions to structural problems as well as to restricted background knowledge which are implied by our analytical trace and predicate construction method. On the other hand, by keeping the construction and generalization of traces one would (1) presumably hold some advantage regarding run time compared to pure "generate-and-test" algorithms because searching for traces is less elaborate than searching for a recursive program and (2) hold the important point of constructing programs which are assured to terminate. Thus this could be a good compromise regarding the conflicting aspects of generality of the induced programs, computational effort of the induction algorithm, and assurance of termination for the induced programs.

2. http://www-ia.hiof.no/~rolando/

7. Conclusion and Further Research

We presented an EBG approach to inducing sets of recursive equations representing functional programs from I/O-examples. The underlying methodologies are inspired by classical approaches to induction of functional Lisp-programs, particularly by the approach of Summers (1977). The presented approach goes in three main aspects beyond Summers' approach: *Sets* of recursive equations can be induced at once instead of only one recursive equation, each equation may contain more than one recursive call, and additionally needed parameters are introduced systematically. We have implemented prototypes for both steps. The generalizer works domain-independent and all problems which comply to our general program scheme (Definition 1) with the restrictions described in Section 3.3 can be solved, whereas construction of initial terms as described in Section 4 relies on knowledge of datatypes.

We are investigating several extensions for the first synthesis step: First, we try to integrate knowledge about further datatypes such as trees and natural numbers. For example, we believe, that if we introduce *zero* and *succ*, denoting the natural number 0 and the successor function resp. as constructors for natural numbers, *prev* for "decomposing" natural numbers and the predicate *zerop* as bottom test on natural numbers, then it should be possible to induce a program returning the length of a list for example. Another extension will be to allow for more than one input parameter in the I/O-examples, such that *append* becomes inducable for example. A third extension should be the ability to use user-defined or in a previous step induced functions within an induction step.

Until now our approach suffers from the restriction to structural problems due to the principal approach to calculate traces deterministically without search in the first synthesis step. We work on overcoming this restriction, i.e., on extending the first synthesis step to the ability of dealing with problems which are not (only) structural, list sorting for example. A strong extension to the second step would be the ability to deal with nested recursive calls, yet this would imply a much more complex structural analysis on the initial terms.

Acknowledgments

We would like to acknowledge previous work from Martin Mühlpfordt and Fritz Wysotzki. Martin Mühlpfordt implemented the second synthesis step. We also like to thank three anonymous reviewers for their very helpful comments and suggestions for improving an earlier draft of this paper.

References

M. Berardi, A. Varlaro, and D. Malerba. On the effect of caching in recursive theory learning. In R. Camacho, R. D. King, and A. Srinivasan, editors, *Inductive Logic Programming: ILP 2004*, pages 44–62. Springer, 2004.

A. W. Biermann, G. Guiho, and Y. Kodratoff, editors. *Automatic Program Construction Techniques*. Collier Macmillan, 1984.

R. Mike Cameron-Jones and J. Ross Quinlan. Avoiding pitfalls when learning recursive theories. In *IJCAI*, pages 1050–1055. Morgan Kaufmann, 1993.

N. Dershowitz and J.-P. Jouanaud. Rewrite systems. In J. Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B. Elsevier, 1990.

P. Flener. Inductive logic program synthesis with DIALOGS. In S. Muggleton, editor, *Proceedings of ILP'96*, pages 175–198. Springer, 1997.

P. Flener and D. Partridge. Inductive programming. *Autom. Softw. Eng.*, 8(2):131–137, 2001.

P. Flener and S. Yilmaz. Inductive synthesis of recursive logic programs: Achievements and prospects. *Journal of Logic Programming*, 41(2–3):141–195, 1999.

E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

E. Kitzelmann. Inductive functional program synthesis – a term-construction and folding approach. Master's thesis, Dept. of Computer Science, TU Berlin, 2003. http://www.cogsys.wiai.uni-bamberg.de/kitzelmann/documents/thesis.ps.

M. L. Lowry and R. D. McCarthy. *Autmatic Software Design*. MIT Press, Cambridge, Mass., 1991.

D. Malerba. Learning recursive theories in the normal ILP setting. *Fundamenta Informaticae*, 57(1):39–77, 2003.

S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming, Special Issue on 10 Years of Logic Programming*, 19-20:629–679, 1994.

S. H. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the First Conference on Algorithmic Learning Theory*, pages 368–381, Tokyo, 1990. Ohmsha.

R. Olsson. Inductive functional programming using incremental program transformation. *Artificial Intelligence*, 74(1):55–83, 1995.

G. D. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1969.

J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

M. R. K. Krishna Rao. Inductive inference of term rewriting systems from positive data. In *Algorithmic Learning Theory*, pages 69–82, 2004.

M. R. K. Krishna Rao and A. Sattar. Polynomial-time learnability of logic programs with local variables from entailment. *Theoretical Computer Science*, 268(2):179–198, 2001.

U. Schmid. *Inductive Synthesis of Functional Programs – Universal Planning, Folding of Finite Programs, and Schema Abstraction by Analogical Reasoning*. Springer, 2003.

U. Schmid and F. Wysotzki. Applying inductive programm synthesis to macro learning. In *Proc. 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, pages 371–378. AAAI Press, 2000.

J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

D. R. Smith. The synthesis of LISP programs from examples: A survery. In A. W. Biermann, G. Guiho, and Y. Kodratoff, editors, *Automatic Program Construction Techniques*, pages 307–324. Macmillan, 1984.

P. D. Summers. A methodology for LISP program construction from examples. *Journal ACM*, 24(1):162–175, 1977.

L. G. Valiant. A theory of the learnable. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445, New York, NY, USA, 1984. ACM Press.

F. Wysotzki and U. Schmid. Synthesis of recursive programs from finite examples by detection of macro-functions. Technical Report 01-2, Dept. of Computer Science, TU Berlin, Germany, 2001.

Journal of Machine Learning Research 7 (2006) 455–491 Submitted 12/04; Revised 10/05; Published 2/06

Optimising Kernel Parameters and Regularisation Coefficients for Non-linear Discriminant Analysis

Tonatiuh Peña Centeno TPENA@DCS.SHEF.AC.UK
Neil D. Lawrence NEIL@DCS.SHEF.AC.UK
Department of Computer Science
The University of Sheffield
Regent Court, 211 Portobello Street
Sheffield, S1 4DP, U.K.

Editor: Greg Ridgeway

Abstract

In this paper we consider a novel Bayesian interpretation of Fisher's discriminant analysis. We relate Rayleigh's coefficient to a noise model that minimises a cost based on the most probable class centres and that abandons the 'regression to the labels' assumption used by other algorithms. Optimisation of the noise model yields a direction of discrimination equivalent to Fisher's discriminant, and with the incorporation of a prior we can apply Bayes' rule to infer the posterior distribution of the direction of discrimination. Nonetheless, we argue that an additional constraining distribution has to be included if sensible results are to be obtained. Going further, with the use of a Gaussian process prior we show the equivalence of our model to a regularised kernel Fisher's discriminant. A key advantage of our approach is the facility to determine kernel parameters and the regularisation coefficient through the optimisation of the marginal log-likelihood of the data. An added bonus of the new formulation is that it enables us to link the regularisation coefficient with the generalisation error.

1. Introduction

Data analysis typically requires a preprocessing stage to give a more parsimonious representation of data, such preprocessing consists of selecting a group of characteristic features according to an optimality criterion. Tasks such as data description or discrimination commonly rely on this preprocessing stage. For example, Principal Component Analysis (PCA) describes data more efficiently by projecting it onto the principal components and then by minimising the reconstruction error, see e.g. (Jolliffe, 1986). In contrast, Fisher's linear discriminant (Fisher, 1936) separates classes of data by selecting the features[1] that maximise the ratio of projected class means to projected intraclass variances.

The intuition behind Fisher's linear discriminant (FLD) consists of looking for a vector of compounds $\mathbf{w}$ such that, when a set of training samples are projected on to it, the class centres are far apart while the spread within each class is small, consequently producing a small overlap between classes (Schölkopf and Smola, 2002). This is done by maximising a cost function known in some contexts as Rayleigh's coefficient, $J(\mathbf{w})$. Kernel Fisher's discriminant (KFD) is a nonlinearisation

1. In Fisher's terminology the features are grouped into a vector of 'compounds'.

that follows the same principle but in a typically high-dimensional feature space $\mathcal{F}$. In this case, the algorithm is reformulated in terms of $J(\alpha)$, where α is the new direction of discrimination. The theory of reproducing kernels in Hilbert spaces (Aronszajn, 1950) gives the relation between vectors $\mathbf{w}$ and α, see Section 5.1. In either case, the objective is to determine the most 'plausible' direction according to the statistic J.

Mika et al. (1999) demonstrated that KFD can be applied to classification problems with competitive results. KFD shares many of the virtues of other kernel based algorithms: the appealing interpretation of a kernel as a mapping of an input to a high dimensional space and good performance in real life applications, among the most important. However, it also suffers from some of the deficiencies of kernelised algorithms: the solution will typically include a regularisation coefficient to limit model complexity and parameter estimation will rely on some form of cross validation. Unfortunately, there is no principled approach to set the former, while the latter precludes the use of richer models.

In this paper we introduce a novel probabilistic interpretation of Fisher's discriminant. Classical FLD is revised in Section 2 while an alternative noise model is outlined in Section 3. We build on the model in Section 4 by first applying priors over the direction of discrimination to develop a *Bayesian* Fisher discriminant and later we use a Gaussian process prior to reformulate the problem. In Section 5, we compare our model to other approaches. We explore the connections of our model to the expected generalisation error in Section 6. Section 7 details an EM-based algorithm for estimating the parameters of the model (kernel and regularisation coefficients) by optimising the marginal log likelihood. We present the results of our approach by applying it on toy data and by classifying benchmark data sets, in Section 8. Finally we address future directions of our work in Section 9.

2. Fisher's Discriminant Analysis

As mentioned above, discriminant analysis involves finding a vector of compounds $\mathbf{w} \in \mathbb{R}^{d\times 1}$ for which class separation will be maximised according to some defined statistic. Considering a set of training data and labels, $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \left\{\mathbf{x}^{(n)}, y^{(n)}\right\}_{n=1}^{N} \in \mathbb{R}^{N\times(d+1)}$, the discriminant reduces the dimensionality of the data through a linear combination, such that a set of single variates $\left\{\left(\mu_1, \sigma_1^2\right), \left(\mu_0, \sigma_0^2\right)\right\}$ is produced; where we define $\left(\mu_q, \sigma_q^2\right)$ as the sample mean and variance of each projected group. The hope is that both groups will be distinguished from one another by using this new set. Fisher was the first to conclude that the compounds should be given by maximising the ratio of between to within class variances,

$$J = \frac{\left(\mu_1 - \mu_0\right)^2}{\sigma_1^2 + \sigma_0^2}. \tag{1}$$

We will use the following definitions. A vector of projections is generated by taking the product $\mathbf{f} = \mathbf{X}\mathbf{w} \in \mathbb{R}^{N\times 1}$ and the sample means for each class are $\mathbf{m}_q = N_q^{-1}\sum_{n\in N_q}\mathbf{x}_q^{(n)}$, hence the projected mean and variance are given by

$$\begin{aligned} \mu_q &= N_q^{-1}\mathbf{w}^T\mathbf{m}_q \\ &= N_q^{-1}\mathbf{f}^T\mathbf{y}_q, \end{aligned} \tag{2}$$

and

$$\begin{aligned}\sigma_q^2 &= \sum_{n\in N_q}\left(\mathbf{w}^T\mathbf{x}_q^{(n)}-\mu_q\right)^2 \\ &= \sum_{n\in N_q}\left(f^{(n)}-\mu_q\right)^2, \end{aligned} \tag{3}$$

respectively. Abusing the notation, we have split the training data into two disjoint groups $(\mathbf{X},\mathbf{y}) = (\mathbf{X}_0,\mathbf{y}_0)\cup(\mathbf{X}_1,\mathbf{y}_1)$, with $y_q^{(n)}\in\{0,1\}$. The coefficient N_q is the cardinality of each group, $q\in\{0,1\}$.

Modern texts on pattern recognition and machine learning (Fukunaga, 1990; Duda and Hart, 1973; Bishop, 1995; Ripley, 1996) prefer to make explicit the dependence of this statistic on the vector of compounds. Hence, with some manipulation and the introduction of a couple of matrices we arrive at

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\Sigma_B\mathbf{w}}{\mathbf{w}^T\Sigma_w\mathbf{w}}, \tag{4}$$

where $\Sigma_B = (\mathbf{m}_1-\mathbf{m}_0)(\mathbf{m}_1-\mathbf{m}_0)^T$ and $\Sigma_w = \sum_{q\in\{0,1\}}\sum_{n=1}^{N_q}\left(\mathbf{x}_q^{(n)}-\mathbf{m}_q\right)\left(\mathbf{x}_q^{(n)}-\mathbf{m}_q\right)^T$, are between and within covariance matrices respectively. Matrix Σ_B measures the separation between class means while Σ_w gives an estimation of the spread around them. A solution for this problem consists of taking the derivative of Equation 4 w.r.t. $\mathbf{w}$ and solving. This leads to a generalised eigenvalue problem of the form $\Sigma_w^{-1}\Sigma_B\mathbf{w} = \lambda\mathbf{w}$, with λ being the eigenvalues. A solution for the discriminant can also be derived from geometric arguments. Given a test point $\mathbf{x}^\star$, the discriminant is a hyperplane $D(\mathbf{x}^\star) = \mathbf{w}^T\mathbf{x}^\star + b$, that outputs a number according to the class membership of the test point, where b is a bias term. In this context $\mathbf{w}$ is a vector that represents the direction of discrimination. Following this line, the solution $\mathbf{w}\propto\Sigma_w^{-1}(\mathbf{m}_0-\mathbf{m}_1)$ is sometimes easier to interpret than the eigenvalue problem.

As it was demonstrated by Mika (2001), a more detailed analysis of FLD allows it to be cast as a quadratic programming problem. In order to do so, we observe that the magnitude of the solution is not relevant, so for example, the numerator of Equation 1 can be fixed to an arbitrary scalar while the denominator is minimised. In other words, the variance of the projections is minimised while the distance between projected means is kept at, say $d = \mu_0-\mu_1$. Rayleigh's statistic can then be written as $J = d^2/\left(\sigma_1^2+\sigma_0^2\right)$. The subsequent discussion will make use of this 'average distance' constraint to reformulate the discriminant problem.

3. Probabilistic Interpretation

We introduce some notation that will be used throughout the rest of the paper. The set of variables $\mathcal{D} = (\mathbf{X},\mathbf{y})\in\mathbb{R}^{N\times(d+1)}$ is observed or instantiated, $\mathbf{f}\in\mathbb{R}^{N\times1}$ is a dependent or latent variable and $\mathbf{t}\in\mathbb{R}^{N\times1}$ is a vector of targets that have been observed as well. The random variables will follow some probability law and in this model, in particular, we study the relationship between observed and latent variables: the noise model. From Section 2, we know that every observation in $\mathcal{D}$ is projected into a single variate that ideally can take only two values which are the projected class centres, where the variance around the projections tries to be minimised. We define the parameters c_0 and c_1 as the true class centres in the projected space. Additionally, we introduce a precision β that corresponds to the variance around the projected data. Because of the nature of the mapping

process, it is convenient to define some auxiliary variables as well, $\mathbf{t}_1$ is a vector filled with c_1's whenever $y^{(n)} = 1$ and filled with zeros otherwise; $\mathbf{t}_0$ is a vector filled with c_0's whenever $y^{(n)} = 0$ and with zeros otherwise. We also take $\mathbf{y}_1 = \mathbf{y}$ and $\mathbf{y}_0 = \mathbf{1} - \mathbf{y}$ and denote by $\hat{\mathbf{v}}$ the maximum likelihood estimate of a vector/scalar $\mathbf{v}$.

3.1 The Noise Model

Figure 1 models the causal relationship between the observations $\mathcal{D}$ and the variables $\mathbf{f}$ and $\mathbf{t}$, such that the distribution $p(\mathbf{f},\mathbf{t}|\mathcal{D},)$ can be decomposed into noise model $p(\mathbf{t}|\mathbf{y},\mathbf{f})$ and prior $p(\mathbf{f}|\mathbf{X})$, disregarding the parameter β. For the moment, we will ignore the prior and consider only the noise model. In graphical notation every fully shaded circle corresponds to an observed variable and a blank circle indicates a latent variable. We make use as well of partially shaded circles to indicate the binary nature of the discriminant, that is, that targets should only take one of two different values. In Figure 1 the variable $t_1^{(n)}$ is observed whenever $y^{(n)} = 1$; and $t_0^{(n)}$, whenever $y^{(n)} = 0$. Both variables $\mathbf{t}_0$ and $\mathbf{t}_1$ are discrete, with each of their elements being given by the class centres c_0 and c_1, nevertheless, we will make a Gaussian[2] approximation such that every element $t_q^{(n)} \sim \mathcal{N}\left(f^{(n)}, \beta^{-1}\right)$. From this approximation the noise model can be defined as

$$p(\mathbf{t}|\mathbf{y},\mathbf{f},\beta) = \frac{\beta^{\frac{N}{2}}}{(2\pi)^{\frac{N}{2}}} \exp\left\{-\frac{\beta}{2}\sum_{q\in\{0,1\}} (\mathbf{t}_q - \mathbf{f})^T \operatorname{diag}(\mathbf{y}_q)(\mathbf{t}_q - \mathbf{f})\right\}. \tag{5}$$

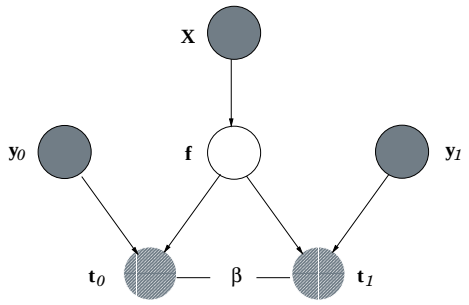


Figure 1: The proposed graphical model for discriminant analysis. The graph models the joint distribution over the latent variables $\mathbf{f}$ and the targets $\mathbf{t} = \mathbf{t}_0 \cup \mathbf{t}_1$, which have been decomposed into their two possible types. Disregarding the parameter β, the joint probability is factorised as $p(\mathbf{f},\mathbf{t}|\mathcal{D}) = p(\mathbf{t}|\mathbf{y},\mathbf{f})\,p(\mathbf{f}|\mathbf{X})$, where the noise model is given by $p(\mathbf{t}|\mathbf{y},\mathbf{f})$ and the prior by $p(\mathbf{f}|\mathbf{X})$. Note that we express the labels into two different groups $\mathbf{y}_0$ and $\mathbf{y}_1$. Shaded nodes indicate instantiated variables, blank ones correspond to latent variables and partially shaded ($\mathbf{t}_0$ and $\mathbf{t}_1$) nodes are only observed according to the values of the labels ($\mathbf{y}_0$ and $\mathbf{y}_1$, respectively). We assume that every observed target is distributed according to $t_q^{(n)} \sim \mathcal{N}\left(f^{(n)}, \beta^{-1}\right)$, where β is the precision parameter.

As it can be observed from both the figure and Equation 5, there is a conditional independence assumption on the observed targets given $\mathbf{y}$ and $\mathbf{f}$; in other words, the noise model can be further

2. We use the notation $\mathcal{N}(\mathbf{x}|\mathbf{m},\Sigma)$ to indicate a multivariate Gaussian distribution over $\mathbf{x}$ with mean $\mathbf{m}$ and covariance Σ.

decomposed as $p(\mathbf{t}|\mathbf{y},\mathbf{f}) = p(\mathbf{t}_0|\mathbf{y}_0,\mathbf{f})\,p(\mathbf{t}_1|\mathbf{y}_1,\mathbf{f})$, where we have disregarded the dependence on β.

We can substitute every element $t_q^{(n)}$ by its class centre c_q and take the log of (5) to obtain

$$\mathcal{L}(\mathbf{f},\beta) \quad = \quad -\frac{\beta}{2}\sum_{n=1}^{N}\left[y^{(n)}\left(c_1 - f^{(n)}\right)^2 + \left(1-y^{(n)}\right)\left(c_0 - f^{(n)}\right)^2\right] + \mathrm{C}, \tag{6}$$

where $\mathrm{C} = \dfrac{N}{2}\log\dfrac{\beta}{2\pi}$.
Note that the class centres can be made to coincide with the labels. In such a 'regression to the labels' scheme, FLD can be recovered in a straightforward manner.

3.1.1 MAXIMUM LIKELIHOOD

Parameter estimates can be found by zeroing the gradient of $\mathcal{L}$ with respect to each $f^{(n)}$ and β and solving the resulting expressions for each parameter. This leads to the fixed point equations

$$\hat{f}^{(n)} = \left(1-y^{(n)}\right)c_0 + y^{(n)}c_1 \tag{7}$$

and

$$\hat{\beta} = \frac{N}{\sum_{n=1}^{N} y_n\left(c_1 - f^{(n)}\right)^2 + \sum_{n=1}^{N}(1-y_n)\left(c_0 - f^{(n)}\right)^2}. \tag{8}$$

However, the values of the class centres c_0 and c_1 are not known, so $\mathcal{L}$ can also be maximised w.r.t. them to obtain

$$\hat{c}_q = \frac{1}{N_q}\sum_{n=1}^{N_q} y_q^{(n)} f^{(n)} \text{ for} q \in \{0,1\}. \tag{9}$$

The results $\hat{f}^{(n)}$ and $\hat{c}_q$ suggest applying an iterative scheme to find the maximum. This can be done by substituting $\hat{f}^{(n)}$ and $\hat{c}_q$ on the right hand sides of Equations 9 and 7, respectively, initialising one of the variables to an arbitrary value and updating all of them until convergence.

3.2 Model Equivalence

We now turn to the connections between Rayleigh's statistic and the proposed noise model. In particular, we want to show that maximum likelihood learning in our framework is equivalent to maximisation of Rayleigh's coefficient. In order to do so, we back substitute the values $\hat{c}_q$ into $\mathcal{L}$ (Equation 6) compute the gradient w.r.t β and solve the resulting expression for β. The substitution of each class centre by their most probable values is indispensable and central to our framework. As a result of this substitution we can create a cost function that reduces the error around the most probable class centres. The solution for β leads to an expression of the form

$$\hat{\beta} = \frac{N}{\sigma_1^2 + \sigma_0^2},$$

with σ_q^2 defined in Equation 3, for $q \in \{0,1\}$, and where we have recognised that Equation 2 is equivalent to Equation 9. The result above is proportional to the constrained version of Rayleigh's

quotient mentioned before, $J = d^2 / \left(\sigma_1^2 + \sigma_0^2\right)$, hence we can write

$$J(\mathbf{f}) = \frac{d^2 \hat{\beta}}{N}. \tag{10}$$

It is clear that this quantity monotonically increases over the domain $\mathbb{R}^+$ because $\hat{\beta}$ can only take positive values. Meanwhile the likelihood, the exponential of Equation 6, expressed in terms of the estimate $\hat{\beta}$ takes the form

$$L(\mathbf{f}) = \frac{\hat{\beta}^{N/2}}{(2\pi)^{N/2}} \exp\left\{-\frac{N}{2}\right\}, \tag{11}$$

which is monotonic as well on this estimate.

Therefore, as Equations 10 and 11 are monotonic in $\hat{\beta}$, their maximisation with respect to this parameter must yield equivalent results.

3.3 Parametric Noise Model

In this section we make two modifications to Equations 5 and 6 in order to parameterise the noise model. First, the vector of targets $\mathbf{t}$ is replaced by a new vector filled with the estimates $\hat{c}_q$ such that $\hat{\mathbf{t}} = \hat{\mathbf{t}}_0 \cup \hat{\mathbf{t}}_1$ is generated. Second, every latent variable is related to the observations via a vector of parameters $\mathbf{w}$. In a linear relation this is expressed by the inner product $f^{(n)} = \mathbf{w}^T \mathbf{x}^{(n)}$. Therefore after making these changes the log-likelihood becomes

$$\mathcal{L} \;=\; -\frac{\beta}{2} \sum_{n=1}^{N} \left[y^{(n)} \left(\hat{c}_1 - \mathbf{w}^T \mathbf{x}^{(n)} \right)^2 + \left(1 - y^{(n)} \right) \left(\hat{c}_0 - \mathbf{w}^T \mathbf{x}^{(n)} \right)^2 \right] + \mathrm{C}. \tag{12}$$

Thus a new probabilistic model is obtained, which is depicted in Figure 2.

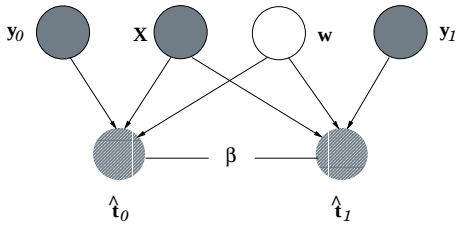


Figure 2: Partially modified graphical model for discriminant analysis. In comparison with Figure 1, the latent variable $\mathbf{f}$ has been replaced by a vector of parameters $\mathbf{w}$. Ignoring the parameter β, the graph factorises the joint distribution $p\left(\hat{\mathbf{t}}, \mathbf{w} \middle| \mathcal{D}\right)$ with the product $p\left(\hat{\mathbf{t}} \middle| \mathcal{D}, \mathbf{w}\right) \times p(\mathbf{w})$, where $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ is the training data; $\hat{\mathbf{t}} = \hat{\mathbf{t}}_1 \cup \hat{\mathbf{t}}_0$, the modified targets and $\mathbf{y}_0$ and $\mathbf{y}_1$ are the class labels. The log of the noise model $p\left(\hat{\mathbf{t}} \middle| \mathcal{D}, \mathbf{w}\right)$ is expressed in Equation 12 while the prior $p(\mathbf{w})$ is specified in Section 4.

Furthermore, we look not only to parameterise the latent variables, but the class centres as well. Equation 9 can be used to this purpose, substituting every $f^{(n)}$ in it with their parametric versions $\mathbf{w}^T \mathbf{x}^{(n)}$ leads to $\hat{c}_q = \frac{1}{N_q} \sum_{n=1}^{N_q} y_q^{(n)} \mathbf{w}^T \mathbf{x}^{(n)}$. The vector of parameters can be pulled out of

the summation and leave a quantity that we recognise to be the sample mean for class q, which we express as $\mathbf{m}_q$. Hence we can write $\hat{c}_q = \mathbf{w}^T \mathbf{m}_q$. Therefore the log of the new noise model can be expressed as

$$\mathcal{L} = -\frac{\beta}{2} \sum_{n=1}^{N} \left[y^{(n)} \left(\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{x}^{(n)} \right)^2 + \left(1 - y^{(n)} \right) \left(\mathbf{w}^T \mathbf{m}_0 - \mathbf{w}^T \mathbf{x}^{(n)} \right)^2 \right] + \text{C}. \tag{13}$$

As it will be seen in Section 5, most models make the assumption that class centres and class labels coincide, that is $c_q = y_q$; including the least squares support vector machine of Suykens and Vandewalle (1999). However this approach is suboptimal because there is no guarantee that class centres should map perfectly with the labels. Instead of following this 'regression to the labels' assumption, we have preferred to make use of the maximum likelihood estimates of the class centres. As we saw above, by taking this step, the class centres can be parameterised as well.

3.3.1 MAXIMUM LIKELIHOOD

Maximisation of this new form of $\mathcal{L}$ (Equation 13) has to be carried out in a slightly different way to the one presented in Section 3.1.1. Previously, the class centres were parameters which we knew beforehand were separated by some given distance. However, their parameterisation implies that the separation constraint must be considered explicitly. We therefore introduce a Lagrange multiplier to force the projected class centres to lie at a distance d, leading to the following function

$$\begin{aligned} &\Lambda(\mathbf{w}, \lambda) = \\ &\quad -\frac{\beta}{2} \sum_{n=1}^{N} \left[y^{(n)} \left(\mathbf{w}^T \mathbf{m}_1 - \mathbf{w}^T \mathbf{x}^{(n)} \right)^2 + \left(1 - y^{(n)} \right) \left(\mathbf{w}^T \mathbf{m}_0 - \mathbf{w}^T \mathbf{x}^{(n)} \right)^2 \right] \\ &\qquad\qquad + \lambda \left[\mathbf{w}^T (\mathbf{m}_0 - \mathbf{m}_1) - d \right] + \text{C}. \end{aligned}$$

A solution for this constrained optimisation problem is given by

$$\hat{\mathbf{w}} = \frac{\lambda}{\beta} \Sigma_w^{-1} (\mathbf{m}_0 - \mathbf{m}_1),$$

with

$$\lambda = d\beta \left[(\mathbf{m}_0 - \mathbf{m}_1)^T \Sigma_w^{-1} (\mathbf{m}_0 - \mathbf{m}_1) \right]^{-1}.$$

Therefore, by letting $\Delta\mathbf{m} = \mathbf{m}_0 - \mathbf{m}_1$, we can express the solution as

$$\hat{\mathbf{w}} = \frac{d\Sigma_w^{-1} \Delta\mathbf{m}}{\Delta\mathbf{m}^T \Sigma_w^{-1} \Delta\mathbf{m}}, \tag{14}$$

which is equivalent to that produced by FLD up to a constant of proportionality (see Section 2).

This completes the discussion of an alternative noise model for FLD. The new probabilistic formulation is based on a noise model that reduces the error around the class centres, instead of the class labels. Furthermore, we were interested on parameterising not only the latent variables in the model but also the centres themselves. Through the introduction of a Lagrange multiplier we saw that a constrained maximisation of the new likelihood was equivalent to standard FLD.
In this section we made use only of one part of the graphical models presented in Figures 1 and 2. In the next section we complete the analysis by including the prior distributions that were left

unattended. First we complete the study of Figure 2 by incorporating a prior over the parameters, $p(\mathbf{w})$, and later study the model of Figure 1 under the assumption that the prior, $p(\mathbf{f}|\mathbf{X})$, is a Gaussian process.

4. Bayesian Formulation

One of the aims of discriminant analysis is to determine the group membership of an input $\mathbf{x}^\star$ outside the training set. From a probabilistic perspective this process is only possible if a noise model and a prior distribution have been identified. Then the posterior over the parameters $p(\mathbf{w}|\mathcal{D})$ can be found as well as the corresponding predictive distribution. The posterior distribution is important because it summarises the knowledge gained after having observed the training set. The application of a Bayesian probabilistic approach offers some intrinsic advantages over other methods, for example the ability to compute 'error bars' and, in the context of our model, the possibility to introduce Gaussian process priors in a natural way.

This section will show that the introduction of a separable Gaussian prior over $\mathbf{w}$ leads to a posterior distribution that is not enough to recover FLD's solution. Later on, it will be argued that an additional step is required to ensure the equivalence is achieved. This additional step will also include the distance constraint previously implemented through a Lagrange multiplier.

4.1 Weight Space Formulation

So far we have found a maximum likelihood estimate of the parameters' vector (see Equation 14). Now what we seek is a distribution over this vector which is obtained by combining the noise model with a prior distribution through Bayes' rule,

$$p\left(\mathbf{w}|\hat{\mathbf{t}},\mathcal{D}\right)=\frac{p\left(\hat{\mathbf{t}}|\mathcal{D},\mathbf{w}\right)p(\mathbf{w})}{p\left(\hat{\mathbf{t}}|\mathcal{D}\right)},$$

where we have used $\mathcal{D}$ to indicate the training set $(\mathbf{X},\mathbf{y})$ and have omitted the dependence on β.

A common choice of prior is a separable Gaussian, $p(\mathbf{w})=\mathcal{N}\left(\mathbf{w}|\mathbf{0},\mathbf{A}^{-1}\right)$, whith zero mean and diagonal covariance $\mathbf{A}^{-1}$. The combination of this prior with the parametric noise model of Equation 13 gives a posterior of the form

$$\begin{aligned}p(\mathbf{w}|\hat{\mathbf{t}},\mathcal{D}) \quad\propto\quad & \exp\left\{-\frac{\beta}{2}\sum_{n=1}^{N}\left[y^{(n)}\left(\mathbf{w}^T\mathbf{m}_1-\mathbf{w}^T\mathbf{x}^{(n)}\right)^2+\ldots\right.\right.\\ & \left.\left.\left(1-y^{(n)}\right)\left(\mathbf{w}^T\mathbf{m}_0-\mathbf{w}^T\mathbf{x}^{(n)}\right)^2\right]-\frac{1}{2}\mathbf{w}^T\mathbf{A}\mathbf{w}\right\}. \qquad (15)\end{aligned}$$

In order to obtain a complete expression for $p(\mathbf{w}|\mathcal{D})$ it is necessary to define the normalisation constant. As the expression is quadratic in $\mathbf{w}$ we know the posterior distribution will be Gaussian. However, it is still necessary to specify the mean and covariance of the distribution. In order to do so, Bayesian methods take advantage of an important property of Gaussians: if two sets of variables are Gaussian, like $\hat{\mathbf{t}}$ and $\mathbf{w}$, then the conditional distribution of one set conditioned on the other is Gaussian as well. On the RHS of (15), we look to condition variable $\mathbf{w}$ on $\hat{\mathbf{t}}$. The process simply consists of considering the variable $\hat{\mathbf{t}}$ as being given and on grouping terms in $\mathbf{w}$. This leads to a Gaussian posterior of the form

$$p(\mathbf{w}|\mathcal{D})=\mathcal{N}\left(\mathbf{w}|\mathbf{0},\mathbf{B}^{-1}\right),$$

with **zero** mean and covariance matrix $\mathbf{B} = \beta \mathbf{X}^T \mathbf{L} \mathbf{X} + \mathbf{A}$, where

$$\mathbf{L} = \mathbf{I} - N_1^{-1} \mathbf{y}_1 \mathbf{y}_1^T - N_0^{-1} \mathbf{y}_0 \mathbf{y}_0^T. \tag{16}$$

The posterior obtained is not equivalent to FLD because the mean of $\mathbf{w}$ is zero. In consequence, the posterior mean projection of any $\mathbf{x}^\star$ will collapse to the origin. Nonetheless, this formulation yields a consistent result if we consider that standard discriminant analysis exhibits a sign symmetry for the vector $\mathbf{w}$, hence the average is zero. What our new model is missing is the incorporation of the distance constraint. In Section 3.3.1, knowledge about the variable d was incorporated to the noise model in the form of a Lagrange multiplier. We look to do the same again but in a Bayesian approach this requires that we deal with every variable in terms of probability distributions.

We propose to use the posterior $p(\mathbf{w}|\mathcal{D})$ as the prior for a new model that is depicted in Figure 3. In the new formulation, d is considered an extra random variable that has been observed and that depends on the distribution over $\mathbf{w}|\mathcal{D}$. From the figure we can deduce that the joint factorises as $p(d,\mathbf{w}|\mathcal{D},\gamma) = p(d|\mathcal{D},\mathbf{w},\gamma)\, p(\mathbf{w}|\mathcal{D})$, with γ being a positive parameter. Note that this time we have made $\mathcal{D} = \left(\hat{\mathbf{t}}, \mathbf{X}, \mathbf{y}\right)$.

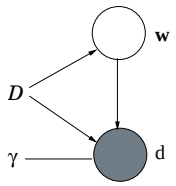


Figure 3: Graphical model to constrain the projected distance d. The graph specifies the distribution $p(d,\mathbf{w}|\mathcal{D},\gamma)$ which is composed by the distributions $p(\mathbf{w}|\mathcal{D})$ and $p(d|\mathcal{D},\mathbf{w},\gamma)$. The former is the posterior over the direction of discrimination, described in Section 4.1, and the latter is the constraining distribution, defined in Equation 17.

One of our main concerns is to keep the model tractable at all stages, but we are also interested in having a realistic representation of the discriminant. In order to guarantee both conditions we assume d is Gaussian with infinite precision γ,

$$p(d|\mathcal{D},\mathbf{w},\gamma) = \lim_{\gamma\to\infty} \frac{\gamma^{\frac{1}{2}}}{\sqrt{2\pi}} \exp\left(-\frac{\gamma}{2}\left(d - \mathbf{w}^T \Delta\mathbf{m}\right)^2\right). \tag{17}$$

We can see that this distribution introduces the same effect as the Lagrangian of Section 3.3.1 by placing all its mass at the point $d = \mu_0 - \mu_1$ when the limit $\gamma \to \infty$ is taken.

The process to determine a posterior $p(\mathbf{w}|\mathcal{D},d)$ is based on combining $p(\mathbf{w}|\mathcal{D})$ with $p(d|\mathcal{D},\mathbf{w},\gamma)$ and then conditioning $\mathbf{w}$ on d. However, a final step needs to be added to work out the limit to eliminate the dependence over γ. As a partial result, the conditional distribution $p(\mathbf{w}|\mathcal{D},d,\gamma)$ will be $\mathcal{N}(\mathbf{w}|\bar{\mathbf{w}},\Sigma)$ with mean

$$\bar{\mathbf{w}} = \lim_{\gamma\to\infty} \gamma d \Sigma \Delta\mathbf{m},$$

and covariance

$$\Sigma = \lim_{\gamma\to\infty} \left(\mathbf{B} + \gamma \Delta\mathbf{m}\Delta\mathbf{m}^T\right)^{-1}.$$

With some algebraic manipulations and the application of the Morrison-Woodbury formula (Golub and Van Loan, 1996) we can arrive to the desired result. See Appendix A for the detailed derivation. After taking the limit, the resulting distribution will be a Gaussian

$$p(\mathbf{w}|\,\mathcal{D},d) = \mathcal{N}(\mathbf{w}|\bar{\mathbf{w}},\Sigma)$$

with parameters

$$\bar{\mathbf{w}} \;=\; \frac{d\mathbf{B}^{-1}\Delta\mathbf{m}}{\Delta\mathbf{m}^T\mathbf{B}^{-1}\Delta\mathbf{m}}$$

and

$$\Sigma \;=\; \mathbf{B}^{-1} - \frac{\mathbf{B}^{-1}\Delta\mathbf{m}\Delta\mathbf{m}^T\mathbf{B}^{-1}}{\Delta\mathbf{m}^T\mathbf{B}^{-1}\Delta\mathbf{m}}.$$

Noticing that $\mathbf{B} = \beta\mathbf{X}^T\mathbf{L}\mathbf{X} + \mathbf{A}$, the mean of the new posterior coincides with the maximum likelihood solution of Section 3.3 when an improper prior is used (i.e. $\mathbf{A} = \lim_{\alpha\to\infty}\alpha\mathbf{I}$). Note that the matrix Σ is positive semidefinite and therefore not invertible, this is a consequence of the fact that any vector $\mathbf{w}$ which does not satisfy the constraint imposed by the distribution $p(\,d|\,\mathcal{D},\mathbf{w},\gamma)$ has a posterior probability of zero. Nevertheless, variances associated with the posterior projections can still be computed by applying

$$\mathrm{var}\left(\mathbf{w}^T\mathbf{x}\right) = \mathbf{x}^T\mathbf{B}^{-1}\mathbf{x} - \frac{\mathbf{x}^T\mathbf{B}^{-1}\Delta\mathbf{m}\Delta\mathbf{m}^T\mathbf{B}^{-1}\mathbf{x}}{\Delta\mathbf{m}^T\mathbf{B}^{-1}\Delta\mathbf{m}},$$

which will be zero if the point $\mathbf{x}$ is on the direction of $\Delta\mathbf{m}$.

The Bayesian approach we have outlined leads to a posterior distribution over the direction of discrimination which can be used to compute expected outputs and their associated variances for any given input $\mathbf{x}$. However, the limitation imposed by applying a linear model is a strong one. There is an extensive amount of literature explaining why linear models are not always convenient. A common solution is to use a set of nonlinear basis functions ϕ such that the new function is linear in the parameters but nonlinear in the input space $f = \mathbf{w}^T\phi(\mathbf{x})$, see for example (Ruppert et al., 2003) and (Bishop, 1995). However the problem is shifted to that of specifying which and what number of basis functions to use. In the next section we shall consider the alternative approach of placing a prior directly over the vector of projections $\mathbf{f}$, such that we will be working with a possibly infinite amount of basis functions. This approach will lead to a regularised version of kernel Fisher's discriminant and ultimately to an alternative strategy to select model parameters.

4.2 Gaussian Process Formulation

The choice of a Gaussian probability measure over functions has been justified by the study of the limiting prior distribution in the neural network case when the number of hidden units 'reaches' infinity, (Neal, 1996). A Gaussian process (GP) is a type of stochastic process that is defined by a mean and a covariance function. By stochastic process we understand that a countable infinite set of observations $\{f_1,\ldots,f_N\}$ has been sampled from a common probability distribution.

In GP's (O'Hagan, 1978) a prior is placed directly over the latent variables such that a posterior distribution over them can be inferred. Although there are many GP's with an equivalent 'weight space' prior, there exists a large class of them for which no finite dimensional expansion exists. In

this regard, a covariance function (or kernel) measures *a priori* the expected correlation between any two pair of points $\mathbf{x}^{(n)}$ and $\mathbf{x}^{(m)}$ in the training set. For example, in a function parameterised as

$$f^{(n)} = \mathbf{w}^T \phi\left(\mathbf{x}^{(n)}\right),$$

with a prior over $\mathbf{w}$ specified by a spherical Gaussian with zero mean, $p(\mathbf{w}) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}\right)$, the implied correlation between two points is

$$E\left[f^{(n)}, f^{(m)} \middle| \mathbf{w}\right] = \alpha^{-1}\phi\left(\mathbf{x}^{(n)}\right)^T \phi\left(\mathbf{x}^{(m)}\right).$$

In other words, provided that the product is positive and symmetric, the correlation between the two points will lead to a Mercer kernel; see (Schölkopf and Smola, 2002). However, under these circumstances it no longer makes sense to talk about a prior over the vector $\mathbf{w}$, but rather a prior over instantiations of the functions is considered.

4.2.1 PREDICTION OVER A TEST POINT

In order to adopt GP's we need to go back to the formulation of the discriminant presented in Figure 1. In this figure the graph models the joint distribution $p(\mathbf{f}, \mathbf{t}|\mathcal{D})$ with the product of noise model $p(\mathbf{t}|\mathbf{y}, \mathbf{f})$ and prior $p(\mathbf{f}|\mathbf{X})$. In this section we need to make two assumptions before doing any kind of prediction. First of all, the joint distribution over every instance f belonging to the training set or not will be a multivariate Gaussian, that is a GP. Secondly, we will continue to work with the maximum likelihood estimates of the class centres, which were denoted $\hat{c}_q$. In other words, if we use Equation 9 to form a vector $\hat{\mathbf{t}}$ and substitute it into Equation 5 we will obtain the distribution $p\left(\hat{\mathbf{t}}|\mathbf{y}, \mathbf{f}\right)$.

Following the steps of the previous section, we could work out the posterior distribution $p\left(\mathbf{f}|\hat{\mathbf{t}}, \mathcal{D}\right)$. However, this is not what we are looking for because what we truly want is to make predictions out of new test data. Therefore, what we seek ultimately is the distribution $p(f^\star|\mathcal{D}, d)$, where the distance variable d has been included. In order to do so, first we propose to compute the joint distribution $p\left(\hat{\mathbf{t}}, d, \mathbf{f}_+ \middle| \mathbf{y}, \gamma\right)$, where the variable $\mathbf{f}_+$ is given by an extended vector of the form $\mathbf{f}_+ = \left[\mathbf{f}^T, f^\star\right]^T$, with $f^\star$ being a point outside the training set. Second, the distribution $p(f^\star|\mathcal{D}, d)$ can be found from $p\left(\hat{\mathbf{t}}, d, \mathbf{f}_+ \middle| \mathbf{y}, \gamma\right)$ by marginalising out the variables $\mathbf{f}$ and conditioning the resulting distribution on the variables $\hat{\mathbf{t}}$ and d. Lastly, the dependence on the parameter γ can be eliminated by taking the limit $\gamma \to \infty$.

This process is facilitated if the joint distribution is factorised into well known factors. For example, $p\left(\hat{\mathbf{t}}, d, \mathbf{f}_+ \middle| \mathbf{y}, \gamma\right)$, can be given by the product of noise model, $p\left(\hat{\mathbf{t}}|\mathbf{y}, \mathbf{f}\right)$; Gaussian process prior $p(\mathbf{f}_+)$; and constraining distribution $p(d|\mathbf{y}, \mathbf{f}, \gamma)$. Firstly, the modified noise model is defined in terms of $\mathbf{f}$ by applying the values of $\hat{c}_q$ and rearranging, (see Appendix B). The result is

$$p\left(\hat{\mathbf{t}}|\mathbf{y}, \mathbf{f}\right) \quad \propto \quad \exp\left(-\frac{\beta}{2}\mathbf{f}^T\mathbf{L}\mathbf{f}\right), \tag{18}$$

with $\mathbf{L}$ defined in Equation 16. Secondly, let the augmented vector $\mathbf{f}_+$ be correlated with a covariance matrix $\mathbf{K}_+ \in \mathbb{R}^{(n+1)\times(n+1)}$, then the prior is a GP of the form

$$p(\mathbf{f}_+) \propto \exp\left(-\frac{1}{2}\mathbf{f}_+^T\mathbf{K}_+^{-1}\mathbf{f}_+\right). \tag{19}$$

For future reference, the inverse of $\mathbf{K}_+$ is partitioned as

$$\mathbf{K}_+^{-1} = \begin{pmatrix} \mathbf{C} & \mathbf{c} \\ \mathbf{c}^T & c_\star \end{pmatrix},$$

with

$$\begin{aligned} c_\star &= \left(k_\star - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}\right)^{-1}, \\ \mathbf{c} &= -c_\star\mathbf{K}^{-1}\mathbf{k}, \\ \mathbf{C} &= \mathbf{K}^{-1} + c_\star\mathbf{K}^{-1}\mathbf{k}\mathbf{k}^T\mathbf{K}^{-1}. \end{aligned}$$

Note that the vector $\mathbf{k} \in \mathbb{R}^{N\times 1}$ is filled with scalars $k_{(n)} = \mathbf{K}\left(\mathbf{x}^{(n)}, \mathbf{x}\right)$ for $\mathbf{x} \in \mathcal{X}$.
Finally, the model still needs to consider that projected class means must be separated by the distance d. The introduction of a constraining distribution of the form of Equation 17 is what is needed. We can express this distribution in terms of $\mathbf{f}$ by replacing the term $\mathbf{w}^T\Delta\mathbf{m}$ inside the exponential by $\mathbf{f}^T\Delta\hat{\mathbf{y}}$, where $\Delta\hat{\mathbf{y}} = N_0^{-1}\mathbf{y}_0 - N_1^{-1}\mathbf{y}_1$. Therefore the constraint becomes

$$p\left(d|\mathbf{y},\mathbf{f},\gamma\right) = \lim_{\gamma\to\infty}\frac{\gamma^{\frac{1}{2}}}{\sqrt{2\pi}}\exp\left(-\frac{\gamma}{2}\left(d - \mathbf{f}^T\Delta\hat{\mathbf{y}}\right)^2\right). \tag{20}$$

Hence we can write the marginal distribution (after marginalisation of $\mathbf{f}$) as

$$p\left(f^\star, \hat{\mathbf{t}}, d\middle|\mathbf{y},\gamma\right) = \int p\left(\hat{\mathbf{t}}\middle|\mathbf{y},\mathbf{f}\right)p\left(d|\mathbf{y},\mathbf{f},\gamma\right)p\left(\mathbf{f}_+\right)\partial\mathbf{f}.$$

This is a Gaussian integral that can be solved straightforwardly by applying (for example) the material on exponential integrals (Bishop, 1995) that we present in Appendix C. After conditioning $f^\star$ on both $\hat{\mathbf{t}}$ and d, the solution is a Gaussian of the form

$$p\left(f^\star|\mathcal{D},d,\gamma\right) \propto \exp\left\{-\frac{1}{2\left(\sigma^\star\right)^2}\left(f^\star - \bar{f}^\star\right)^2\right\}$$

with mean

$$\bar{f}^\star = \lim_{\gamma\to\infty} -\gamma d\left(\sigma^\star\right)^2\mathbf{c}^T\mathbf{Q}^{-1}\Delta\hat{\mathbf{y}}.$$

and variance

$$\left(\sigma^\star\right)^2 = \lim_{\gamma\to\infty}\left(c_\star - \mathbf{c}^T\mathbf{Q}^{-1}\mathbf{c}\right)^{-1},$$

where we have defined the matrix $\mathbf{Q} = \beta\mathbf{L} + \mathbf{C} + \gamma\Delta\hat{\mathbf{y}}\Delta\hat{\mathbf{y}}^T$.

Just as in Section 4.1, the dependence on γ is eliminated by taking the limit as $\gamma\to\infty$. This procedure is detailed in Appendix C. The parameters of the distribution are

$$\bar{f}^\star = \frac{d\mathbf{k}^T\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}}{\Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}}, \tag{21}$$

and

$$(\sigma^\star)^2 = k_\star - \mathbf{k}^T \left(\mathbf{K}^{-1} - \mathbf{D}^{-1}\right) \mathbf{k}, \tag{22}$$

with the matrices

$$\mathbf{D} = \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}} \left(\Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}\right)^{-1} \Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1}\right)^{-1}$$

and

$$\mathbf{A} = \beta\mathbf{K}\mathbf{L}\mathbf{K} + \mathbf{K}. \tag{23}$$

The predictive mean is given by a linear combination of the observed labels, in this case expressed by $\Delta\hat{\mathbf{y}}$. Additionally, the predictive variance is composed by two terms, one representing the test point and the other representing the observed data. These results are similar to those of typical GP regression, described in (Williams, 1999). The scheme proposed above will be termed Bayesian Fisher's discriminant (BFD) to facilitate its referencing.

5. Relationship with Other Models

There are several well known connections between discriminant analysis and other techniques. In the statistics community, FLD is equivalent to a t-test or F-test for significant difference between the mean of discriminants for two sampled classes, in fact, the statistic is designed to have the largest possible value (Michie et al., 1994). In this section, however, we prefer to explore the connections of our approach to some algorithms that have been applied to machine learning problems, namely kernel Fisher's discriminant and the least-squares and proximal support vector machines.

5.1 Kernel Fisher's Discriminant

The algorithm known as kernel Fisher's discriminant consists of a two stage procedure. The first consists of embedding the data space $\mathcal{X}$ into a possibly infinite dimensional reproducing kernel Hilbert space $\mathcal{F}$ via a kernel function k. The second simply consists of applying FLD in this new data space. As the second stage is exactly the same as standard linear discriminant, many of the properties for FLD observed in $\mathcal{X}$ will hold also in $\mathcal{F}$; for example, some form of regularisation needs to be included. However there is an extra effort involved in preparing the original data for a new data representation in the induced space, namely in terms of the kernel function.

Data embedding is carried out by applying a non-linear transformation $\phi : \mathcal{X} \to \mathcal{F}$ that induces a positive definite kernel function. From the theory of reproducing kernels (Aronszajn, 1950) it is well known that the vector of compounds is a weighted combination of the training samples, such that $\mathbf{w} = \sum_{i=1}^{N} \alpha^{(i)} \phi\left(\mathbf{x}^{(i)}\right)$. The application of this property plus the decomposition of the kernel into its spectrum:

$$k(\mathbf{x}, \mathbf{x}') \quad = \quad \sum_{i=1}^{d} \lambda_i \phi_i(\mathbf{x}) \phi_i\left(\mathbf{x}'\right)$$

leads to the formulation of the Rayleigh coefficient in the feature space. Following the path of other kernel methods, the novelty in (Mika et al., 1999) resides in defining the kernel function directly and working without any reference to the spectral-based formulation.

A direct implication of working in an infinite dimensional space is that there is no form to express directly the matrices Σ_w and Σ_B. Nonetheless, the discriminant function can still be written

as the rule $D(\mathbf{x}^\star) = \sum_{i=1}^{N} \alpha^{(i)} k\left(\mathbf{x}^\star, \mathbf{x}^{(i)}\right) + b$ with the coefficients $\alpha^{(i)}$'s being obtained as the solution of maximizing a new form of the statistic

$$J(\alpha) = \frac{\alpha^T \mathbf{M} \alpha}{\alpha^T \mathbf{N} \alpha}.$$

Where $\mathbf{M} = \left(\mathbf{m}_0^{\mathcal{F}} - \mathbf{m}_1^{\mathcal{F}}\right)\left(\mathbf{m}_0^{\mathcal{F}} - \mathbf{m}_1^{\mathcal{F}}\right)^T$, $\mathbf{N} = \mathbf{KLK}$ and $\mathbf{m}_q^{\mathcal{F}} = N_q^{-1}\mathbf{K}\mathbf{y}_q$. Just as in FLD, in KFD the 'within scatter' matrix is not full rank. This implies that some form of regularisation will need to be applied when inverting $\mathbf{N}$ and this will generally be done by applying $\mathbf{N}_\delta = \mathbf{N} + \delta\mathbf{C}$, with $\mathbf{C}$ being the identity or the kernel matrices. Therefore the solution can be computed by either solving a generalised eigenproblem or by taking

$$\alpha_{KFD} \propto (\mathbf{N} + \delta\mathbf{C})^{-1}\left(\mathbf{m}_0^{\mathcal{F}} - \mathbf{m}_1^{\mathcal{F}}\right). \tag{24}$$

We are now in position to show the equivalence of KFD and our scheme, BFD.

Demonstration Disregarding the bias term, the projection of a new test point under KFD will be

$$\bar{f}^\star = \alpha_{KFD}^T \mathbf{k}. \tag{25}$$

Our claim is that Equation 21 is equivalent to Equation 25. In other words, that the projection of a new test point in KFD is equal to the mean of the predictive distribution for a test point under BFD. As in both equations the vector $\mathbf{k}$ is the same, we can write Equation 21 as

$$\bar{f}^\star = \alpha_{BFD}^T \mathbf{k},$$

with the vector

$$\alpha_{BFD} \propto d\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}} \tag{26}$$

and the constant of proportionality being given by the denominator of (21). Then our proof reduces to showing that the coefficients α_{KFD} and α_{BFD} are the same.

On one hand, we start by analysing KFD's main result which is given by Equation 24. From the definition of $\mathbf{m}_q^{\mathcal{F}}$, the difference $\left(\mathbf{m}_0^{\mathcal{F}} - \mathbf{m}_1^{\mathcal{F}}\right)$ can be written as $\mathbf{K}\Delta\hat{\mathbf{y}}$, with $\Delta\hat{\mathbf{y}} = \left(N_0^{-1}\mathbf{y}_0 - N_1^{-1}\mathbf{y}_1\right)$, and by regularising $\mathbf{N}$ with a multiple of the kernel matrix we obtain

$$\alpha_{KFD} \propto \left(\mathbf{KLK} + \beta^{-1}\mathbf{K}\right)^{-1}\mathbf{K}\Delta\hat{\mathbf{y}},$$

where β^{-1} is the regularisation coefficient.
On the other hand, substituting the value of $\mathbf{A}$ (Equation 23) into Equation 26, premultiplying by β and ignoring d we get

$$\alpha_{BFD} \propto \left(\mathbf{KLK} + \beta^{-1}\mathbf{K}\right)^{-1}\mathbf{K}\Delta\hat{\mathbf{y}},$$

which clearly is the regularised version of KFD that we were talking about.

As an additional insight, we observe that the coefficients α_{BFD} have an equivalent α_{KFD} if and only if KFD uses a regularisation based on a multiple of the kernel matrix. This equivalence is lost if the regulariser is based on the identity matrix.

5.2 Least Squares Support Vector Machines

A least squares support vector machine (LS-SVM) implements a two-norm cost function[3] and uses equality constraints instead of the inequalities present in the standard SVM (Vapnik, 1995). This greatly simplifies the way to obtain the solution as the resulting system of equations is linear. Unfortunately the sparseness which is characteristic of the SVM is lost. LS-SVM's have been related to ridge regression with modified targets, discriminant analysis in the feature space (KFD) and, as many other kernelised algorithms, to GP's.

Given a set of training data $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ with labels $y^{(i)} \in \{-1, 1\} \forall i$, the primal optimisation problem for an LS-SVM is expressed as

$$\begin{aligned} \min \quad & C = \frac{\mu}{2}\mathbf{w}^T\mathbf{w} + \frac{\zeta}{2}\sum_{n=1}^{N}\left(e^{(n)}\right)^2 \\ s.t. \quad & e^{(n)} = \left(y^{(n)} - \mathbf{w}^T\mathbf{x}^{(n)}\right) \quad \forall n, \end{aligned}$$

with μ and ζ being positive coefficients. This formulation in particular was given by Van Gestel et al. (2002) to elaborate the Bayesian framework of the LS-SVM. Such framework is nothing else but the recognition that the primal problem implements a regularised least squares cost function with regression to the labels. This cost function arises from the model depicted in Figure 4. In this figure, the joint distribution over labels and parameters factorises as $p(\mathbf{y}, \mathbf{w}|\mathbf{X}) = p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \times p(\mathbf{w})$, with noise model $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}\left(\mathbf{X}\mathbf{w}, \zeta^{-1}\mathbf{I}\right)$ and prior $p(\mathbf{w}) = \mathcal{N}\left(\mathbf{0}|\mu^{-1}\mathbf{I}\right)$.

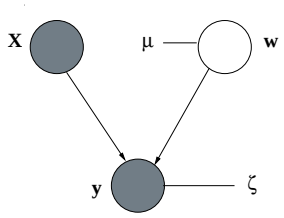


Figure 4: LS-SVM noise model assumes a regularised least squares cost function. The model depicted can be interpreted as the joint distribution $p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{X}, \mathbf{w})\, p(\mathbf{w})$, whereby the noise is Gaussian, $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}\left(\mathbf{X}\mathbf{w}, \zeta^{-1}\mathbf{I}\right)$, as is the prior $p(\mathbf{w}) = \mathcal{N}\left(\mathbf{0}|\mu^{-1}\mathbf{I}\right)$. In this model the targets and the labels are the same $\mathbf{t} \equiv \mathbf{y}$.

It is clear from the figure that LS-SVM employs a different noise model than BFD. In practice, the regression to the labels assumption can work well. However, it suffers from the fundamental missconception that the class labels ± 1 have to coincide with the projected class centres c_q. The main difference with our algorithm is that the LS-SVM assumes that targets and labels are the same, $\mathbf{t} \equiv \mathbf{y}$, but we do not.

Van Gestel et al. (2002) were aware of this limitation[4] and relaxed the assumption $\mathbf{t} \equiv \mathbf{y}$ by modelling the distribution $p\left(\hat{\mathbf{t}}_q \middle| \mathbf{X}, \mathbf{w}\right)$ by application of Bayes' rule. In other words, they computed $p\left(\hat{\mathbf{t}}_q \middle| \mathbf{X}, \mathbf{w}\right) \propto p\left(\mathbf{X}|\hat{\mathbf{t}}_q, \mathbf{w}\right) p\left(\hat{\mathbf{t}}_q\right)$. This is in marked contrast with the strategy adopted in this paper. As is shown by Equation 12, in BFD we model directly the distribution $p\left(\hat{\mathbf{t}}_q \middle| \mathbf{X}, \mathbf{y}, \mathbf{w}\right)$. Hence it can

3. This is instead of the traditional ℓ_1.
4. See Section 3.2 of their paper.

be seen that in our approach $\mathbf{y}$ is used as a conditioning value whereas in Van Gestel's paper it is not.

5.2.1 PROXIMAL SUPPORT VECTOR MACHINES

Another related approach is known as the proximal support vector machine or P-SVM, proposed by Fung and Mangasarian (2001). A P-SVM is very close to LS-SVM in the sense that both of them consider equality constraints and implement regularised least squares cost functions. However, P-SVM's have been interpreted from the point of view of classifying points by clustering data around two parallel hyperplanes; whereas LS-SVM's have been interpreted from the more classical point of view of maximising the margin around a single hyperplane. P-SVM's have also been approached from a probabilistic point of view by Agarwal (2002). Indeed, by following Agarwal's work it is possible to see that they also implement the graphical model depicted in Figure 4, except for a few changes in parameters. Ignoring the bias term, in P-SVM's the joint distribution $p(\mathbf{y}, \mathbf{w}|\mathbf{X})$ is factorised according to the noise model $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}\left(\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}\right)$ and the prior distribution $p(\mathbf{w}) = \mathcal{N}\left(\mathbf{0}, \nu\sigma^2\mathbf{I}\right)$. The parameter σ^2 is the variance of the residuals[5] while ν is known as ridge parameter. In many applications, such as data mining, the ridge parameter is chosen by cross-validation. It is clear that this task becomes unfeasible if the ridge parameter is taken to the extreme of considering one parameter for every 'predictor', in other words, if we take as rigde parameter a matrix of the form $diag(\nu_1, \ldots, \nu_d)$.

In (Agarwal, 2002) the problem of tuning the ridge parameter is addressed by studying its effects on ridge regression. This can be observed by writing up the regularised P-SVM cost function

$$C_{PSVM} = \frac{1}{\sigma^2}\left[(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{1}{\nu}\mathbf{w}^T\mathbf{w}\right].$$

Whenever ν becomes small, the ridge part takes over, but if it becomes large the 'noise' part will dominate. Nevertheless, it is clear that BFD implements a different type of noise model when compared to LS-SVM's and P-SVM's.

6. Connections with the Generalisation Error

In Section 3.1.1 we saw that optimisation of the proposed noise model and that of Rayleigh's coefficient give equivalent results. In both cases the solution to the discriminant problem was given by adjusting the level of β. In order to understand better the physical significance that this represents, it is useful to analyse the problem from the point of view of classification of two populations. Specifically, during this section we will refer to the plot in Figure 5 and always assume that both classes have the same cost of misclassification.

In Figure 5, it can be observed that both mapping distributions share the same precision. Under this assumption, for fixed d, we can see that the generalisation error will decrease as β increases, *i.e.* as $\beta^{-1/2}$ decreases. From the point of view of projected data, the problem has shifted from computing the direction of discrimination to that of minimising the generalisation error through the adjustment of the variable β.

The likelihood function $L(\mathbf{f})$ defined in Equation 11 allows us to think of β as an extra random variable. Hence placing a prior over it not only places a prior over the generalisation error but on

5. The residuals are defined as $e^{(n)} = y^{(n)} - \mathbf{w}^T\mathbf{x}^{(n)}, \forall n$.

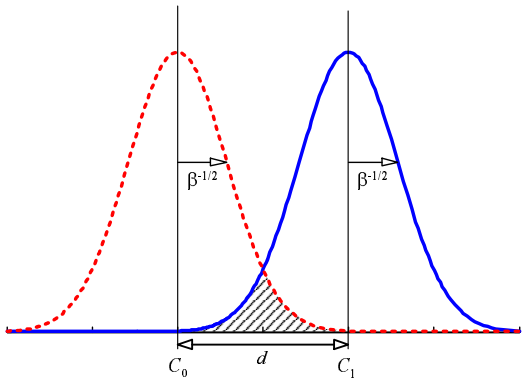


Figure 5: Generalisation error as it relates to β and d. The shaded area gives the generalisation error if the true densities conform to those given by two Gaussians with equal precision β. The class centres have been denoted by c_q with $q \in \{1,0\}$.

Rayleigh's coefficient as well. Consider, for example, the case where $d = 2$ and the class priors are equal: if the data does truly map to the mixture distribution, then the generalisation error will be

$$E_{\text{eq}} = \frac{1}{2} - \frac{1}{2}\text{erf}\left(\sqrt{\frac{\beta}{2}}\right).$$

Let Equation 11 be a 'likelihood function', then by considering a gamma distribution $\mathcal{G}(\beta|a,b)$ as a prior,

$$p(\beta) = \frac{b^a}{\Gamma(a)}\beta^{a-1}\exp(-b\beta),$$

the MAP solution for β will be (see Appendix D)

$$\hat{\beta}_{\text{MAP}} = \frac{N + 2a - 2}{\sigma_1^2 + \sigma_0^2 + 2b}. \tag{27}$$

By setting $a = b = 0.5$ we indirectly obtain a uniform distribution over E_{eq}, which is also a chi-square distribution with one degree of freedom. This special case leads to a new expression of the form

$$\hat{\beta}_{\text{MAP}} = \frac{N-1}{\sigma_1^2 + \sigma_0^2 + 1}, \tag{28}$$

which can be viewed as a regularised version of Equation 8. The prior could also be used to bias β towards low or high generalisation errors if this is thought appropriate.

From the discussion of Section 5.1, taking the limit as $\beta \to \infty$ leads to the standard kernel Fisher's discriminant. From Figure 5 it can be seen that an *a priori* setting of β^{-1} to zero is equivalent to assuming that we can achieve a generalisation error of zero.

OTHER SPECIAL CASES

Taking the limit as $\beta \rightarrow 0$ causes the mean prediction for $f^\star$ and its variance to take on a much simpler form,

$$\bar{f}^\star = \alpha_\beta^T \mathbf{k}$$

where

$$\alpha_\beta = \frac{d\Delta\hat{\mathbf{y}}}{\Delta\hat{\mathbf{y}}^T \mathbf{K} \Delta\hat{\mathbf{y}}},$$

and

$$(\sigma^\star)^2 = k_\star - \mathbf{k}^T \frac{\Delta\hat{\mathbf{y}}^T \Delta\hat{\mathbf{y}}}{\Delta\hat{\mathbf{y}}^T \mathbf{K} \Delta\hat{\mathbf{y}}} \mathbf{k}.$$

This result is remarkable for the absence of any requirement to invert the kernel matrix, which greatly reduces the computational requirements of this algorithm. In fact, driving β to zero leads to the well known Parzen windows classifier, sometimes known as probabilistic neural network, (Duda and Hart, 1973). See the work of Schölkopf and Smola (2002) or Roth (2005) for some related studies in limiting cases.

7. Optimising Kernel Parameters

One key advantage of our formulation is that it leads to a principled approach for determining all the model parameters. In the Bayesian formalism it is quite common to make use of the marginal likelihood to reach this purpose, therefore we look to optimise

$$\mathcal{L}(\Theta_t) = \log p(\mathbf{t}|\mathcal{D}, \Theta_t),$$

with respect to the model parameters Θ_t. Recall in Section 3.1 that we optimised the likelihood with respect to the parameters c_0 and c_1 leading to a new encoding of the targets

$$\hat{\mathbf{t}}_q = \left(\frac{\mathbf{f}^T \mathbf{y}_q}{N_q}\right) \mathbf{y}_q.$$

We back substituted these values into the likelihood in order to demonstrate the equivalence with maximisation of Rayleigh's coefficient. Unfortunately, one side effect of this process is that it makes the new targets $\hat{\mathbf{t}}$ dependent on the inputs. As a consequence, the targets will shift whenever the kernel parameters are changed. As expressed in Section 3.1.1, one solution could be to iterate between determining $\mathbf{t}_0$, $\mathbf{t}_1$ and optimising the rest of the parameters. This approach is simple, but it may be difficult to prove convergence properties. We therefore prefer to rely on an expectation-maximisation (EM) algorithm (Dempster et al., 1977) which finesses this issue and for which convergence is proved.

7.1 EM Algorithm

We denote the parameters of the prior as Θ_k and the complete set of model parameters as $\Theta_t = \{\Theta_k, \beta\}$. Then the goal is to solve the problem $\arg\max_{\Theta_t} \log p\left(\hat{\mathbf{t}} \middle| \mathbf{X}, \Theta_t\right)$, where we have made use again of the modified targets $\hat{\mathbf{t}}$. In order to solve the problem, a variational lower bound on the marginal log-likelihood is imposed

$$\mathcal{L}(\Theta_t) \geq \int q(\mathbf{f}) \log \frac{p\left(\hat{\mathbf{t}} \middle| \mathbf{y}, \mathbf{f}, \beta\right) p(\mathbf{f}|\mathbf{X}, \Theta_k)}{q(\mathbf{f})} d\mathbf{f}, \tag{29}$$

where $q(\mathbf{f})$ is a distribution over the latent variables that is independent on the current value Θ_t. EM consists of the alternation of the maximisation of $\mathcal{L}$ with respect to $q(\mathbf{f})$ and Θ_t, respectively, by holding the other fixed. This procedure repeated iteratively guarantees a local maxima for the marginal likelihood will be found. Thus our algorithm will be composed of the alternation of the following steps:

E-step Given the current parameters Θ_t^{it}, approximate the posterior with

$$q^{it}(\mathbf{f}) \propto \exp\left(-\frac{1}{2}\mathbf{f}^T\Sigma_p^{-1}\mathbf{f}\right),$$

where

$$\Sigma_p = \left(\mathbf{K}^{-1} + \beta\mathbf{L}\right)^{-1}. \tag{30}$$

M-step Fix $q^{it}(\mathbf{f})$ to its current value and make the update

$$\Theta_t^{it+1} = \arg\max_{\Theta_t} \mathcal{L}, \tag{31}$$

where the evidence is computed as $\mathcal{L} = \left\langle \log p\left(\hat{\mathbf{t}}|\,\mathbf{y},\mathbf{f},\beta\right) p(\mathbf{f}|\mathbf{X},\Theta_k)\right\rangle_{q(\mathbf{f})}$. We have used the notation $\langle\cdot\rangle_{p(x)}$ to indicate an expectation under the distribution $p(x)$.

Maximisation with respect to Θ_k, the kernel parameters, cannot be done in closed form and has to rely on some optimisation routine, for example gradient descent, therefore it is necessary to specify the gradients of Equation 29 w.r.t. Θ_k. An update for β can be worked out quite easily because the maximisation of $\mathcal{L}$ with respect to this parameter has a closed form solution. The expression obtained is of the form

$$\hat{\beta}^{it} = \frac{N}{\overline{\sigma}_1^2 + \overline{\sigma}_0^2},$$

where $\overline{\sigma}_1^2 = \sum y_n \left\langle (f_n - \mu_1)^2 \right\rangle$ and the expectation $\langle\cdot\rangle$ is computed under the predictive distribution for the nth training point, see Equation 21. An expression for $\overline{\sigma}_0^2$ is given in a similar way.

7.2 Updating β

Following our discussion in Section 6, we propose (and in fact used) Equation 28 to update the value of β at every iteration. We repeat the expression here

$$\hat{\beta}_{\text{MAP}}^{it} = \frac{N-1}{\overline{\sigma}_1^2 + \overline{\sigma}_0^2 + 1}. \tag{32}$$

The resulting optimisation framework is outlined in Algorithm 1.

8. Experiments

In this section we report the results of experiments that we carried out to test our algorithmic approach. A first batch of experiments was carried out on classification of synthetic data with the purpose of understanding better the behaviour of the algorithm, and in order to test it more realistically, a second batch of experiments was carried out on benchmark data.

Algorithm 1 A possible ordering of the updates.

Select Convergence tolerances η_β and η_{Θ_k}.
Set Initial values $\Theta_k^{(0)}$ and $\hat{\beta}^{(0)}$.
Require Data-set $\mathcal{D} = (\mathbf{X}, \mathbf{y})$.
while change in $\hat{\beta}^{(it)} < \eta_\beta$ and change in $\Theta_k^{(it)} < \eta_{\Theta_k}$ **do**

- Compute kernel matrix $\mathbf{K}$ using $\Theta_k^{(it)}$.
- Update Σ_p with Equation 30
- Use scale conjugate gradients to maximise $\mathcal{L}$ with respect to $\Theta_k^{(it)}$. Apply Equation 31
- Update $\hat{\beta}^{(it)}$, use Equation 32.

end

8.1 Toy Data

As a first experiment, we compared the KFD, LS-SVM and BFD algorithms on four synthetic data sets using an RBF kernel. Additionally, as a second experiment, we used BFD with an ARD prior on the same data sets to observe some of the capabilities of our approach. In order to facilitate further reference, each data set will be named according to its characteristics. Firstly, **Spiral**[6] can only be separated by highly non-linear decision boundaries. **Overlap** comes from two Gaussian distributions with equal covariance, and is expected to be separated by a linear plane. **Bumpy** comes from two Gaussians but by being rotated at 90 degrees, quadratic boundaries are called for. Finally, **Relevance** is a case where only one dimension of the data is relevant to separate the data.

We hypothesized BFD would perform better than LS-SVM and KFD in all the cases because it models directly the class conditional densities. In order to compare the three approaches, we trained KFD, LS-SVM and BFD classifiers with a standard RBF kernel, as specified in Appendix E. Model parameters for KFD and LS-SVM were selected by 10-fold cross-validation whereas BFD was trained by maximising the evidence, using Algorithm 1.

In Figure 6 we present a comparison of the three algorithms. We can observe a similar performance in the case of **Spiral**; however it is encouraging to observe that BFD gives more accurate results in the rest of the cases. Despite not producing a straight line, KFD and BFD give accurate results in **Overlap**, whereas LS-SVM overfits. If none of the algorithms separates this data set with a line it is because obtaining a linear boundary from an RBF kernel is extremely difficult (see Gramacy and Lee, 2005). In **Bumpy**, the three algorithms give arguably the same solution, with BFD having the smoothest boundary. Lastly, in **Relevance** all the algorithms provide accurate results, with BFD giving the smoothest solution. In all these experiments we set the initial $\Theta_t = \mathbf{1}$ for BFD and furthermore, observed that BFD did not present any initialisation problems. In all our simulations, we let the algorithm stop whenever $\eta_\beta < 1 \times 10^{-6}$ or the change in $\eta_{\Theta_k} < 1 \times 10^{-6}$.

As a second experiment, we were interested in training BFD to test the different facets of the following kernel

$$k\left(\mathbf{x}^i, \mathbf{x}^j\right) = \theta_1 \exp\left(-\frac{\theta_2}{2}\left(\mathbf{x}^i - \mathbf{x}^j\right)^T \Theta_{ard}\left(\mathbf{x}^i - \mathbf{x}^j\right)\right) + \theta_3\left(\mathbf{x}^i\right)^T \Theta_{ard}\mathbf{x}^j + \theta_4 + \theta_5\delta_{ij}, \tag{33}$$

6. This was first used by Lang and Witbrock (1988).

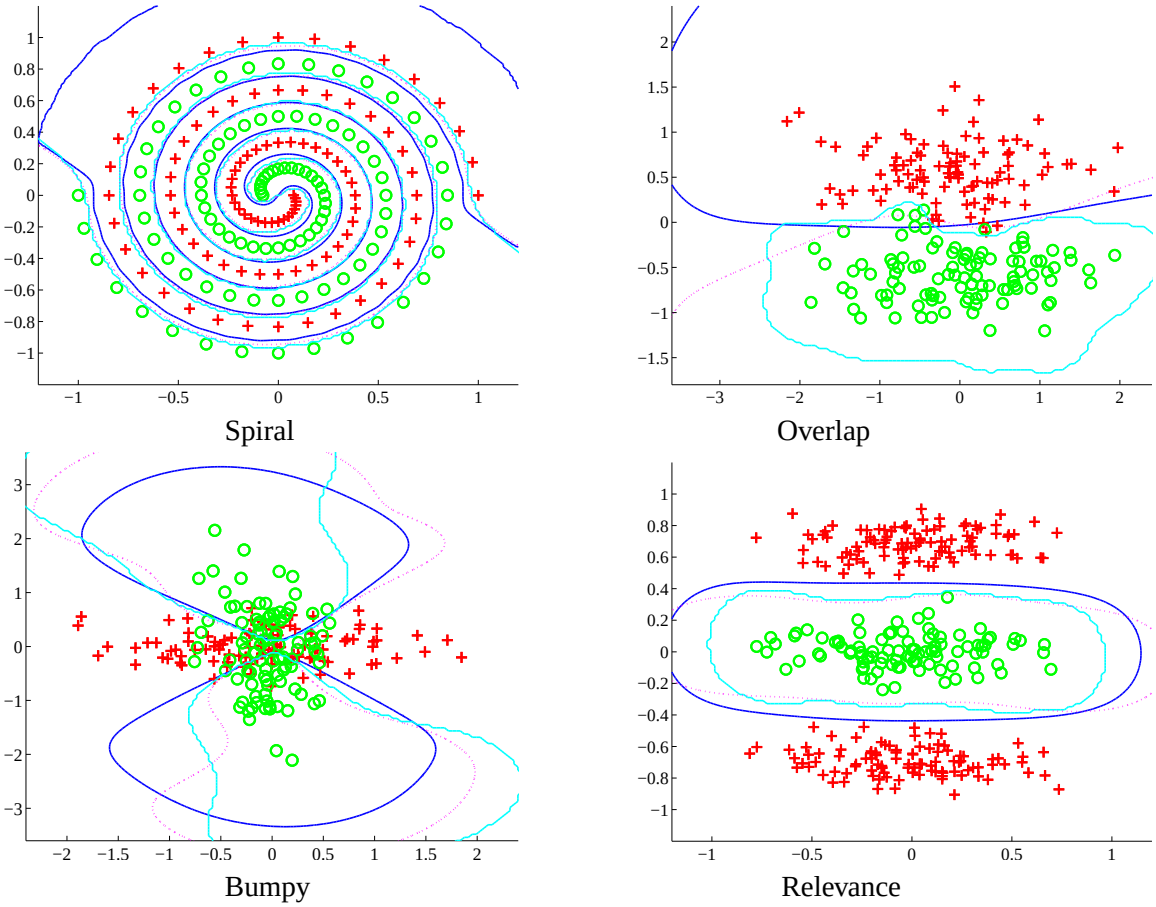


Figure 6: Comparison of classification of synthetic data sets using an RBF kernel. Two classes are shown as pluses and circles. The separating lines were obtained by projecting test data over a grid. The lines in blue (dark), magenta (dashed) and cyan (gray) were obtained with BFD, KFD and LS-SVM respectively. Kernel and regularisation parameters for KFD and LS-SVM were obtained by 10-fold cross validation, whereas BFD related parameters were obtained by evidence maximisation. We trained BFD using Algorithm 1; details of our implementations are given in Appendix E.

where δ_{ij} is the Kronecker delta and the matrix $\Theta_{ard} = diag(\theta_6, \ldots, \theta_{6+d-1})$ with d being the dimension of $\mathbf{X}$. This kernel has four components: an RBF part composed of $(\theta_1, \theta_2, \Theta_{ard})$; a linear part, composed of (θ_3, Θ_{ard}); a bias term given by θ_4 and the so-called 'nugget' term θ_5 which, for a large enough value θ_5, ensures that $\mathbf{K}$ is positive definite and therefore invertible at all times. Therefore, the parameters of the model are $\Theta_t = (\Theta_k, \beta)$, with $\Theta_k = (\theta_1, \ldots, \theta_{6+d-1})$.

On this occassion, BFD got stuck into local minima so we resorted to do model selection to choose the best solution. This process was carried out by training each data set with three different initial values for θ_2 while the remaining $\theta_{i\neq 2}$ were always initialised to 1. In the cases of **Bumpy** and **Relevance** we made the initial $\theta_2 = \left[10^{-2}, 10^{-1}, 1\right]$, for **Spiral** we made it equal to $[1, 10, 100]$ and for **Overlap**, $\left[1.5 \times 10^{-2}, 10^{-1}, 1\right]$. From the resulting solutions (three per data set), we selected the model that produced the highest marginal likelihood $\mathcal{L}$. In all our simulations, we let the algorithm

stop whenever $\eta_\beta < 1 \times 10^{-6}$ or the change in $\eta_{\Theta_k} < 1 \times 10^{-6}$. The parameter β was always initialised to 1. The selected models for each set are summarised in Figure 7.

The results are promising. In **Spiral**, the separating plane is highly non-linear as expected. Meanwhile, we observe in **Overlap** that the predominating decision boundary in the solution is linear. In **Bumpy**, the boundary starts to resemble a quadratic and, finally, for **Relevance**, only one dimension of the data is used to classify the data. Note that the values for Θ_k, summarised in Table 1, go in accordance with these observations. For example, in **Overlap** and **Relevance**, the value of θ_6 is significantly lower than θ_7, indicating that only one dimension of the data is relevant for the solution. This is markedly different to the cases of **Spiral** and **Bumpy**, where both dimensions (θ_6 and θ_7) have been given relatively the same weights. Hence, for every case we have obtained sensible solutions. All the kernel parameters determined by the algorithm, for the four experiments, are given in Table 1.

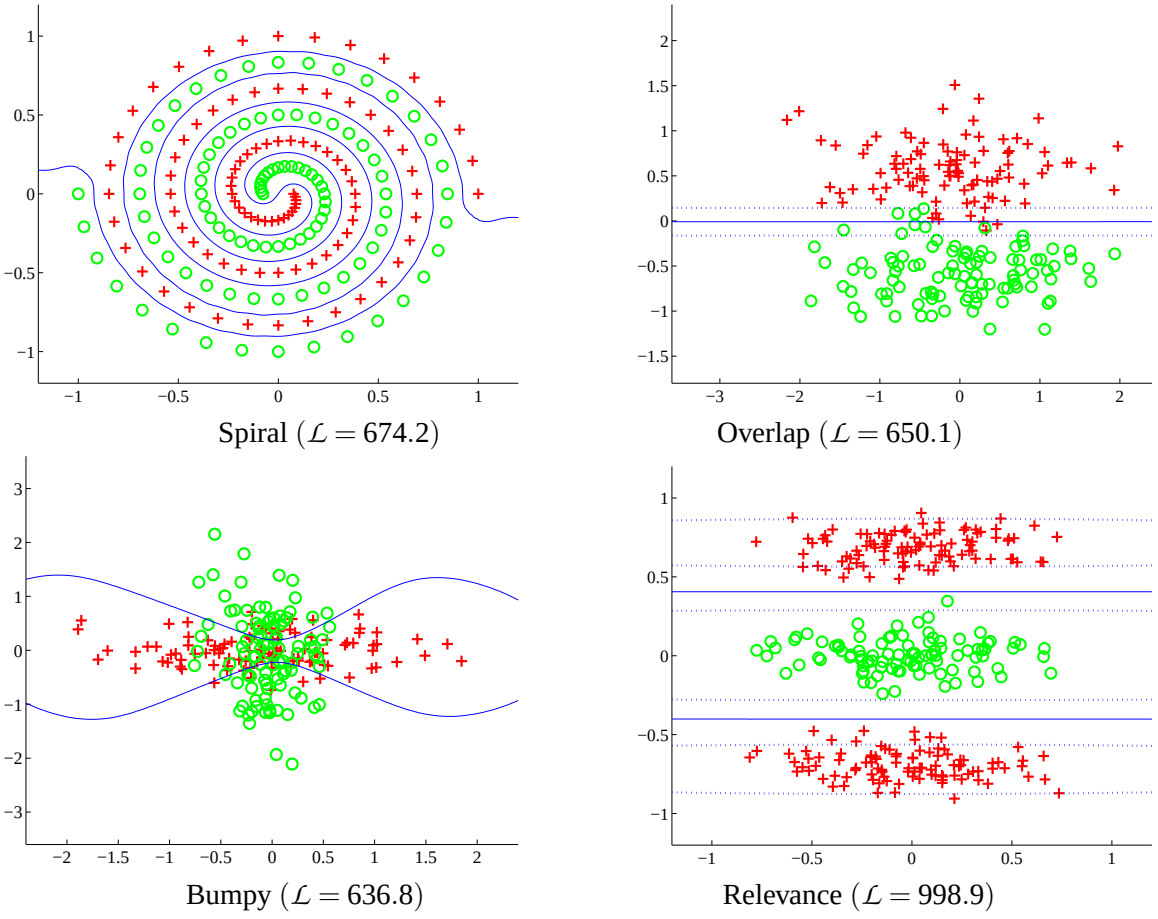


Figure 7: Classification results on toy data sets using an ARD prior. Two classes are shown as pluses and circles. The decision boundary is given by the solid line. Dotted lines indicate points at $1/4$ of the distance (as measured in the projected space) from the decision boundary to the class mean. Log-likelihood values appear enclosed by brackets.

Figure 8 shows an example of the result of training **Spiral** with a poor initialisation. It can be seen that the value of the marginal likelihood in this case is smaller to the one presented in Figure 7. However, this behaviour is not exclusive of BFD, indeed we observed a very similar situation with a poorly initialised Bayesian LS-SVM and with KFD cross-validated with a badly selected grid.

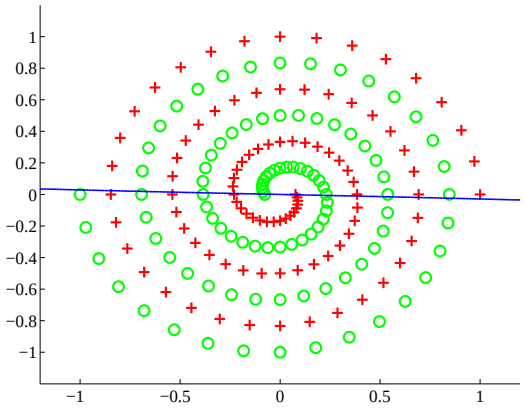


Figure 8: The solution for the spiral data with a poor initialisation $\theta_2 = 1$. Associated log-likelihood $\mathcal{L} = 562.7$.

Experiment	$\ln\theta_1$	$\ln\theta_2$	$\ln\theta_3$	$\ln\theta_4$	$\ln\theta_5$	$\ln\theta_6$	$\ln\theta_7$
Spiral	8.5015	−9.5588	1.0139	−4.9759	−10.6373	−2.78	−2.9609
Overlap	0.5011	−7.9801	1.1455	−4.8319	−8.5990	−6.9953	−0.1026
Bumpy	4.9836	−10.8222	1.1660	−4.7495	−13.5996	−3.9131	−3.7030
Relevance	4.6004	−9.5036	1.2734	−4.9351	−13.8155	−6.9968	−1.5386

Table 1: *log*-values of the parameters learnt with BFD for the different toy experiments. In **Overlap** and **Relevance**, the weights of the feature θ_6 are low if compared with the feature θ_7. This is in contrast with **Spiral** and **Bumpy**, where both features have been given relatively the same weights.

8.2 Benchmark Data Sets

In order to evaluate the performance of our approach, we tested five different algorithms on well known problems. The algorithms used were: linear and quadratic discriminants (LDA and QDA), KFD, LS-SVM and BFD. The last two algorithms provided the opportunity to use ARD priors so they were reported as well. We used a synthetic set (**banana**) along with 12 other real world data sets coming from the **UCI**, **DELVE** and **STATLOG** repositories.[7] In particular, we used instances of these data that had been preprocessed an organised by Rätsch et al. (1998) to do binary classification tests. The main difference between the original data and Rätsch's is that he converted every problem

7. The **breast cancer** domain was obtained from the University Medical Center, Institute of Oncology, Ljubljana, Yugoslavia. Thanks to M. Zwitter and M. Soklic for the data.

into binary classes and randomly partitioned every data set into 100 training and testing instances.[8] In addition, every instance was normalised to have zero mean and unit standard deviation. More details can be found at (Rätsch et al., 1998).

Mika et al. (1999) and Van Gestel et al. (2002) have given two of the most in depth comparisons of algorithms related to FLD. Unfortunately, the reported performance in both cases is given in terms of test-set accuracy (or error rates), which implied not only the adjustment of the bias term but also the implicit assumption that the misclassification costs of each class were known. Given that discriminant methods operate independently of the method of bias choice, we felt it more appropriate to use a bias independent measure like the area under the ROC curve (AUC).

The LDA and QDA classifiers were provided by the Matlab function `classify` with the options 'linear' and 'quadratic', respectively. In both cases, no training phase was required, as described by Michie et al. (1994). The output probabilities were used as latent values to trace the curves. Meanwhile, for KFD's parameter selection we made use of the parameters obtained previously by Mika et al. (1999) and which are available at `http://mlg.anu.edu.au/~raetsch`. The ROC curves for KFD were thus generated by projecting every instance of the test set over the direction of discrimination.

Mika trained a KFD on the first five training partitions of a given data set and selected the model parameters to be the median over those five estimates. A detailed explanation of the experimental setup for KFD and related approaches can be found in Rätsch et al. (1998) and Mika et al. (1999). In the case of LS-SVM, we tried to follow a similar process to estimate the parameters, hence we trained LS-SVM's on the first five realisations of the training data and then selected the median of the resulting parameters as estimates. In the same way, projections of test data were used to generate the ROC curves.

Finally, for BFD we also tried to follow the same procedure. We trained a BFD model with $N_x = 8$ different initialisations over the first five training instances of each data set. Hence we obtained an array of parameters of dimensions 8×5 where the rows were the initialisations, the columns were the partitions and each element a vector Θ_t. For each column, we selected the results that gave the highest marginal likelihood, so that the array reduced from 40 to only 5 elements. Then we followed the KFD procedure of selecting the median over those parameters. In these experiments, we used the tolerances η_β and η_{Θ_k} to be less than 1×10^{-6}. More details of the experimental setup are given in Appendix E.

In Table 2 we report the averages of the AUC's over all the testing instances of a given data set. In the cases of KFD, LS-SVM and BFD we used the RBF kernel of Appendix E. Computation of the ROC curves were done with the function `ROC` provided by Pelckmans et al. (2003) and Suykens et al. (2002) and no further processing of the curves was required, for instance removing convexities was unnecessary.

It can be observed that BFD outperforms all the other methods in $6/13$ data sets, comes second in 3 cases and third in the remaining 4. In particular, it is remarkable to see BFD performing consistently better than KFD across most of the problem domains. It seems that leaving the 'regression to the labels' assumption pays-off in terms of areas under the ROC curves. It is also interesting to observe that LDA performs well in almost all the problems (except **banana**) and it thus indicates that most of these data sets could be separated with a linear hyperplane with acceptable results. From these results we can conclude that the better designed noise model in BFD allows it to outperform 'similar'

8. Data sets can be obtained from `http://mlg.anu.edu.au/~raetsch`.

state of the art approaches. The P-SVM was not included in the experiments because it is a 'type-of' LS-SVM.

RBF	Banana	Breast	Diabetis	German	Heart	Image
LDA	53.7 (1.3)	71.2 (5.2)	**82.7** (**1.6**)	78.5 (2.5)	90.1 (2.6)	87.9 (0.7)
QDA	64.7 (2.5)	70.8 (5.5)	80.3 (2.0)	76.5 (2.6)	86.9 (3.1)	91.2 (1.5)
KFD	**96.1** (**0.4**)	70.9 (5.8)	76.7 (2.3)	69.9 (4.7)	88.8 (3.0)	**99.5** (**0.1**)
LS-SVM	95.5 (0.4)	61.1 (5.2)	73.7 (2.3)	74.0 (2.8)	89.9 (2.8)	98.8 (0.3)
BFD	95.1 (0.6)	**73.4** (**5.3**)	81.1 (1.9)	**79.0** (**2.5**)	**90.9** (**2.7**)	98.2 (0.4)

RBF	Ringnorm	Flare S.	Splice	Thyroid	Titanic	Twonorm	Waveform
LDA	80.0 (0.8)	**73.9** (**1.9**)	91.8 (0.4)	86.6 (5.8)	70.8 (1.0)	99.7 (0.0)	**92.5** (**0.7**)
QDA	99.8 (0.0)	61.6 (1.8)	**93.0** (**0.4**)	97.5 (1.7)	71.4 (2.0)	99.5 (0.0)	91.2 (0.4)
KFD	99.8 (0.0)	65.6 (2.5)	91.3 (0.5)	97.4 (3.6)	70.9 (1.0)	**99.8** (**0.0**)	88.6 (0.5)
LS-SVM	99.6 (0.1)	73.8 (1.6)	88.2 (0.7)	97.8 (1.4)	**73.8** (**2.4**)	93.8 (0.8)	83.3 (1.2)
BFD	**99.9** (**0.0**)	72.9 (2.0)	91.6 (0.5)	**98.5** (**1.1**)	71.6 (0.6)	**99.8** (**0.0**)	91.6 (0.8)

Table 2: Average classification results of benchmark data. We report mean and standard deviations (within brackets) of the AUC over all testing instances. The compared algorithms are: linear discriminant (LDA), quadratic discriminant (QDA), kernel Fisher's discriminant (KFD), least squares support vector machine (LS-SVM) and Bayesian Fisher's discriminant (BFD). In all the experiments an RBF kernel was used. It can be observed that BFD performs better in 6 out of 13 problem domains.

The BFD framework allows for the inclusion of some type of ARD priors. Incorporation of this type of prior performs feature selection by assigning very high weights to some of the posterior values of the hyperparameters and hence prunning out features, (see Mackay, 1995). We were interested in comparing our approach with the Bayesian version of the LS-SVM, which can also make use of ARD priors. Our results are presented in Table 3. In this case, however, the comparison is tighter with LS-SVM performing narrowly better than BFD in 7 out of the 13 problems. The EM algorithm we proposed is slower to converge than direct optimisation of the marginal likelihood as can be applied to the LS-SVM. Our use of the EM algorithm is necessary due to the nature of the moving targets, this is a disadvantage of our approach. Hence to obtain a solution in a reasonable time, we were obliged to reduce the number of initialisations to $N_x = 3$ and to increase the tolerances η_β and η_{Θ_k} to be less than 1×10^{-5} and 1×10^{-6}, respectively.

In Figure 9 we show a comparison of the weights assigned to each feature in two data sets, Ringnorm and Splice. We were interested on showing if there was any correlation on the degree of importance assigned to each feature by the two algorithms. Ringnorm and Splice were specially selected because they were examples in which BFD performed better (marginally) and worse than LS-SVM, respectively. In the figure, we report the values of the inverse weights, Θ_{ard}, for BFD while for LS-SVM we report the 'ranking' coefficients produced by the LS-SVM implementation we used (see Appendix E). These coefficients are related to the values Θ_{ard}. For identical results we expected to observe a high degree of correlation between BFD weights and LS-SVM rankings. As a first example, in Figure 9 we observe that Ringnorm is assigned varied values by BFD and LS-SVM. In fact, for features $[3-6]$ and 18, there is a reasonable degree of overlap between assigned values, and this could help to explain why both algorithms performed similarly well. This observation goes

in accordance with our intuition. It is noticeable that none of the features in BFD has been driven to zero, which indicates that this algorithm required all of the features to learn a solution. The second case corresponds to Splice. In this data set, LS-SVM performed better than BFD by a wide margin and this could well be explained by the aggressive pruning of features that BFD performed; as it is shown in the figure.

ARD	Banana	Breast	Diabetis	German	Heart	Image
LS-SVM	91.6 (1.0)	72.3 (5.4)	**83.3** (**1.7**)	**79.5** (**2.5**)	90.5 (2.6)	**98.9** (**0.6**)
BFD	**95.1** (0.6)	**74.2** (**5.1**)	81.1 (1.6)	77.9 (2.6)	**90.9** (**2.7**)	76.8 (0.9)

ARD	Ringnorm	Flare S.	Splice	Thyroid	Titanic	Twonorm	Waveform
LS-SVM	99.8 (0.0)	66.0 (3.3)	**95.7** (**0.3**)	**99.5** (**0.5**)	**73.6** (**2.6**)	99.6 (0.0)	**96.4** (**0.2**)
BFD	**99.9** (**0.0**)	**73.2** (**1.7**)	88.8 (0.5)	98.9 (0.8)	71.9 (1.1)	**99.7** (**0.0**)	94.0 (0.1)

Table 3: Average classification results of benchmark data. We report mean and standard deviations (within brackets) of the AUC over all testing instances. This table compares the BFD algorithm against an LS-SVM, both employing ARD based kernels. In this case Bayesian LS-SVM outperforms BFD in 7 out of 13 cases.

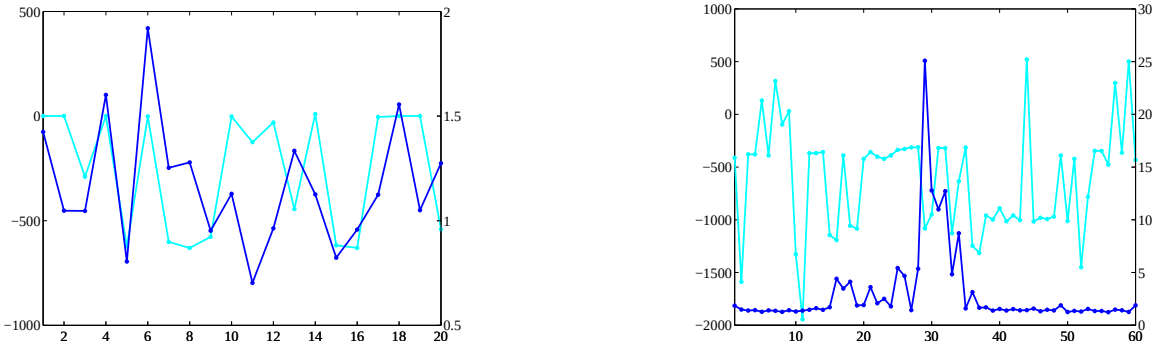


Figure 9: Plot of the feature values learned by BFD and LS-SVM on Ringnorm and Splice data sets. For both algorithms we used an ARD kernel. The weights learned in LS-SVM are plotted in cyan (gray) and correspond to the left *y*-axis, whereas the weights learned in BFD are plotted in blue (dark) and correspond to the right *y*-axis. We report weight values for BFD while for LS-SVM, 'ranking' coefficients. These coefficients are related to the weights Θ_{ard}. Ideally we would expect to see a perfect match between feature values of BFD and rankings in LS-SVM. As it is shown in Table 3, BFD performed better in Ringnorm while LS-SVM did better in Splice.

8.2.1 HISTOGRAMS OF PROJECTED DATA

A good way to visualize whether an FLD-based algorithm is performing adequately consists of generating histograms of projected data. We briefly compare the output distributions generated by BFD and KFD on training and test instances of the Waveform and Twonorm data sets. We used the data produced from the experiments with an RBF kernel to generate Figures 10 and 11.

In the Figure 10, BFD produced very consistent outputs between training and test sets for both Twonorm and Waveform. In particular, it is encouraging to see that Twonorm projects very close to two Gaussian distributions because this data set comes from two multivariate Gaussians. Meanwhile, in Figure 11, KFD produced very consistent output distributions in Twonorm but failed to do so in Waveform. Following similar arguments to those of Mika (2002), we believe this is one of the reasons for BFD performing better than KFD, in terms of AUC.

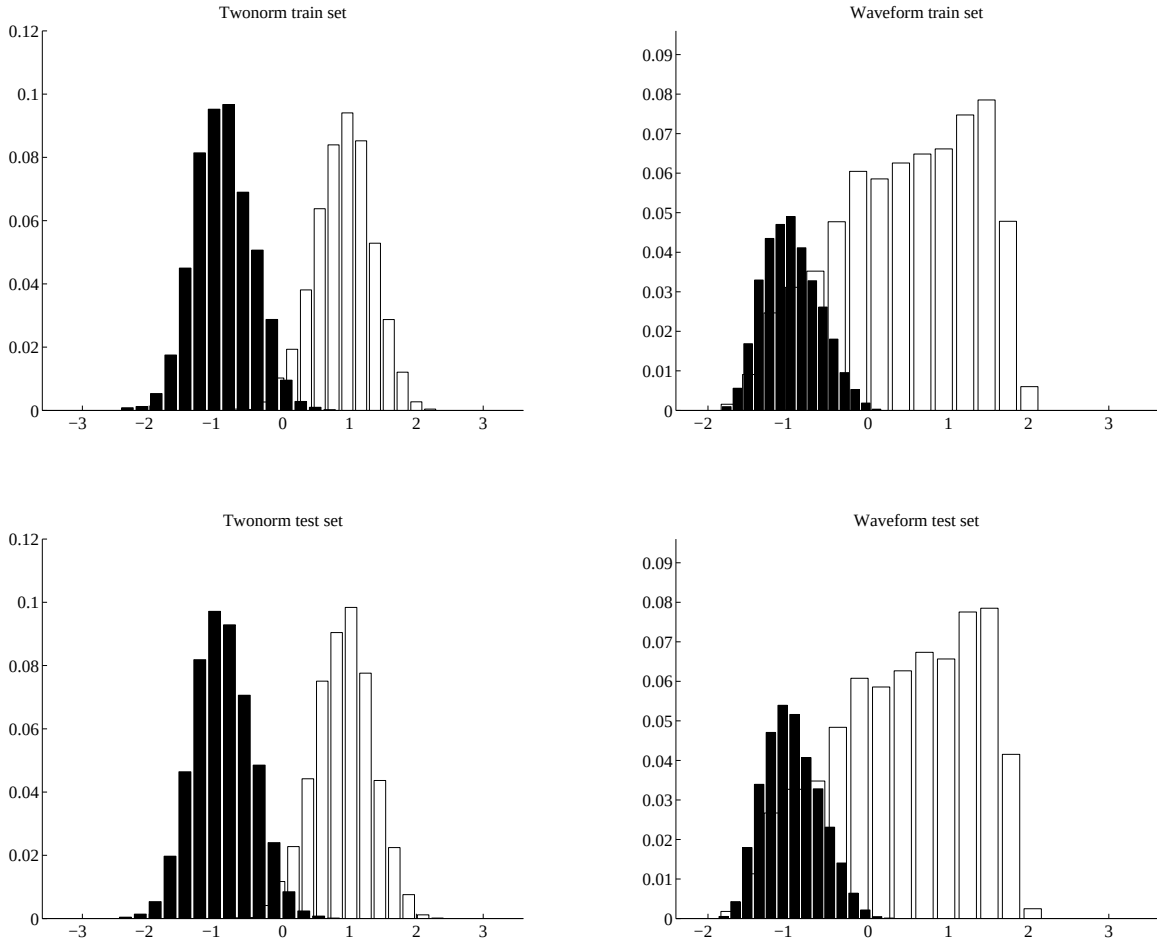


Figure 10: Comparison of output distributions on training and test sets for BFD. The data sets depicted are Twonorm and Waveform, respectively. It is clearly observable that training and test set distributions for BFD are quite consistent.

9. Conclusions and Future Work

We have presented a Bayesian probabilistic approach to discriminant analysis that can correspond to kernel Fisher's discriminant. Regularisation of the discriminant arises naturally in the proposed framework and through maximisation of the marginal likelihood we were able to determine kernel

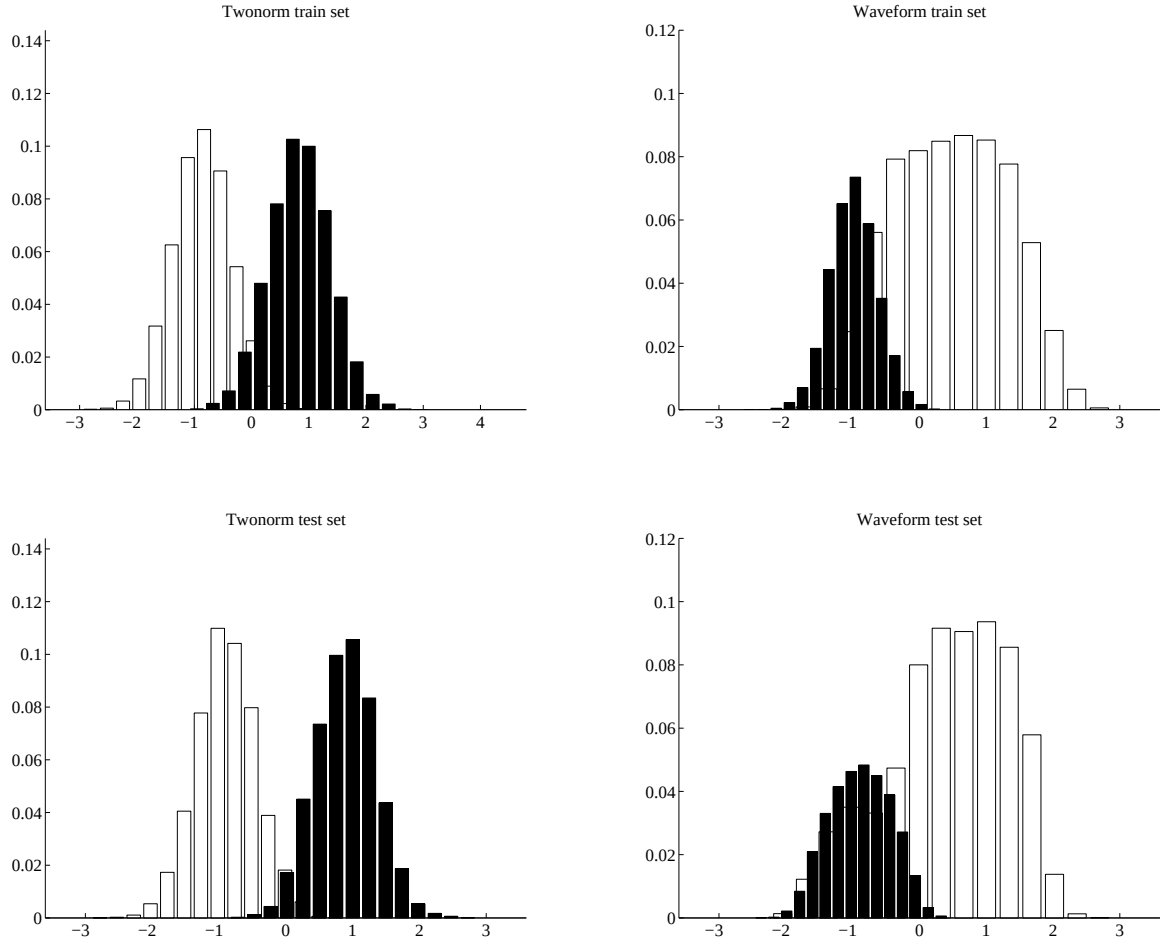


Figure 11: Comparison of output distributions on training and test sets for KFD. The data sets depicted are Twonorm and Waveform. Observe that training and test distributions for Waveform are noticeably different; this might explain KFD's lower classification performance if compared to BFD, see Table 2.

parameters. This paper has established the theoretical foundations of the approach and has shown that for a range of simple toy problems the methodology does discover sensible kernel parameters. The optimisation is only guaranteed to find a local minimum and therefore the quality of the solution can be sensitive to the initialisation. We performed experiments on real world data obtaining results which are competitive with the state of the art, moreover, we were able to do some relevance determination on the data set features.

Future directions of this work will be centred on sparsifying the kernel matrix. We intend to adapt the informative vector machine model to our framework (Lawrence et al., 2003). This should make larger data sets practical because at present, we are restricted by the $O\left(N^3\right)$ complexity associated with inverting the kernel matrix. Another direction of research will consist of allowing

the model to learn in the presence of label noise, building on work by Lawrence and Schölkopf (2001).

Acknowledgements

Both authors gratefully acknowledge support from the EPSRC grant number GR/R84801/01 'Learning Classifiers from Sloppily Labelled Data'. T.P.C. wishes to thank the continuous support provided by Banco de México and helpful discussions with Guido Sanguinetti and Gavin C. Cawley. Finally, we thank Jonathan Laidler for comments on the final version of this manuscript.

Appendix A. Weight Space Approach

In order to derive the distribution of $\mathbf{w}$ under the constraint d, we first realise that the combination of $p(\mathbf{w}|\mathcal{D})$ and $p(d|\mathcal{D},\mathbf{w},\gamma)$ yields a Gaussian distribution. Therefore, after conditioning on d, the resulting distribution will be Gaussian with the form $p(\mathbf{w}|\mathcal{D},d,\gamma) = \lim_{\gamma\to\infty} \mathcal{N}(\bar{\mathbf{w}},\Sigma)$ and parameters

$$\bar{\mathbf{w}} = \lim_{\gamma\to\infty} \gamma d \Sigma \Delta\mathbf{m} \tag{34}$$

and

$$\Sigma = \lim_{\gamma\to\infty} \left(\mathbf{B} + \gamma\Delta\mathbf{m}\Delta\mathbf{m}^T\right)^{-1}. \tag{35}$$

Inversion of Σ through Morrison-Woodbury formula allows us to take the limit, such as is shown below

$$\Sigma = \lim_{\gamma\to\infty} \left(\mathbf{B}^{-1} - \frac{\mathbf{B}^{-1}\Delta\mathbf{m}\Delta\mathbf{m}^T\mathbf{B}^{-1}}{\gamma^{-1} + \Delta\mathbf{m}^T\mathbf{B}^{-1}\Delta\mathbf{m}}\right),$$

hence

$$\Sigma = \mathbf{B}^{-1} - \frac{\mathbf{B}^{-1}\Delta\mathbf{m}\Delta\mathbf{m}^T\mathbf{B}^{-1}}{\Delta\mathbf{m}^T\mathbf{B}^{-1}\Delta\mathbf{m}}.$$

The mean $\bar{\mathbf{w}}$ can be obtained by substituting Equation 35 (without evaluating the limit) into Equation 34. Then, the application of Morrison-Woodbury formula and some extra manipulations will lead to a form suitable for taking the limit.

Appendix B. Expressing $p\left(\hat{\mathbf{t}}\middle|\mathbf{f},\mathbf{y}\right)$ in terms of f and L

Disregarding an additive constant, the log of the modified noise model $p\left(\hat{\mathbf{t}}\middle|\mathbf{f},\mathbf{y}\right)$ is

$$\mathcal{L}(\hat{c}_0,\hat{c}_1) = -\frac{\beta}{2}\sum_{n=1}^{N}\left[y_n(\hat{c}_1 - f_n)^2 + (1-y_n)(\hat{c}_0 - f_n)^2\right],$$

where we have used Equation 6 as base. From Equation 9, we substitute the values of each estimate $\hat{c}_q$ so that

$$\begin{aligned}\mathcal{L} &= -\frac{\beta}{2}\sum_{n=1}^{N}\left[y_n\left(\frac{1}{N_1}\mathbf{y}_1^T\mathbf{f}-f_n\right)^2+(1-y_n)\left(\frac{1}{N_0}\mathbf{y}_0^T\mathbf{f}-f_n\right)^2\right]\\ &= -\frac{\beta}{2}\left(\mathbf{f}\mathbf{f}^T-\frac{1}{N_1}\mathbf{y}_1^T\mathbf{f}\mathbf{f}^T\mathbf{y}_1-\frac{1}{N_0}\mathbf{y}_0^T\mathbf{f}\mathbf{f}^T\mathbf{y}_0\right)\\ &= -\frac{\beta}{2}\left(\mathbf{f}^T\mathbf{L}\mathbf{f}\right).\end{aligned}$$

Appendix C. Gaussian Process Approach

In this section we find the parameters of a new projected data point, namely its mean $\bar{f}^\star$ and variance $(\sigma^\star)^2$, as specified in Equations 21 and 22. In the model we have three types of distributions: $p\left(\hat{\mathbf{t}}\middle|\mathbf{y},\mathbf{f}\right)$, the noise model; $p(\mathbf{f}_+)$, an extended GP prior that includes the point $f^\star$ and the constraint $p(d|\mathbf{y},\mathbf{f},\gamma)$. In order to derive the distribution $p(f^\star|\mathcal{D},d,\gamma)$ we first compute the joint distribution

$$p\left(\mathbf{f}_+,\hat{\mathbf{t}},d\middle|\mathbf{y},\gamma\right)=p\left(\hat{\mathbf{t}}\middle|\mathbf{y},\mathbf{f}\right)p(d|\mathbf{y},\mathbf{f},\gamma)\,p(\mathbf{f}_+),$$

and then take advantage of the separability of $\mathbf{f}_+$ into $\left[\mathbf{f}^T,f^\star\right]^T$ to be able to marginalise the latent variables $\mathbf{f}$, which are associated with the training set. In other words we do

$$p\left(f^\star,\hat{\mathbf{t}},d\middle|\mathbf{y},\gamma\right) = \int p\left(f^\star,\mathbf{f},\hat{\mathbf{t}},d\middle|\mathbf{y},\gamma\right)\partial\mathbf{f} \tag{36}$$

The rest of the process consists of conditioning $f^\star$ on the targets $\hat{\mathbf{t}}$ and the distance d and on taking the limit $\gamma\to\infty$. In the remaining part of this section we detail this process.

C.1 Derivations

Grouping Equations 18, 19 and 20 gives

$$p(\mathbf{f}_+,\hat{\mathbf{t}},d\big|\mathbf{y},\gamma) \propto \exp\left\{-\frac{\beta}{2}\mathbf{f}^T\mathbf{L}\mathbf{f}-\frac{1}{2}\mathbf{f}_+\mathbf{K}_+^{-1}\mathbf{f}_+-\frac{\gamma}{2}\left(d-\mathbf{f}^T\Delta\hat{\mathbf{y}}\right)^2\right\}.$$

The idea consists of expanding and collecting terms in $\mathbf{f}$ and $f^\star$. In order to do so, we partition the inverse of the extended kernel by making $\mathbf{K}_+^{-1}=\begin{bmatrix}\mathbf{C} & \mathbf{c}\\ \mathbf{c}^T & c_\star\end{bmatrix}$. Thus we know that the product

$$\mathbf{f}_+^T\mathbf{K}_+^{-1}\mathbf{f}_+ = \mathbf{f}^T\mathbf{C}\mathbf{f}+2f^\star\mathbf{c}^T\mathbf{f}+c_\star(f^\star)^2.$$

Hence we get

$$p(f^\star,\mathbf{f},\hat{\mathbf{t}},d\big|\mathbf{y},\gamma) \propto \exp\left\{-\tfrac{1}{2}\mathbf{f}^T\mathbf{Q}\mathbf{f}-\left(f^\star\mathbf{c}-\gamma d\Delta\hat{\mathbf{y}}\right)^T\mathbf{f}-\tfrac{1}{2}c_\star(f^\star)^2\right\} \tag{37}$$

with

$$\mathbf{Q}=\left(\beta\mathbf{L}+\mathbf{C}+\gamma\Delta\hat{\mathbf{y}}\Delta\hat{\mathbf{y}}^T\right). \tag{38}$$

MARGINALISING LATENT VARIABLES

We are now in position to determine the distribution for a new mapping $f^\star$. The marginalisation of $\mathbf{f}$ is done by computing the integral of Equation 36 using as integrand the expression in (37). First we observe that the term $\left(-\frac{1}{2}c_\star (f^\star)^2\right)$ will be required when computing the distribution over $f^\star$ and when taking the limit, so it will be kept apart from the integral and used at a later stage.

The integral in Equation 36 is of exponential form, so we know how to solve it in a straightforward way. See below that

$$\int \exp\left(-\frac{1}{2}\mathbf{f}^T\mathbf{Q}\mathbf{f}+\mathbf{h}^T\mathbf{f}\right)\partial\mathbf{f} \propto \exp\left(\frac{1}{2}\mathbf{h}^T\mathbf{Q}^{-1}\mathbf{h}\right),$$

where we recognise that

$$\mathbf{h} \quad = \quad -(f^\star\mathbf{c}-\gamma d\Delta\hat{\mathbf{y}})^T \ .$$

Therefore the result, after incorporating $\left(-\frac{1}{2}c_\star (f^\star)^2\right)$, is

$$p(\, f^\star, \hat{\mathbf{t}}, d \,|\, \mathbf{y}, \gamma) \quad \propto \quad \exp\left\{\frac{1}{2}\mathbf{h}^T\mathbf{Q}^{-1}\mathbf{h}-\frac{1}{2}c_\star (f^\star)^2\right\} \ . \tag{39}$$

OBTAINING CONDITIONAL DISTRIBUTION

The distribution $p(\, f^\star | \, \mathcal{D}, d, \gamma)$, with $\mathcal{D} = \left(\hat{\mathbf{t}}, \mathbf{y}\right)$, is obtained by conditioning the expression in (39) on $\hat{\mathbf{t}}$ and d. Therefore we will work with the argument inside the *exponential* of (39) and group terms in $f^\star$, ignoring the rest. We begin by substituting $\mathbf{h}$ and expanding

$$\frac{1}{2}(f^\star\mathbf{c}-\gamma d\Delta\hat{\mathbf{y}})^T\mathbf{Q}^{-1}(f^\star\mathbf{c}-\gamma d\Delta\hat{\mathbf{y}})-\frac{1}{2}c_\star (f^\star)^2 =$$

$$-\frac{1}{2}\left(c_\star-\mathbf{c}^T\mathbf{Q}^{-1}\mathbf{c}\right)\left[(f^\star)^2+\frac{2\gamma d\mathbf{c}^T\mathbf{Q}^{-1}\Delta\hat{\mathbf{y}}}{c_\star-\mathbf{c}^T\mathbf{Q}^{-1}\mathbf{c}}f^\star\right]+\frac{1}{2}\left[(\gamma d)^2\Delta\hat{\mathbf{y}}^T\mathbf{Q}^{-1}\Delta\hat{\mathbf{y}}\right].$$

Completing the squares on $f^\star$ gives a Gaussian

$$p(\, f^\star | \, \mathcal{D}, d, \gamma) \quad \propto \quad \exp\left\{-\tfrac{1}{2(\sigma^\star)^2}\left(f^\star-\bar{f^\star}\right)^2\right\}$$

where the variance is

$$(\sigma^\star)^2 = \lim_{\gamma\to\infty}\left(c_\star-\mathbf{c}^T\mathbf{Q}^{-1}\mathbf{c}\right)^{-1} \tag{40}$$

and the mean,

$$\bar{f^\star} = -\lim_{\gamma\to\infty}\gamma d\,(\sigma^\star)^2\,\mathbf{c}^T\mathbf{Q}^{-1}\Delta\hat{\mathbf{y}}. \tag{41}$$

WORKING OUT $(\sigma^\star)^2$

We now determine the limit $\gamma\to\infty$ in Equation 40. First, we express each of the terms that form the inverse of the kernel matrix:

$$c_\star = \left(k_\star-\mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}\right)^{-1}, \tag{42}$$

$$\mathbf{c} = -c_\star \mathbf{K}^{-1}\mathbf{k}, \tag{43}$$

and

$$\mathbf{C} = \mathbf{K}^{-1} + c_\star \mathbf{K}^{-1}\mathbf{k}\mathbf{k}^T\mathbf{K}^{-1}. \tag{44}$$

Partitioning the kernel matrix implies that

$$\mathbf{K}_+ \quad = \quad \begin{pmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k_\star \end{pmatrix}.$$

Substituting (38) and (43) into Equation 40 gives

$$(\sigma^\star)^2 = \lim_{\gamma\to\infty} \left(c_\star - c_\star^2 \mathbf{k}^T \left[\beta\mathbf{KLK} + \gamma\mathbf{K}\Delta\hat{\mathbf{y}}\Delta\hat{\mathbf{y}}^T\mathbf{K} + \mathbf{KCK} \right]^{-1} \mathbf{k} \right)^{-1}.$$

The product **KCK** can be worked out by using Equation 44. Therefore

$$(\sigma^\star)^2 = \lim_{\gamma\to\infty} \left(c_\star - c_\star^2 \mathbf{k}^T \left[\beta\mathbf{KLK} + \gamma\mathbf{K}\Delta\hat{\mathbf{y}}\Delta\hat{\mathbf{y}}^T\mathbf{K} + \mathbf{K} + c_\star\mathbf{k}\mathbf{k}^T \right]^{-1} \mathbf{k} \right)^{-1}.$$

Defining

$$\mathbf{D}_\gamma = \gamma\mathbf{K}\Delta\hat{\mathbf{y}}\Delta\hat{\mathbf{y}}^T\mathbf{K} + \mathbf{A}, \tag{45}$$

with $\mathbf{A} = \beta\mathbf{KLK} + \mathbf{K}$ leads to

$$\begin{aligned} (\sigma^\star)^2 &= \lim_{\gamma\to\infty} \left[c_\star - c_\star^2 \mathbf{k}^T \left(\mathbf{D}_\gamma + c_\star \mathbf{k}\mathbf{k}^T \right)^{-1} \mathbf{k} \right]^{-1}, \\ &= c_\star^{-1} + \mathbf{k}^T \mathbf{D}_\gamma^{-1} \mathbf{k}. \end{aligned}$$

Using Equation 42, we arrive to an expression for which $(\sigma^\star)^2$ only depends on γ by the term $\mathbf{D}_\gamma$,

$$(\sigma^\star)^2 = \lim_{\gamma\to\infty} \left[k_\star - \mathbf{k}^T \left(\mathbf{K}^{-1} - \mathbf{D}_\gamma^{-1} \right) \mathbf{k} \right]. \tag{46}$$

Working with $\mathbf{D}_\gamma^{-1}$ Inversion of $\mathbf{D}_\gamma$ (Equation 45) through the Morrison-Woodbury lemma allows us to obtain **D** by taking the limit. See below.

$$\mathbf{D}_\gamma^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}} \left(\frac{1}{\gamma} + \Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}} \right)^{-1} \Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1}.$$

Therefore, by taking $\gamma \to \infty$ we obtain

$$\mathbf{D} = \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}} \left(\Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}} \right)^{-1} \Delta\hat{\mathbf{y}}^T\mathbf{K}\mathbf{A}^{-1} \right)^{-1}.$$

Substituting this expression into (46) gives the desired result,

$$(\sigma^\star)^2 = k_\star - \mathbf{k}^T \left(\mathbf{K}^{-1} - \mathbf{D}^{-1} \right) \mathbf{k}.$$

WORKING OUT THE MEAN

Substituting Equations 38 and 43, the values of $\mathbf{Q}$ and $\mathbf{c}$, into Equation 41 leads to

$$\bar{f}^{\star} = \lim_{\gamma\to\infty} \gamma d \left(\sigma^{\star}\right)^2 c_{\star} \mathbf{k}^T \left(\mathbf{D}_{\gamma} + c_{\star}\mathbf{k}\mathbf{k}^T\right)^{-1} \mathbf{K}\Delta\hat{\mathbf{y}}.$$

Inverting the matrix $\left(\mathbf{D}_{\gamma} + c_{\star}\mathbf{k}\mathbf{k}^T\right)$ and substituting the value of $c_{\star}$ gives

$$\bar{f}^{\star} = \lim_{\gamma\to\infty} \gamma d \left(\sigma^{\star}\right)^2 \left[k_{\star} - \mathbf{k}^T \left(\mathbf{K}^{-1} - \mathbf{D}_{\gamma}^{-1}\right)\mathbf{k}\right]^{-1} \mathbf{k}^T \mathbf{D}_{\gamma}^{-1} \mathbf{K}\Delta\hat{\mathbf{y}}.$$

Using (46) implies that

$$\bar{f}_{\star} = \lim_{\gamma\to\infty} \gamma d \mathbf{k}^T \mathbf{D}_{\gamma}^{-1} \mathbf{K}\Delta\bar{\mathbf{y}}.$$

Substituting (45) and inverting gives

$$\bar{f}^{\star} = \lim_{\gamma\to\infty} d \left(\frac{1}{\gamma} + \Delta\hat{\mathbf{y}}^T \mathbf{K}\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}\right)^{-1} \mathbf{k}^T \mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}.$$

Taking the limit gives the desired result

$$\bar{f}_{\star} = \frac{d\mathbf{k}^T \mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}}{\Delta\hat{\mathbf{y}}^T \mathbf{K}\mathbf{A}^{-1}\mathbf{K}\Delta\hat{\mathbf{y}}}.$$

Appendix D. Obtaining MAP Solution for β

Making

$$\begin{aligned} V &= \Sigma_{n=1}^{N} y_n \left(\hat{c}_1 - f_n\right)^2 + \Sigma_{n=1}^{N} \left(1 - y_n\right) \left(\hat{c}_0 - f_n\right)^2, \\ &= \sigma_1^2 + \sigma_0^2. \end{aligned}$$

the modified noise model[9] becomes

$$p\left(\hat{\mathbf{t}}|\mathbf{f},\beta\right) = \frac{\beta^{N/2}}{(2\pi)^{N/2}} \exp\left\{-\frac{\beta}{2}V\right\}.$$

Then combining it with a gamma prior $G\left(\beta|a,b\right)$ gives a posterior of the form $G\left(\beta|N/2+a,(V/2+b)\right)$, that is

$$p\left(\beta|\hat{\mathbf{t}},\mathbf{f}\right) \propto \beta^{N/2+a-1} \exp\left\{-\beta\left(\frac{V}{2}+b\right)\right\}.$$

Taking the derivative of the log of this distribution, equating the result to zero and solving will give Equation 27.

Appendix E. Experimental Setup

In this section we give more details on the way we carried out our experiments on toy and real data.

9. Use Equation 6 and substitute the class centres c_q by their estimates $\hat{c}_q$.

E.1 Toy Data

In all our experiments with synthetic data we used an RBF kernel of the form

$$k(\mathbf{x}_i, \mathbf{x}_j) = \theta_1 \exp\left(-\frac{\theta_2}{2}\left|\left|\mathbf{x}^i - \mathbf{x}^j\right|\right|^2\right) + \theta_3 \delta_{ij}. \tag{47}$$

For KFD and LS-SVM we worked with the bandwidth of the kernel $\sigma = 1/\theta_2$, whereas for BFD we used θ_2 itself. The nugget parameter θ_3 ensures that $\mathbf{K}$ can be inverted at all times.

Regarding model training, we used the matlab implementation of Baudat and Anouar (2000) to solve the generalized eigenvalue problem, see their function `BuildGDA`. Furthermore, we used the function `crossvalidate`, provided by Pelckmans et al. (2003) and Suykens et al. (2002), to cross-validate the values of σ and of the 'threshold' of the mininum accepted eigenvalue. The latter was used instead of the regularisation coefficient because of the way the eigenvalue problem is solved, see Baudat's implementation for more details. In the LS-SVM case, we used the toolbox LS-SVMlab of Pelckmans et al. (2003) and Suykens et al. (2002) to do the classifications. The values of σ and C were cross-validated, with the latter being the coefficient associated with the support vector formulation. Lastly, in BFD, we used our own implementation which is available at

`http://www.dcs.shef.ac.uk/~neil/bfd`.

In this case, the function `scg` provided in Netlab's toolbox (Nabney, 2002) was used to adapt kernel parameters.

E.2 Benchmark Data Sets

In the BFD experiments with an RBF kernel, we used $N_x = 8$ different initialisations of the parameter θ_2 (Equation 47) during the training phase. The initialisations selected ranged from 1×10^{-4} to 1×10^4; initialisations that produced numerical errors were ignored. We trained on the first 5 realisations (partitions) of each data set and computed their marginal loglikelihood. In this way, an array of $N_x \times 5$ elements was obtained. See below.

$$\begin{array}{cccc} \mathit{Init}. & \mathit{Part}.1 & \dots & \mathit{Part}.5 \\ \theta_2^1 & \Theta_t^{(11)} & \dots & \Theta_t^{(15)} \\ \vdots & \vdots & \vdots & \vdots \\ \theta_2^{N_x} & \Theta_t^{(N_x 1)} & \dots & \Theta_t^{(N_x 5)} \end{array}$$

For each partition, we selected the vector $\Theta_t^{(ip)}$ with highest associated marginal likelihood, with $p \in [1,5]$ and $i \in [1,8]$. Hence the original array of 40 elements Θ_t^{ip} was reduced to an array of 5 elements. The final vector of parameters was determined by extracting the trained values θ_2 from each element of the reduced array and taking the median over them. The selected vector Θ_t^{sel} was the one associated with the median value of θ_2.

For the ARD experiments, we changed $N_x = 3$. The initialisations were given by $\left[1 \times 10^{-3}, 1, 10\right]$.

The Bayesian LS-SVM experiments were carried out using the toolbox LS-SVMLab (see Suykens et al., 2002). We trained LS-SVM's on the first five realisations of the training data and selected the median of the parameters. We observed that this algorithm was susceptible to fall into local minima so in order to avoid this problem, we used the function `bay_initlssvm` to have good initialisations. The ARD rankings were obtained by applying the function `bay_lssvmARD`.

References

Deepak K. Agarwal. Shrinkage estimator generalizations of proximal support vector machines. In *KDD '02: Proceedings of the eighth ACM SIGKDD International conference on Knowledge Discovery and Data Mining*, pages 173–182, New York, NY, USA, 2002. ACM Press.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, May 1950.

Gaston Baudat and Fatiha Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39:1–38, 1977.

Richard O. Duda and Peter E. Hart. *Pattern Recognition and Scene Analysis*. John Wiley, 1973.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179, 1936.

Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press Inc., Boston, Massachusetts, 2nd edition, 1990.

Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In *KDD '01: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86, New York, NY, USA, 2001. ACM Press.

Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, USA, 3rd edition, 1996.

Robert B. Gramacy and Herbert K. H. Lee. Gaussian processes and limiting linear models. Technical Report ams2005-01, Department of Applied Mathematics and Statistics, University of California, Santa Cruz., 2005.

Ian T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1st edition, 1986.

Kevin J. Lang and Michael J. Witbrock. Learning to tell two spirals apart. In David S. Touretzky, Geoff E. Hinton, and Terrence J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*. Morgan Kauffman, 1988.

Neil D. Lawrence and Bernhard Schölkopf. Estimating a kernel Fisher discriminant in the presence of label noise. In Carla E. Brodley and Andrea P. Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning*, Williamstown, MA, July 2001. Morgan Kauffman.

Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: the informative vector machine. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616, Cambridge, MA, 2003. MIT Press.

David J. C. Mackay. Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995.

Donald Michie, David J. Spiegelhalter, and Charles C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

Sebastian Mika. A mathematical approach to kernel Fisher algorithm. In Todd K. Leen and Thomas G. Dietterich Völker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 591–597, Cambridge, MA, 2001. MIT Press.

Sebastian Mika. *Kernel Fisher Discriminants*. PhD thesis, Technischen Universität, Berlin, Germany, 2002.

Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, and Klaus-Robert Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, E. Wilson J. Larsen, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

Ian T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, 2002.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

Anthony O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, Series B (Methodological)*, 40(1):1–42, 1978.

Kristiaan Pelckmans, Johan A. K. Suykens, Tony Van Gestel, Jos De Brabanter, Lukas Lukas, Bart Hamers, Bart De Moor, and Joos Vandewalle. *LS-SVMlab Toolbox User's Guide*. Katholieke Universiteit, Leuven. ESAT-SCD-SISTA, 2003.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. Technical Report NC-TR-98-021, Royal Holloway College, University of London, U. K., 1998.

Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

Volker Roth. Outlier detection with one-class kernel Fisher discriminants. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1169–1176, Cambridge, MA, 2005. MIT Press.

David Ruppert, Matthew P. Wand, and Raymond J. Carroll. *Semiparametric Regression*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, U. K., 2003.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

Johan A. K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machines. *Neural Processing Letters*, 9(3):293–300, 1999.

Tony Van Gestel, Johan A. K. Suykens, Gert Lanckriet, Annemie Lambrechts, Bart de Moor, and Joos Vandewalle. Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel discriminant analysis. *Neural Computation*, 14(5):1115–1147, 2002.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

Christopher K. I. Williams. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In Michael I. Jordan, editor, *Learning in Graphical Models*, D, Behavioural and social sciences 11. Kluwer, Dordrecht, The Netherlands, 1999.

Journal of Machine Learning Research 7 (2006) 493–518 Submitted 7/05; Published 3/06

Learning Recursive Control Programs from Problem Solving

Pat Langley LANGLEY@CSLI.STANFORD.EDU
Dongkyu Choi DONGKYUC@STANFORD.EDU
Computational Learning Laboratory
Center for the Study of Language and Information
Stanford University
Stanford, CA 94305–4115 USA

Editors: Roland Olsson and Ute Schmid

Abstract

In this paper, we propose a new representation for physical control – teleoreactive logic programs – along with an interpreter that uses them to achieve goals. In addition, we present a new learning method that acquires recursive forms of these structures from traces of successful problem solving. We report experiments in three different domains that demonstrate the generality of this approach. In closing, we review related work on learning complex skills and discuss directions for future research on this topic.

Keywords: teleoreactive control, logic programs, problem solving, skill learning

1. Introduction

Human skills have a hierarchical character, with complex procedures defined in terms of more basic ones. In some domains, these skills are recursive in nature, in that structures are specified in terms of calls to themselves. Such recursive procedures pose a clear challenge for machine learning that deserves more attention than it has received in the literature. In this paper we present one response to this problem that relies on a new representation for skills and a new method for acquiring them from experience.

We focus here on the task of learning controllers for physical agents. We are concerned with acquiring the structure and organization of skills, rather than tuning their parameters, which we view as a secondary learning issue. We represent skills as *teleoreactive logic programs*, a formalism that incorporates ideas from logic programming, reactive control, and hierarchical task networks. This framework can encode hierarchical and recursive procedures that are considerably more complex than those usually studied in research on reinforcement learning (Sutton & Barton, 1998) and behavioral cloning (Sammut, 1996), but they can still be executed in a reactive yet goal-directed manner. As we will see, it also embodies constraints that make the learning process tractable.

We assume that an agent uses hierarchical skills to achieve its goals whenever possible, but also that, upon encountering unfamiliar tasks, it falls back on problem solving. The learner begins with primitive skills for the domain, including knowledge of their applicability conditions and their effects, which lets it compose them to form candidate solutions. When the system overcomes such an impasse successfully, which may require substantial search, it learns a new skill that it stores in memory for use on future tasks. Thus, skill acquisition is incremental and intertwined with problem

solving. Moreover, learning is cumulative in that skills acquired early on form the building blocks for those mastered later. We have incorporated our assumptions about representation, performance, and learning into ICARUS, a cognitive architecture for controlling physical agents.

Any approach to acquiring hierarchical and recursive procedures from problem solving must address three issues. These concern identifying the hierarchical organization of the learned skills, determining when different skills should have the same name or head, and inferring the conditions under which each skill should be invoked. To this end, our approach to constructing teleoreactive logic programs incorporates ideas from previous work on learning and problem solving, but it also introduces some important innovations.

In the next section, we specify our formalism for encoding initial and learned knowledge, along with the performance mechanisms that interpret them to produce behavior. After this, we present an approach to problem solving on novel tasks and a learning mechanism that transforms the results of this process into executable logic programs. Next, we report experimental evidence that the method can learn control programs in three recursive domains, as well as use them on tasks that are more complex than those on which they were acquired. We conclude by reviewing related work on learning and proposing some important directions for additional research.

2. Teleoreactive Logic Programs

As we have noted, our approach revolves around a representational formalism for the execution of complex procedures – teleoreactive logic programs. We refer to these structures as "logic programs" because their syntax is similar to the Horn clauses used in Prolog and related languages. We have borrowed the term "teleoreactive" from Nilsson (1994), who used it to refer to systems that are goal driven but that also react to their current environment. His examples incorporated symbolic control rules but were not cast as logic programs, as we assume here.

A teleoreactive logic program consists of two interleaved knowledge bases. One specifies a set of concepts that the agent uses to recognize classes of situations in the environment and describe them at higher levels of abstraction. These monotonic inference rules have the same semantics as clauses in Prolog and a similar syntax. Each clause includes a single head, stated as a predicate with zero or more arguments, along with a body that includes one or more positive literals, negative literals, or arithmetic tests. In this paper, we assume that a given head appears in only one clause, thus constraining definitions to be conjunctive, although the formalism itself allows disjunctive concepts.

ICARUS distinguishes between primitive conceptual clauses, which refer only to percepts that the agent can observe in the environment, and complex clauses, which refer to other concepts in their bodies. Specific percepts play the same role as ground literals in traditional logic programs, but, because they come from the environment and change over time, we do not consider them part of the program. Table 1 presents some concepts from the Blocks World. Concepts like *unstackable* and *pickupable* are defined in terms of the concepts *clear*, *on*, *ontable*, and *hand-empty*, the subconcept *clear* is defined in terms of *on*, and *on* is defined using two cases of the percept *block*, along with arithmetic tests on their attributes.

A second knowledge base contains a set of skills that the agent can execute in the world. Each skill clause includes a head (a predicate with zero or more arguments) and a body that specifies a set of start conditions and one or more components. Primitive clauses have a single start condition (often a nonprimitive concept) and refer to executable actions that alter the environment. They also

```
((on ?block1 ?block2)
 :percepts  ((block ?block1 xpos ?x1 ypos ?y1)
             (block ?block2 xpos ?x2 ypos ?y2 height ?h2))
 :tests     ((equal ?x1 ?x2) (>= ?y1 ?y2) (<= ?y1 (+ ?y2 ?h2))))
((ontable ?block ?table)
 :percepts  ((block ?block xpos ?x1 ypos ?y1)
             (table ?table xpos ?x2 ypos ?y2 height ?h2))
 :tests     ((>= ?y1 ?y2) (<= ?y1 (+ ?y2 ?h2))))
((clear ?block)
 :percepts  ((block ?block))
 :negatives ((on ?other ?block)))
((holding ?block)
 :percepts  ((hand ?hand status ?block)
             (block ?block)))
((hand-empty)
 :percepts  ((hand ?hand status ?status))
 :tests     ((eq ?status empty)))
((three-tower ?b1 ?b2 ?b3 ?table)
 :percepts  ((block ?b1) (block ?b2) (block ?b3) (table ?table))
 :positives ((on ?b1 ?b2) (on ?b2 ?b3) (ontable ?b3 ?table)))
((unstackable ?block ?from)
 :percepts  ((block ?block) (block ?from))
 :positives ((on ?block ?from) (clear ?block) (hand-empty)))
((pickupable ?block ?from)
 :percepts  ((block ?block) (table ?from))
 :positives ((ontable ?block ?from) (clear ?block) (hand-empty)))
((stackable ?block ?to)
 :percepts  ((block ?block) (block ?to))
 :positives ((clear ?to) (holding ?block)))
((putdownable ?block ?to)
 :percepts  ((block ?block) (table ?to))
 :positives ((holding ?block)))
```

Table 1: Examples of concepts from the Blocks World.

specify the effects of their execution, stated as literals that hold after their completion, and may state requirements that must hold during their execution. Table 2 shows the four primitive skills for the Blocks World, which are similar in structure and spirit to STRIPS operators, but may be executed in a durative manner.

In contrast, nonprimitive skill clauses specify how to decompose activity into subskills. Because a skill may refer to itself, either directly or through a subskill, the formalism supports recursive definitions. For this reason, nonprimitive skills do not specify effects, which can depend on the number of levels of recursion, nor do they state requirements. However, the head of each complex skill refers to some concept that the skill aims to achieve, an assumption Reddy and Tadepalli

```
((unstack ?block ?from)
 :percepts  ((block ?block ypos ?y)
             (block ?from))
 :start     ((unstackable ?block ?from))
 :actions   ((*grasp ?block) (*move-up ?block ?y))
 :effects   ((clear ?from)
             (holding ?block)))
((pickup ?block ?from)
 :percepts  ((block ?block ypos ?y)
             (table ?from))
 :start     ((pickupable ?block ?from))
 :actions   ((*grasp ?block) (*move-up ?block ?y))
 :effects   ((holding ?block)))
((stack ?block ?to)
 :percepts  ((block ?block)
             (block ?to xpos ?x ypos ?y height ?height))
 :start     ((stackable ?block ?to))
 :actions   ((*move-over ?block ?x)
             (*move-down ?block (+ ?y ?height))
             (*ungrasp ?block))
 :effects   ((on ?block ?to)
             (hand-empty)))
((putdown ?block ?to)
 :percepts  ((block ?block)
             (table ?to ypos ?y height ?height))
 :start     ((putdownable ?block ?to))
 :actions   ((*move-sideways ?block)
             (*move-down ?block (+ ?y ?height))
             (*ungrasp ?block))
 :effects   ((ontable ?block ?to)
             (hand-empty)))
```

Table 2: Primitive skills for the Blocks World domain. Each skill clause has a head that specifies its name and arguments, a set of typed variables, a single start condition, a set of effects, and a set of executable actions, each marked by an asterisk.

(1997) have also made in their research on task decomposition. This connection between skills and concepts constitutes a key difference between the current approach and our earlier work on hierarchical skills in ICARUS (Choi et al., 2004; Langley & Rogers, 2004), and it figures centrally in the learning methods we describe later. Table 3 presents some recursive skills for the Blocks World, including two clauses for achieving the concept *clear*.

Teleoreactive logic programs are closely related to Nau et al.'s SHOP (1999) formalism for hierarchical task networks. This organizes knowledge into tasks, which serve as heads of clauses, and methods, which specify how to decompose tasks into subtasks. Primitive methods describe the effects of basic actions, much like STRIPS operators. Each method also states its application

```
((clear ?B) 1
 :percepts  ((block ?C) (block ?B))
 :start     ((unstackable ?C ?B))
 :skills    ((unstack ?C ?B)))

((hand-empty) 2
 :percepts  ((block ?C) (table ?T))
 :start     ((putdownable ?C ?T))
 :skills    ((putdown ?C ?T)))

((unstackable ?B ?A) 3
 :percepts  ((block ?A) (block ?B))
 :start     ((on ?B ?A) (hand-empty))
 :skills    ((clear ?B) (hand-empty)))

((clear ?A) 4
 :percepts  ((block ?B) (block ?A))
 :start     ((on ?B ?A) (hand-empty))
 :skills    ((unstackable ?B ?A)
             (unstack ?B ?A)))
```

Table 3: Some nonprimitive skills for the Blocks World domain that involve recursive calls. Each skill clause has a head that specifies the goal it achieves, a set of typed variables, one or more start conditions, and a set of ordered subskills. Numbers after the head distinguish different clauses that achieve the same goal.

conditions, which may involve predicates that are defined in logical axioms. In our framework, skill heads correspond to tasks, skill clauses are equivalent to methods, and concept definitions play the role of axioms. In this mapping, teleoreactive logic programs are a special class of hierarchical task networks in which nonprimitive tasks always map onto declarative goals and in which top-level goals and the preconditions of primitive methods are always single literals. We will see that these two assumptions play key roles in our approach to problem solving and learning.

Note that every skill/task *S* can be expanded into one or more sequences of primitive skills. For each skill *S* in a teleoreactive logic program, if *S* has concept *C* as its head, then every expansion of *S* into such a sequence must, if executed successfully, produce a state in which *C* holds. This constraint is weaker than the standard assumption made for macro-operators (e.g., Iba, 1988); it does not guarantee that, once initiated, the sequence will achieve *C*, since other events may intervene or the agent may encounter states in which one of the primitive skills does not apply. However, if the sequence of primitive skills can be run to completion, then it will achieve the goal literal *C*. The approach to learning that we report later is designed to acquire programs with this characteristic, and we give arguments to this effect at the close of Section 4.

3. Interpreting Teleoreactive Logic Programs

As their name suggests, teleoreactive logic programs are designed for reactive execution in a goal-driven manner, within a physical setting that changes over time. As with most reactive controllers, the associated performance element operates in discrete cycles, but it also involves more sophisticated processing than most such frameworks.

On each decision cycle, ICARUS updates a perceptual buffer with descriptions of all objects that are visible in the environment. Each such percept specifies the object's type, a unique identifier, and zero or more attributes. For example, in the Blocks World these would include structures like *(block A xpos 5 ypos 1 width 1 height 1).* In this paper, we emphasize domains in which the agent perceives the same objects on successive time steps but in which some attributes change value. However, we will also consider teleoreactive systems for domains like in-city driving (Choi et al., 2004) in which the agent perceives different objects as it moves through the environment.

Once the interpreter has updated the perceptual buffer, it invokes an inference module that elaborates on the agent's perceptions. This uses concept definitions to draw logical conclusions from the percepts, which it adds to a conceptual short-term memory. This dynamic store contains higher-level beliefs, cast as relational literals, that are instances of generic concepts. The inference module operates in a bottom-up, data-driven manner that starts from descriptions of perceived objects, such *(block A xpos 5 ypos 1 width 1 height 1)* and *(block B xpos 5 ypos 0 width 1 height 1)*, matches these against the conditions in concept definitions, and infers beliefs about primitive concepts like *(on A B)*. These trigger inferences about higher-level concepts, such as *(clear A)*, which in turn support additional beliefs like *(unstackable A B)*. This process continues until the agent has added all beliefs that are implied by its perceptions and concept definitions.[1]

After the inference module has augmented the agent's perceptions with high-level beliefs, the architecture's execution module inspects this information to decide what actions to take in the environment. To this end, it also examines its current goal, which must be encoded as an instance of some known concept, and its skills, which tell it how to accomplish such goals. Unlike inference, the execution process proceeds in a top-down manner, finding paths through the skill hierarchy that terminate in primitive skills with executable actions. We define a *skill path* to be a chain of skill instances that starts from the agent's goal and descends through the hierarchy along subskill links, unifying the arguments of each subskill consistently with those of its parent.

Furthermore, the execution module only considers skill paths that are *applicable*. This holds if no concept instance that corresponds to a goal along the path is satisfied, if the requirements of the terminal (primitive) skill instance are satisfied, and if, for each skill instance in the path not executed on the previous cycle, the start condition is satisfied. This last constraint is necessary because skills may take many cycles to achieve their desired effects, making it important to distinguish between their initiation and their continuation. To this end, the module retains the path through the skill hierarchy selected on the previous time step, along with the variable bindings needed to reconstruct it.

For example, imagine a situation in which the block C is on B, B is on A, and A is on the table, in which the goal is *(clear A)*, and in which the agent knows the primitive skills in Table 2 and the recursive skills in Table 3. Further assume that this is the first cycle, so that no previous activities are under way. In this case, the only path through the skill hierarchy is *[(clear A) 4], [(unstackable B A) 3], [(clear B) 1], [(unstack C B)]*. Applying the primitive skill *(unstack C B)* produces a new situation that leads to new inferences, and in which the only applicable path is *[(clear A) 4], [(unstackable B A) 3], [(hand-empty) 2], [(putdown C T)]*. This enables a third path on the next cycle, *[(clear A) 4], [(unstack B A)]*, which generates a state in which the agent's goal is satisfied. Note that this process operates much like the proof procedure in Prolog, except that it involves activities that extend over time.

The interpreter incorporates two preferences that provide a balance between reactivity and persistence. First, given a choice between two or more subskills, it selects the first one for which the corresponding concept instance is not satisfied. This bias supports reactive control, since the agent reconsiders previously completed subskills and, if unexpected events have undone their effects, re-executes them to correct the situation. Second, given a choice between two or more applicable skill paths, it selects the one that shares the most elements from the start of the path executed on the

1. Although this mechanism reasons over structures similar to Horn clauses, its operation is closer in spirit to the elaboration process in Soar (Laird et al., 1986) than to the query-driven reasoning in Prolog.

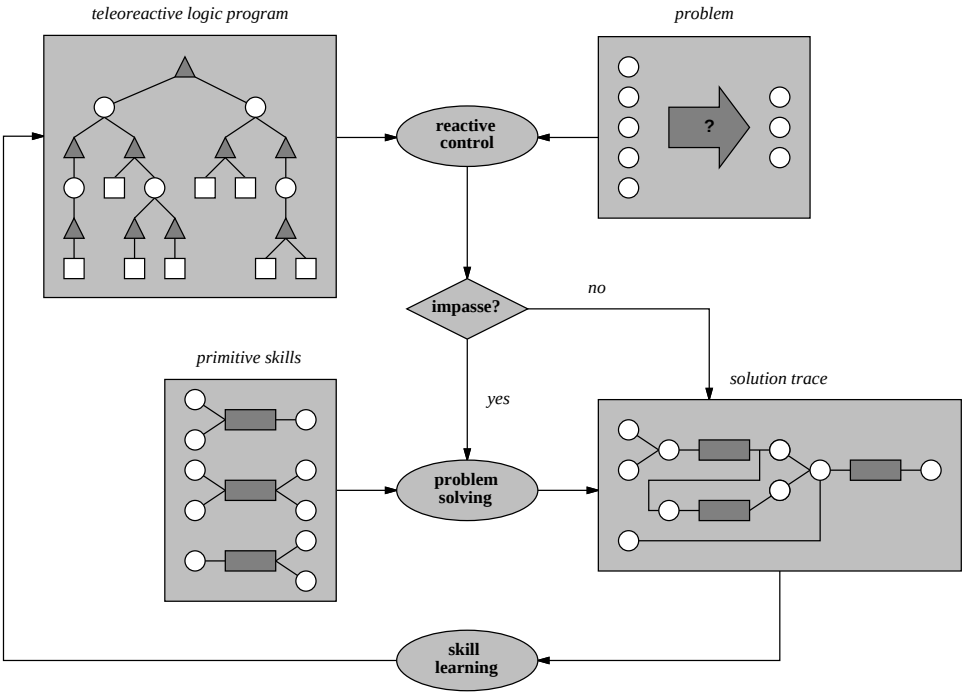


Figure 1: Organization of modules for reactive execution, problem solving, and skill learning, along with their inputs and outputs.

previous cycle. This bias encourages the agent to keep executing a high-level skill it has started until it achieves the associated goal or becomes inapplicable.

Most research on reactive execution emphasizes dynamic domains in which unexpected events can occur that fall outside the agent's control. Domains like the Blocks World do not have this character, but this does not mean one cannot utilize a reactive controller to direct behavior (e.g., see Fern et al., 2004). Moreover, we have also demonstrated (Choi et al., 2004) the execution module's operation in the domain of in-city driving, which requires reactive response to an environment that changes dynamically. Our framework is relevant to both types of settings.

To summarize, ICARUS' procedure for interpreting teleoreactive logic programs relies on two interacting processes – conceptual inference and skill execution. On each cycle, the architecture perceives objects and infers instances of conceptual relations that they satisfy. After this, it starts from the current goal and uses these beliefs to check the conditions on skill instances to determine which paths are applicable, which in turn constrains the actions it executes. The environment changes, either in response to these actions or on its own, and the agent begins another inference-execution cycle. This looping continues until the concept that corresponds to the agent's top-level goal is satisfied, when it halts.

4. Solving Problems and Learning Skills

Although one can construct teleoreactive logic programs manually, this process is time consuming and prone to error. Here we report an approach to learning such programs whenever the agent encounters a problem or subproblem that its current skills do not cover. In such cases, the architecture attempts to solve the problem by composing its primitive skills in a way that achieves the goal. Typically, this problem-solving process requires search and, given limited computational resources,

may fail. However, when the effort is successful the agent produces a trace of the solution in terms of component skills that achieved the problem's goal. The system transforms this trace into new skill clauses, which it adds to memory for use on future tasks.

Figure 1 depicts this overall organization. As in some earlier problem-solving architectures like PRODIGY (Minton, 1988) and Soar (Laird et al., 1986), problem solving and learning are tightly linked and both are driven by impasses. A key difference is that, in these systems, learning produces search-control knowledge that makes future problem solving more effective, whereas in our framework it generates teleoreactive logic programs that the agent uses in the environment. Nevertheless, there remain important similarities that we discuss later at more length.

4.1 Means-Ends Problem Solving

As described earlier, the execution module selects skill clauses that should achieve the current goal and that have start conditions which match its current beliefs about the environment. Failure to retrieve such a clause produces an impasse that leads the architecture to invoke its problem-solving module. Table 4 presents pseudocode for the problem solver, which utilizes a variant of means-ends analysis (Newell & Simon, 1961) that chains backward from the goal. This process relies on a goal stack that stores both subgoals and skills that might accomplish them. The top-level goal is simply the lowest element on this stack.

Despite our problem-solving method's similarity to means-ends analysis, it differs from standard formulation in three important ways:

- whenever the skill associated with the topmost goal on the stack becomes applicable, the system executes it in the environment, which leads to tight interleaving of problem solving and control;
- both the start conditions of primitive skills (i.e., operators) and top-level goals must be cast as single relational literals, which may be defined concepts;[2]
- backward chaining can occur not only off the start condition of primitive skills but also off the definition of a concept, which means the single-literal assumption causes no loss of generality.

As we will see shortly, the second and third of these assumptions play key roles in the mechanism for learning new skills, but we should first examine the operation of the problem-solving process itself.

As Table 4 indicates, the problem solver pushes the current goal G onto the goal stack, then checks it on each execution cycle to determine whether it has been achieved. If so, then the module pops the stack and focuses on G's parent goal or, upon achieving the top-level goal, simply halts. If the current goal G is not satisfied, then the architecture retrieves all nonprimitive skills with heads that unify with G and, if any participate in applicable paths through the skill hierarchy, selects the first one found and executes it. This execution may require many cycles, but eventually it produces a new environmental state that either satisfies G or constitutes another impasse.

If the problem solver cannot find any complex skills indexed by the goal G, it instead retrieves all primitive skills that produce G as one of their effects. The system then generates candidate instances of these skills by inserting known objects as their arguments. To select among these skill instances, it expands the instantiated start condition of each skill instance to determine how many of its primitive components are satisfied, then selects the one with the fewest literals unsatisfied in the current situation. If the candidates tie on this criterion, then it selects one at random. If the selected

2. We currently define all concepts manually, but it would not be difficult to have the system define them automatically for operator preconditions and conjunctive goals.

```
Solve(G)
  Push the goal literal G onto the empty goal stack GS.
  On each cycle,
     If the top goal G of the goal stack GS is satisfied,
     Then pop GS.
     Else if the goal stack GS does not exceed the depth limit,
          Let S be the skill instances whose heads unify with G.
          If any applicable skill paths start from an instance in S,
          Then select one of these paths and execute it.
          Else let M be the set of primitive skill instances that
                  have not already failed in which G is an effect.
               If the set M is nonempty,
               Then select a skill instance Q from M.
                    Push the start condition C of Q onto goal stack GS.
               Else if G is a complex concept with the unsatisfied
                       subconcepts H and with satisfied subconcepts F,
                    Then if there is a subconcept I in H that has not yet failed,
                         Then push I onto the goal stack GS.
                         Else pop G from the goal stack GS.
                              Store information about failure with G's parent.
                    Else pop G from the goal stack GS.
                         Store information about failure with G's parent.
```

Table 4: Pseudocode for interleaving means-ends problem solving with skill execution.

skill instance's condition is met, the system executes the skill instance in the environment until it achieves the associated goal, which it then pops from the stack. If the condition is not satisfied, the architecture makes it the current goal by pushing it onto the stack.

However, if the problem solver cannot find any skill clause that would achieve the current goal G, it uses G's concept definition to decompose the goal into subgoals. If more than one subgoal is unsatisfied, the system selects one at random and calls the problem solver on it recursively, which makes it the current goal by pushing it onto the stack. This leads to chaining off the start condition of additional skills and/or the definitions of other concepts. Upon achieving a subgoal, the architecture pops the stack and, if other subconcepts remain unsatisfied, turns its attention to achieving them. Once all have been satisfied, this means the parent goal G has been achieved, so it pops the stack again and focuses on the parent.

Of course, the problem-solving module must make decisions about which skills to select during skill chaining and the order in which it should tackle subconcepts during concept chaining. The system may well make the incorrect choice at any point, which can lead to failure on a given subgoal when no alternatives remain or when it reaches the maximum depth of the goal stack. In such cases, it pops the current goal, stores the failed candidate with its parent goals to avoid considering them in the future, and backtracks to consider other options. This strategy produces depth-first search through the problem space, which can require considerable time on some tasks.

Figure 2 shows an example of the problem solver's behavior on the Blocks World in a situation where block A is on the table, block B is on A, block C is on B, and the hand is empty. Upon being given the objective *(clear A)*, the architecture looks for any executable skill with this goal as its head. When this fails, it looks for a skill that has the objective as one of its effects. In this case,

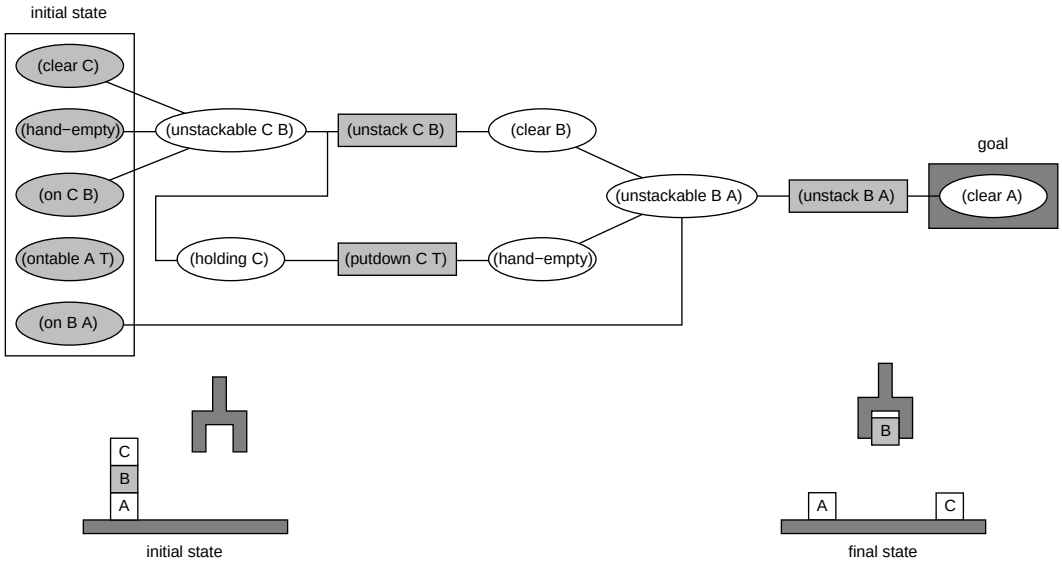


Figure 2: A trace of successful problem solving in the Blocks World, which ellipses indicating concepts/goals and rectangles denoting primitive skills.

invoking the primitive skill instance *(unstack B A)* would produce the desired result. However, this cannot yet be applied because its instantiated start condition, *(unstackable B A)*, does not hold, so the system stores the skill instance with the initial goal and pushes this subgoal onto the stack.

Next, the problem solver attempts to retrieve skills that would achieve *(unstackable B A)* but, because it has no such skills in memory, it resorts to chaining off the definition of *unstackable*. This involves three instantiated subconcepts – *(clear)*, *(on B A)*, and *(hand-empty)* – but only the first of these is unsatisfied, so the module pushes this onto the goal stack. In response, it considers skills that would produce this literal as an effect and retrieves the skill instance *(unstack C B)*, which it stores with the current goal.

In this case, the start condition of the selected skill, *(unstackable C B)*, already holds, so the architecture executes *(unstack C B)*, which alters the environment and causes the agent to infer *(clear B)* from its percepts. In response, it pops this goal from the stack and reconsiders its parent, *(unstackable B A)*. Unfortunately, this has not yet been achieved because executing the skill has caused the third of its component concept instances, *(hand-empty)*, to become false. Thus, the system pushes this onto the stack and, upon inspecting memory, retrieves the skill instance *(putdown C T)*, which it can and does execute.

This second step achieves the subgoal *(hand-empty)*, which in turn lets the agent infer *(unstackable B A)*. Thus, the problem solver pops this element from the goal stack and executes the skill instance it had originally selected, *(unstack B A)*, in the new situation. Upon completion, the system perceives that the altered environment satisfies the top-level goal, *(clear A)*, which leads it to halt, since it has solved the problem. Both our textual description and the graph in Figure 2 represent the trace of successful problem solving; as noted earlier, finding such a solution may well involve search, but we have omitted missteps that require backtracking for the sake of clarity.

Despite the clear evidence that humans often resort to means-ends analysis when they encounter novel problems (Newell & Simon, 1961), this approach to problem solving has been criticized in

the AI planning community because it searches over a space of totally ordered plans. As a result, on problems for which the logical structure of a workable plan is only partially ordered, it can carry out extra work by considering alternative orderings that are effectively equivalent. However, the method also has clear advantages, such as low memory load because it must retain only the current stack rather than a partial plan. Moreover, it provides direct support for interleaving of problem solving and execution, which is desirable for agents that must act in their environment.

Of course, executing a component skill before it has constructed a complete plan can lead the system into difficulty, since the agent cannot always backtrack in the physical world and can produce situations from which it cannot recover without starting over on the problem. In such cases, the problem solver stores the goal for which the executed skill caused trouble, along with everything below it in the stack. The system begins the problem again, this time avoiding the skill and selecting another option. If a different execution error occurs this time, the module again stores the problematic skill and its context, then starts over once more. In this way, the architecture continues to search the problem space until it achieves its top-level goal or exceeds the number of maximum allowed attempts.[3]

4.2 Goal-Driven Composition of Skills

Any method for learning teleoreactive logic programs or similar structures must address three issues. First, it must determine the structure of the hierarchy that decomposes problems into subproblems. Second, the technique must identify when different clauses should have the same head and thus be considered in the same situations. Finally, it must infer the conditions under which to invoke each clause. The approach we describe here relies on results produced by the problem solver to answer these questions. Just as problem solving occurs whenever the system encounters an impasse, that is, a goal it cannot achieve by executing stored skills, so learning occurs whenever the system resolves an impasse by successful problem solving. The ICARUS architecture shares this idea with earlier frameworks like Soar and PRODIGY, although the details differ substantially.

The response to the first issue is that *hierarchical structure is determined by the subproblems handled during problem solving*. As Figure 2 illustrates, this takes the form of a semilattice in which each subplan has a single root node. This structure follows directly from our assumptions that each primitive skill has one start condition and each goal is cast as a single literal. Because the problem solver chains backward off skill and concept definitions, the result is a hierarchical structure that suggests a new skill clause for each subgoal. Table 5 (a) presents the clauses that the system proposes based on the solution to the *(clear A)* problem, without specifying their heads or conditions. Figure 2 depicts the resulting hierarchical structure, using numbers to indicate the order in which the system generates each clause.

The answer to the second question is that *the head of a learned skill clause is the goal literal that the problem solver achieved for the subproblem that produced it*. This follows from our assumption that the head of each clause in a teleoreactive logic program specifies some concept that the clause will produce if executed. At first glance, this appears to confound skills with concepts, but another view is that it indexes skill clauses by the concepts they achieve. Table 5 (b) shows the clauses learned from the problem-solving trace in Figure 2 once the heads have been inserted. Note that this

3. The problem solver also starts over if it has not achieved the top-level objective within a given number of cycles. Jones and Langley (in press) report another variant of means-ends problem solving that uses a similar restart strategy but keeps no explicit record of previous failed paths.

(a)
```
(<head> 1                                (<head> 3
 :percepts ((block ?C) (block ?B))        :percepts ((block ?A) (block ?B))
 :start    <conditions>                   :start    <conditions>
 :skills   ((unstack ?C ?B)))             :skills   ((clear ?B) (hand-empty)))

(<head> 2                                (<head> 4
 :percepts ((block ?C) (table ?T))        :percepts ((block ?B) (block ?A))
 :start    <conditions>                   :start    <conditions>
 :skills   ((putdown ?C ?T)))             :skills   ((unstackable ?B ?A)
                                                     (unstack ?B ?A)))
```

(b)
```
((clear ?B) 1                            ((unstackable ?B ?A) 3
 :percepts ((block ?C) (block ?B))        :percepts ((block ?A) (block ?B))
 :start    <conditions>                   :start    <conditions>
 :skills   ((unstack ?C ?B)))             :skills   ((clear ?B) (hand-empty)))

((hand-empty) 2                          ((clear ?A) 4
 :percepts ((block ?C) (table ?T))        :percepts ((block ?B) (block ?A))
 :start    <conditions>                   :start    <conditions>
 :skills   ((putdown ?C ?T)))             :skills   ((unstackable ?B ?A)
                                                     (unstack ?B ?A)))
```

(c)
```
((clear ?B) 1                            ((unstackable ?B ?A) 3
 :percepts ((block ?C) (block ?B))        :percepts ((block ?A) (block ?B))
 :start    ((unstackable ?C ?B))          :start    ((on ?B ?A) (hand-empty))
 :skills   ((unstack ?C ?B)))             :skills   ((clear ?B) (hand-empty)))

((hand-empty) 2                          ((clear ?A) 4
 :percepts ((block ?C) (table ?T))        :percepts ((block ?B) (block ?A))
 :start    ((putdownable ?C ?T))          :start    ((on ?B ?A) (hand-empty))
 :skills   ((putdown ?C ?T)))             :skills   ((unstackable ?B ?A)
                                                     (unstack ?B ?A)))
```

Table 5: Skill clauses for the Blocks World learned from the trace in Figure 2 (a) after hierarchical structure has been determined, (b) after the heads have been identified, and (c) after the start conditions have been inserted. Numbers after the heads indicate the order in which clauses are generated.

strategy leads directly to the creation of recursive skills whenever a conceptual predicate *P* is the goal and *P* also appears as a subgoal. In this example, because *(clear A)* is the top-level goal and *(clear B)* occurs as a subgoal, one of the clauses learned for *clear* is defined recursively, although this happens indirectly through *unstackable*.

Clearly, introducing recursive statements can easily lead to overly general or even nonterminating programs. Our approach avoids the latter because the problem solver never considers a subgoal if it already occurs earlier in the goal stack; this ensures that subgoals which involve the same predicate always have different arguments. However, we still require some means to address the third issue of determining conditions on learned clauses that guards against the danger of overgen-

```
Learn(G)
  If the goal G involves skill chaining,
  Then let S1 and S2 be G's first and second subskills.
       If subskill S1 is empty,
       Then create a new skill clause N with head G,
              with the head of S2 as the only subskill,
              and with the same start condition as S2.
            Return the literal for skill clause N.
       Else create a new skill clause N with head G,
              with the heads of S1 and S2 as ordered subskills,
              and with the same start condition as S1.
            Return the literal for skill clause N.
  Else if the goal G involves concept chaining,
       Then let C1, ..., Ck be G's initially satisfied subconcepts.
            Let Ck+1, ..., Cn be G's stored subskills.
            Create a new skill clause N with head G,
              with Ck+1, ..., Cn as ordered subskills,
              and with C1, ..., Ck as start conditions.
            Return the literal for skill clause N.
```

Table 6: Pseudocode for creation of skill clauses through goal-driven composition.

eralization. The response differs depending on whether the problem solver resolves an impasse by chaining backward on a primitive skill or by chaining on a concept definition.

Suppose the agent achieves a subgoal G through skill chaining, say by first applying skill S_1 to satisfy the start condition for S_2 and executing the skill S_2, producing a clause with head G and ordered subskills S_1 and S_2. In this case, *the start condition for the new clause is the same as that for* S_1, since when S_1 is applicable, the successful completion of this skill will ensure the start condition for S_2, which in turn will achieve G. This differs from traditional methods for constructing macro-operators, which analytically combine the preconditions of the first operator and those preconditions of later operators it does not achieve. However, S_1 was either selected because it achieves S_2's start condition or it was learned during its achievement, both of which mean that S_1's start condition is sufficient for the composed skill.[4]

In contrast, suppose the agent achieves a goal concept G through concept chaining by satisfying the subconcepts $G_{k+1}, \ldots, G_n$, in that order, while subconcepts $G_1, \ldots, G_k$ were true at the outset. In response, the system would construct a new skill clause with head G and the ordered subskills $G_{k+1}, \ldots, G_n$, each of which the system already knew and used to achieve the associated subgoal or which it learned from the successful solution of one of the subproblems. In this case, *the start condition for the new clause is the conjunction of subgoals that were already satisfied beforehand*. This prevents execution of the learned clause when some of $G_1, \ldots, G_k$ are not satisfied, in which case the sequence $G_{k+1}, \ldots, G_n$ may not achieve the goal G. Table 6 gives pseudocode that summarizes both methods for determining the conditions on new clauses.

Table 5 (c) presents the conditions learned for each of the skill clauses learned from the trace in Figure 2. Two of these (clauses 1 and 2) are trivial because they result from degenerate subproblems that the system solves by chaining off a single primitive operator. Another skill clause (3) is more

4. If skill S_2 is executed without invoking another skill to meet its start condition, the method creates a new clause, with S_2 as its only subskill, that restates the original skill in a new form with G in its head.

```
Solve(G)
  Push the goal literal G onto the empty goal stack GS.
  On each cycle,
     If the top goal G of the goal stack GS is satisfied,
     Then pop GS and let New be Learn(G).
          If G's parent P involved skill chaining,
          Then store New as P's first subskill.
          Else if G's parent P involved concept chaining,
               Then store New as P's next subskill.
     Else if the goal stack GS does not exceed the depth limit,
          Let S be the skill instances whose heads unify with G.
          If any applicable skill paths start from an instance in S,
          Then select one of these paths and execute it.
          Else let M be the set of primitive skill instances that
                  have not already failed in which G is an effect.
               If the set M is nonempty,
               Then select a skill instance Q from M.
                    Push the start condition C of Q onto goal stack GS.
                    Store Q with goal G as its last subskill.
                    Mark goal G as involving skill chaining.
               Else if G is a complex concept with the unsatisfied
                       subconcepts H and with satisfied subconcepts F,
                    Then if there is a subconcept I in H that has not yet failed,
                         Then push I onto the goal stack GS.
                              Store F with G as its initially true subconcepts.
                              Mark goal G as involving concept chaining.
                         Else pop G from the goal stack GS.
                              Store information about failure with G's parent.
                    Else pop G from the goal stack GS.
                         Store information about failure with G's parent.
```

Table 7: Pseudocode for interleaved problem solving and execution extended to support goal-driven composition of skills. New steps are indicated in italic font.

interesting because it results from chaining off the concept definition for *unstackable*. This has the start conditions *(on ?A ?B)* and *(hand-empty)* because the subconcept instances *(on A B)* and *(hand-empty)* held at the outset.[5] The final clause (4) is most intriguing because it results from using a learned clause (3) followed by the primitive skill instance *(unstack B A)*. In this case, the start condition is the same as that for the first subskill clause (3).

Upon initial inspection, the start conditions for clause 3 for achieving *unstackable* may appear overly general. However, recall that the skill clauses in a teleoreactive logic program are interpreted not in isolation but as parts of chains through the skill hierarchy. The interpreter will not select a path for execution unless all conditions along the path from the top clause to the primitive skill are satisfied. This lets the learning method store very abstract conditions for new clauses with less danger of overgeneralization. On reflection, this scheme is the only one that makes sense for recursive control programs, since static preconditions cannot characterize such structures. Rather,

5. Although primitive skills have only one start condition, we do not currently place this constraint on learned clauses, as they are not used in problem solving and it makes acquired programs more readable.

the architecture must compute appropriate preconditions dynamically, depending on the depth of recursion. The Prolog-like interpreter used for skill selection provides this flexibility and guards against overly general behavior.

We refer to the learning mechanism that embodies these answers as *goal-driven composition*. This process operates in a bottom-up fashion, with new skills being formed whenever a goal on the stack is achieved. The method is fully incremental, in that it learns from single training cases, and it is interleaved with problem solving and execution. The technique shares this characteristic with analytical methods for learning from problem solving, such as those found in Soar and PRODIGY. But unlike these methods, it learns hierarchical skills that decompose problems into subproblems, and, unlike most methods for forming macro-operators, it acquires disjunctive and recursive skills. Moreover, learning is cumulative in that skills learned from one problem are available for use on later tasks. Taken together, these features make goal-driven composition a simple yet powerful approach to learning logic programs for reactive control. Nor is the method limited to working with means-ends analysis; it should operate over traces of any planner that chains backward from a goal.

The architecture's means-ends module must retain certain information during problem solving to support the composition of new skill clauses. Table 7 presents expanded pseudocode that specifies this information and when the system stores it. The form and content is similar to that recorded in Veloso and Carbonell's (1993) approach to derivational analogy. The key difference is that their system stores details about subgoals, operators, and preconditions in specific cases that drive future problem solving, whereas our approach transforms these instances into generalized hierarchical structures for teleoreactive control.

We should clarify that the current implementation invokes a learned clause only when it is applicable in the current situation, so the problem solver never chains off its start conditions. Mooney (1989) incorporated a similar constraint into his work on learning macro-operators to avoid the utility problem (Minton, 1990), in which learned knowledge reduces search but leads to slower behavior. However, we have extended his idea to cover cases in which learned skills can solve subproblems, which supports greater transfer across tasks. In our framework, this assumption means that clauses learned from skill chaining have a left-branching structure, with the second subskill being primitive.

In Section 2, we stated that every skill clause in a teleoreactive logic program can be expanded into one or more sequences of primitive skills, and that each sequence, if executed legally, will produce a state that satisfies the clause's head concept. Here we argue that goal-driven composition learns sets of skill clauses for which this condition holds. As in most research on planning, we assume that the preconditions and effects of primitive skills are accurate, and also that no external forces interfere. First consider a clause with the head H that has been created as the result of successful chaining off a primitive skill. This learned clause is guaranteed to achieve the goal concept H because H must be an effect of its final subskill or the chaining would never have occurred.

Now consider a clause with the head H that has been created as the result of successful chaining off a conjunctive definition of the concept H. This clause describes a situation in which some subconcepts of H hold but others must still be achieved to make H true. Some subconcepts may become unsatisfied in the process and need to be reachieved, but the ordering on subgoals found during problem solving worked for the particular objects involved, and replacing constants with variables will not affect the result. Thus, if the clause's start conditions are satisfied, achieving the subconcepts in the specified order will achieve H. Remember that our method does *not* guaran-

tee, like those for learning macro-operators, that a given clause expansion *will* run to completion. Whether this occurs in a given domain is an empirical question, to which we now turn.

5. Experimental Studies of Learning

As previously reported (Choi & Langley, 2005), the means-ends problem solving and learning mechanisms just described construct properly organized teleoreactive logic programs. After learning, the agent can simply retrieve and execute the acquired programs to solve similar problems without falling back to problem solving. Here we report promising results from more systematic and extensive experiments. The first two studies involve inherently recursive but nondynamic domains, whereas the third involves a dynamic driving task.

5.1 Blocks World

The Blocks World involves an infinitely large table with cubical blocks, along with a manipulator that can grasp, lift, carry, and ungrasp one block at a time. In this domain, we wrote an initial program with nine concepts and four primitive skills. Additionally, we provided a concept for each of four different goals.[6] Theoretically, this knowledge is sufficient to solve any problem in the domain, but the extensive search required would make it intractable to solve tasks with many blocks using only basic knowledge. In fact, only 20 blocks are enough to make the system search for half an hour. Therefore, we wanted the system to learn teleoreactive logic programs that it could execute recursively to solve problems with arbitrary complexity. We have already discussed a recursive program acquired from one training problem, which requires clearing the lowest object in a stack of three blocks, but many other tasks are possible.

To establish that the learned programs actually help the architecture to solve more complex problems, we ran an experiment that compared the learning and non-learning versions. We presented the system with six ten-problem sets of increasing complexity, one after another. More specifically, we used sets of randomly generated problems with 5, 10, 15, 20, 25, and 30 blocks. If the goal-driven composition mechanism is effective, then it should produce noticeable benefits in harder tasks when the learning is active.

We carried out 200 runs with different randomized orders within levels of task difficulty. In each case, we let the system run a maximum of 50 decision cycles before starting over on a problem and attempt a task at most five times before it gave up. For this domain, we set the maximum depth of the goal stack used in problem solving to eight. Figure 3 displays the number of execution cycles and the CPU time required for both conditions, which shows a strong benefits from learning.

With number of cycles as the performance measure, we see a systematic decrease as the system gains more experience. Every tenth problem introduces five additional objects, but the learning system requires no extra effort to solve them. The architecture has constructed general programs that let it achieve familiar goals for arbitrary numbers of blocks without resorting to deliberative problem solving. Inspection reveals that it acquires the nonprimitive skill clauses in Table 3, as well as additional ones that make recursive calls. In contrast, the nonlearning system requires more decision cycles on harder problems, although this levels off later in the curve, as the problem solver gives up on very difficult tasks.

6. These concerned achieving situations in which a given block is clear, one block is on another, one block is on another and a third block is on the table, and three blocks are arranged in a tower.

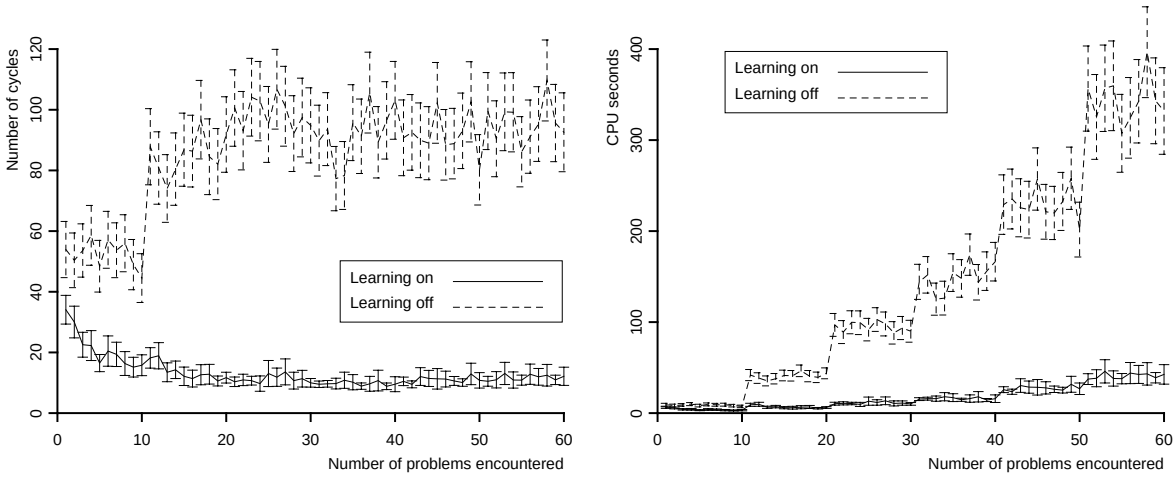


Figure 3: Execution cycles and CPU times required to solve a series of 5, 10, 15, 20, 25, and 30-block problems (10 different tasks at each level) in the Blocks World as a function of the number of tasks with and without learning. Each learning curve shows the mean over 200 different task orders and 95 percent confidence intervals.

The results for solution time show similar benefits, with the learning condition substantially outperforming the condition without. However, the figure also indicates that even the learning version slows down somewhat as it encounters problems with more blocks. Analysis of individual runs suggests this results from the increased cost of matching against objects in the environment, which is required in both the learning and nonlearning conditions. This poses an issue, not for our approach to skill construction but to our architectural framework, so it deserves attention in future research.

Table 8 shows the average results for each level of problem complexity, including the probability that the system can solve a problem within the allowed number of cycles and attempts. In addition to presenting the first two measures at more aggregate levels, it also reveals that, without learning, the chances of finding a solution decrease with the number of blocks in the problem. Letting the system carry out more search would improve these scores, but only at the cost of increasing the number of cycles and CPU time needed to solve the more difficult problems.

5.2 FreeCell Solitaire

FreeCell is a solitaire game with eight columns of stacked cards, all face up and visible to the player, that has been used in AI planning competitions (Bacchus, 2001). There are four free cells, which can hold any single card at a time, and four home cells that correspond to the four different suits. The goal is to move all the cards on the eight columns to the home cells for their suits in ascending order. The player can move only the cards on the top of the eight columns and the ones in the free cells. Each card can be moved to a free cell, to the proper home cell, or to an empty column. In addition, the player can move a card to a column whose top card has the next number and a different color. As in the Blocks World, we provided a simulated environment that allows legal moves and updates the agent's perceptions.

Blocks	Learning			No Learning		
	cycles	CPU	P(sol)	cycles	CPU	P(sol)
5	21.25	4.03	0.997	52.52	8.82	0.958
10	13.61	6.90	0.997	85.15	40.60	0.857
15	11.22	11.13	0.995	98.82	94.93	0.816
20	9.76	16.09	0.997	92.06	149.05	0.863
25	11.04	27.41	0.996	91.77	230.43	0.842
30	11.67	40.85	0.995	95.89	344.49	0.826

Table 8: Aggregate scaling results for the Blocks World.

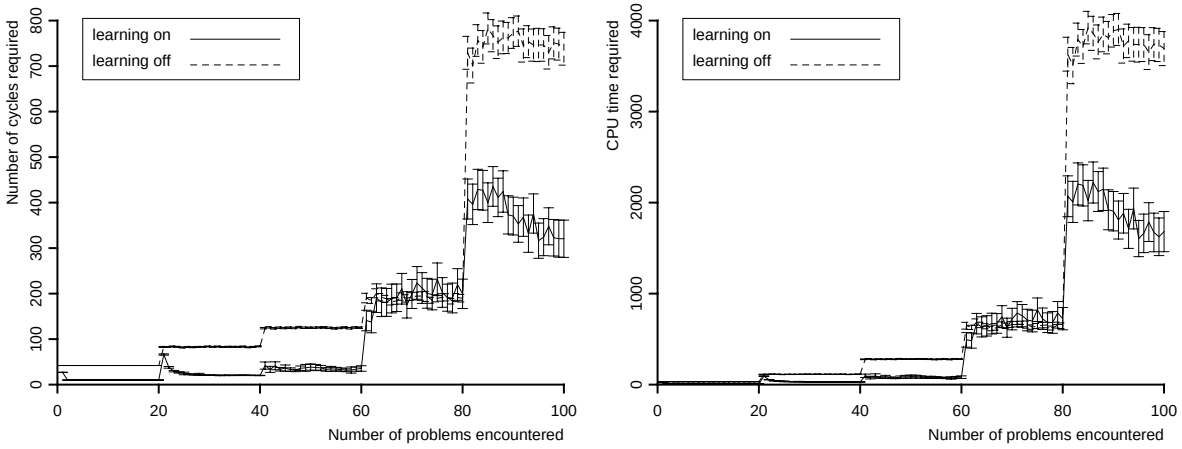


Figure 4: Execution cycles and CPU times required to solve a series of 8, 12, 16, 20, and 24-card FreeCell problems (20 different tasks each) as a function of the number of tasks with and without learning. Each learning curve shows the mean over 300 different task orders and 95 percent confidence intervals.

For this domain, we provided the architecture with an initial program which involves 24 concepts and 12 primitive skills that should, in principle, let it solve any initial configuration with a feasible solution path. (Most but not all FreeCell problems are solvable.) However, the agent may find a solution only after a significant amount of search using its means-ends problem solver. Again we desired the system to learn teleoreactive logic programs that it can execute on complex FreeCell problems with little or no search. In this case, we presented tasks as a sequence of five 20-problem sets with 8, 12, 16, 20, and 24 cards. On each problem, we let the system run at most 1000 decision cycles before starting over, attempt the task no more than five times before halting, and create goal stacks up to 30 in depth. We ran both the learning and nonlearning versions on 300 sets of randomly generated problems and averaged the results. Figure 4 shows the number of cycles and the CPU time required to solve tasks as a function of the number of problems encountered.

In the learning condition, the system rapidly acquired recursive FreeCell programs that reduced considerably the influence of task difficulty as compared to the nonlearning version. As before,

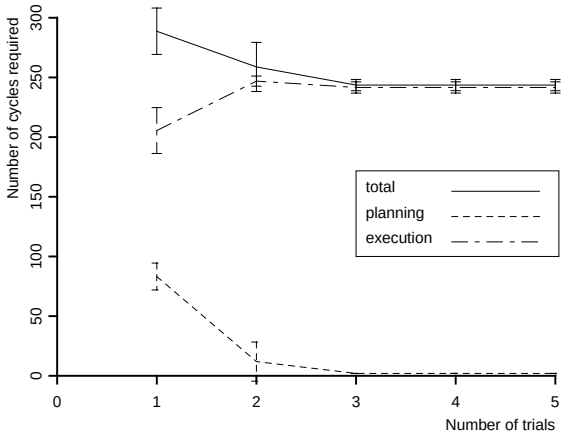


Figure 5: The total number of cycles required to solve a particular right-turn task along with the planning and execution times, as a function of the number of trials. Each learning curve shows the mean computed over ten sets of trials and 95 percent confidence intervals.

the benefits are reflected in both the number of cycles needed to solve problems and in the CPU time. However, increasing the number of cards in this domain can alter the structure of solutions, so the learning system continued to invoke means-ends problem solving in later portions of the curve. For instance, situations with 20 cards often require column-to-column moves that do not appear in simpler tasks, which caused comparable behavior in the two conditions at this complexity level. However, the learning system took advantage of this experience to handle 24-card problems with much less effort. Learning also increased the probability of solution (about 80 percent) over the nonlearning version (around 50 percent) on these tasks.

5.3 In-City Driving

The in-city driving domain involves a medium-fidelity simulation of a downtown driving environment. The city has several square blocks with buildings and sidewalks, street segments, and intersections. Each street segment includes a yellow center line and white dotted lane lines, and it has its own speed limit the agent should observe. Buildings on each block have unique addresses, to help the agent navigate through the city easily and to allow specific tasks like package deliveries. A typical city configuration we used has nine blocks, bounded by four vertical streets and four horizontal streets with four lanes each.

For this domain, we provided the system 41 concepts and 19 primitive skills. With the basic knowledge, the agent can describe its current situation at multiple levels of abstraction and perform actions for accelerating, decelerating, and steering left or right at realistic angles. Thus, it can operate a vehicle, but driving safely in a city environment is a totally different story. The agent must still learn how to stay aligned and centered within lane lines, change lanes, increase or decrease speed for turns, and stop for parking. To encourage such learning, we provided the agent with the task of moving to a destination on a different street segment that requires a right turn. To achieve this task, it resorted to problem solving, which found a solution path that involved changing to the

rightmost lane, staying aligned and centered until the intersection, steering right to place the car in the target segment, and finally aligning and centering in the new lane.

We recorded the total number of cycles to solve this task, along with its breakdown into the cycles devoted to planning and to execution, as a function of the number of trials. Figure 5 shows the learning curve that results from averaging over ten different sets of trials. As the system accumulates knowledge about the driving task, its planning effort effectively disappears, which leads to an overall reduction in the total cycles, even though the execution cycles increase slightly. The latter occurs because the vehicle happens to be moving in the right direction at the outset, which accidently brings it closer to the goal while the system is engaged in problem solving. After learning, the agent takes the same actions intentionally, which produces the increase in execution cycles. We should note that this task is dominated by driving time, which places a lower bound on the benefits of learning even when behavior becomes fully automatized.

We also inspected the skills that the architecture learned for this domain. Table 9 shows the five clauses it acquires by the end of a typical training run. These structures include two recursive references, one in which *in-intersection-for-right-turn* invokes itself directly, but also a more interesting one in which *driving-in-segment* calls itself indirectly through *in-segment*, *in-intersection-for-right-turn*, and *in-rightmost-lane*. Testing this teleoreactive logic program on streets with more lanes than occur in the training task suggests that it generalizes correctly to these situations.

6. Related Research

The basic framework we have reported in this paper incorporates ideas from a number of traditions. Our representation and organization of knowledge draws directly from the paradigm of logic programming (Clocksin & Mellish, 1981), whereas its utilization in a recognize-act cycle has more in common with production-system architectures (Neches, Langley, & Klahr, 1987). The reliance on heuristic search to resolve goal-driven impasses, coupled with the caching of generalized solutions, comes closest to the performance and learning methods used in problem-solving architectures like Soar (Laird, Rosenbloom, & Newell, 1986) and PRODIGY (Minton, 1990). Finally, we have already noted our debt to Nilsson (1994) for the notion of a teleoreactive system.

However, our approach differs from earlier methods for improving the efficiency of problem solvers in the nature of the acquired knowledge. In contrast to Soar and PRODIGY, which create flat control rules, our framework constructs hierarchical logic programs that incorporate nonterminal symbols. Methods for learning macro-operators (e.g., Iba, 1988; Mooney, 1989) have a similar flavor, in that they explicitly specify the order in which to apply operators, but they do not typically support recursive references. Shavlik (1989) reports a system that learns recursive macro-operators but that, like other work in this area, does not acquire reactive controllers.

Moreover, both traditions have used sophisticated analytical methods that rely on goal regression to collect conditions on control rules or macro-operators, nonincremental empirical techniques like inductive logic programming, or combinations of such methods (e.g., Estlin & Mooney, 1997). Instead, goal-driven composition transforms traces of successful means-ends search directly into teleoreactive logic programs, determining their preconditions by a simple method that involves neither analysis or induction, as normally defined, and that operates in an incremental and cumulative fashion.

Previous research on learning for reactive execution, like work on search control, has emphasized unstructured knowledge. For example, Benson's (1995) TRAIL acquires teleoreactive control

```
((driving-in-segment ?me ?g994 ?g1021)
 :percepts ((segment ?g994) (lane-line ?g1021) (self ?me))
 :start    ((in-segment ?me ?g994) (steering-wheel-straight ?me))
 :skills   ((in-lane ?me ?g1021)
            (centered-in-lane ?me ?g994 ?g1021)
            (aligned-with-lane-in-segment ?me ?g994 ?g1021)
            (steering-wheel-straight ?me)))
((driving-in-segment ?me ?g998 ?g1008)
 :percepts ((segment ?g998) (lane-line ?g1008) (self ?me))
 :start ((steering-wheel-straight ?me))
 :skills   ((in-segment ?me ?g998)
            (centered-in-lane ?me ?g998 ?g1008)
            (aligned-with-lane-in-segment ?me ?g998 ?g1008)
            (steering-wheel-straight ?me)))

((in-segment ?me ?g998)
 :percepts ((self ?me) (intersection ?g978) (segment ?g998))
 :start    ((last-lane ?g1021))
 :skills   ((in-intersection-for-right-turn ?me ?g978)
            (steer-for-right-turn ?me ?g978 ?g998)))
((in-intersection-for-right-turn ?me ?g978)
 :percepts ((lane-line ?g1021) (self ?me) (intersection ?g978))
 :start    ((last-lane ?g1021))
 :skills   ((in-rightmost-lane ?me ?g1021)
            (in-intersection-for-right-turn ?me ?g978)))
((in-rightmost-lane ?me ?g1021)
 :percepts ((self ?me) (lane-line ?g1021))
 :start    ((last-lane ?g1021))
 :skills   ((driving-in-segment ?me ?g994 ?g1021)))
```

Table 9: Recursive skill clauses learned for the in-city driving domain.

programs for use in physical environments, but it utilizes inductive logic programming to determine local rules for individual actions rather than hierarchical structures. Fern et al. (2004) report an approach to learning reactive controllers that trains itself on increasingly complex problems, but that also acquires decision lists for action selection. Khardon (1999) describes another method for learning ordered, but otherwise unstructured, control rules from observed problem solutions.

Our approach shares some features with research on inductive programming, which focuses on synthesizing iterative or recursive programs from input-output examples. For instance, Schmid's (2005) IPAL generates an initial program from the results of problem solving by replacing constants with constructive expressions with variables, then transforms it into a recursive program through inductive inference steps. Olsson's (1995) ADATE also generates recursive programs through program refinement transformations, but carries out an iterative deepening search guided by criteria like fit to training examples and syntactic complexity. Schmid's work comes closer to our own, in that both operate over problem-solving traces and generate recursive programs, but our method produces these structures directly, rather than using explicit transformation or revision steps.

Perhaps the closest relative to our approach is Reddy and Tadepalli's (1997) X-Learn, which acquires goal-decomposition rules from a sequence of training problems. Their system does not include an execution engine, but it generates recursive hierarchical plans in a cumulative manner that also identifies declarative goals with the heads of learned clauses. However, because it invokes forward-chaining rather than backward-chaining search to solve new problems, it relies on the trainer to determine program structure. X-Learn also uses a sophisticated mixture of analytical and relational techniques to determine conditions, rather than our much simpler method. Ruby and Kibler's (1991) SteppingStone has a similar flavor, in that it learns generalized decompositions through a mixture of problem reduction and forward-chaining search. Marsella and Schmidt's (1993) system also acquires task-decomposition rules by combining forward and backward search to hypothesize state pairs, which in turn produce rules that it revises after further experience.

Finally, we should mention another research paradigm that deals with speeding up the execution of logic programs. One example comes from Zelle and Mooney (1993), who report a system that combines ideas from explanation-based learning and inductive logic programming to infer the conditions under which clauses should be considered. Work in this area starts and ends with standard logic programs, whereas our system transforms a weak problem-solving method into an efficient program for reactive control. In summary, although our learning technique incorporates ideas from earlier frameworks, it remains distinct on a number of dimensions.

7. Directions for Future Research

Despite the promise of this new approach to representing, utilizing, and learning knowledge for teleoreactive control, our work remains in its early stages. Future research should demonstrate the acquisition of complex skills on additional domains. These should include both classical domains like logistics planning and dynamic settings like in-city driving. We have reported preliminary results on the latter, but our work in this domain to date has dealt with relatively simple skills, such as changing lanes and slowing down to park. Humans' driving knowledge is far more complex, and we should demonstrate that our methods are sufficient to acquire many more of them.

Note that, although driving involves reactive control, it also benefits from route planning and other high-level activities. Recall that our definition of teleoreactive logic programs, and our method for learning them, guarantees only that a skill will achieve its associated goal if it executes successfully, not that such execution is possible. For such guarantees, we must augment the current execution module with some lookahead ability, as Nau et al. (1999) have already done for hierarchical task networks. This will require additional effort from the agent, but still far less than solving a problem with means-ends analysis.

Another response would use inductive logic programming or related methods to learn additional conditions on skill clauses that ensure they will achieve their goal, even without lookahead. To this end, we can transform the results of lookahead search into positive and negative instances of clauses, based on whether they would lead to success, much as in early work on inducing search-control rules from solution paths (Sleeman et al., 1982). Even if such conditions are incomplete, they should still reduce the planning effort required to ensure the agent's actions will produce the desired outcome.

Another important limitation concerns our assumption that the agent always executes a skill to achieve a desired situation. The ability to express less goal-directed activities, such as playing a piano piece, are precisely what distinguishes hierarchical task networks from classical planning (Erol, Hendler, & Nau, 1994). We hope to extend our framework in this direction by generalizing

its notion of goals to include concepts that describe sets of situations that hold during certain time intervals. To support the hierarchical skill acquisition, this augmented representation will require extensions to both the problem solving and learning mechanisms. In addition, we should extend our framework to handle skill learning in nonserializable domains, such as tile-sliding puzzles, which motivated much of the early research on macro-operator formation (e.g., Iba, 1988).

Future work should also address a related form of overgeneralization we have observed on the Tower of Hanoi puzzle. In this domain, the approach learns reasonable hierarchical skills that can solve the task without problem solving, but that only do so about half the time. In other runs, the learned skills attempt to move the smallest disk to the wrong peg, which ultimately causes the system to fail. Humans often make similar errors but also learn to avoid them with experience. Inspection of the behavioral trace suggests this happens because one learned skill clause includes variables that are not mentioned in the head but are bound in the body. We believe that including contextual conditions about variables bound higher in the skill hierarchy will remove this nondeterminism and produce more correct behavior.

In addition, recall that the current system does not chain backward from the start condition of learned skill clauses. We believe that cases will arise in which such chaining, even if not strictly necessary, will make the acquisition of complex skills much easier. Extending the problem solver to support this ability means defining new conceptual predicates that the agent can use to characterize situations in which its learned skills are applicable. This will be straightforward for some domains and tasks, but some recursive skills will need recursively defined start concepts, which requires a new learning mechanism. Augmenting the system in this manner may also lead to a utility problem (Minton, 1990), not during execution of learned teleoreactive logic programs but during the problem solving used for their acquisition, which we would then need to overcome.

Finally, we should note that, although our approach learns recursive logic programs that generalize to different numbers of objects, its treatment of goals is less flexible. For example, it can acquire a general program for clearing a block that does not depend on the number of other objects involved, but it cannot learn a program for constructing a tower with arbitrarily specified components. Extending the system's ability to transfer across different goals, including ones that are defined recursively, is another important direction for future research on learning hierarchical skills.

8. Concluding Remarks

In the preceding pages, we proposed a new representation of knowledge – teleoreactive logic programs – and described how they can be executed over time to control physical agents. In addition, we explained how a means-ends problem solver can use them to solve novel tasks and, more important, transform the traces of problem solutions into new clauses that can be executed efficiently. The responsible learning method – goal-driven composition – acquires recursive, executable skills in an incremental and cumulative manner. We reported experiments that demonstrated the method's ability to acquire hierarchical and recursive skills for three domains, along with its capacity to transfer its learned structures to tasks with more objects than seen during training.

Teleoreactive logic programs incorporate ideas from a number of traditions, including logic programming, adaptive control, and hierarchical task networks, in a manner that supports reactive but goal-directed behavior. The approach which we have described for acquiring such programs, and which we have incorporated into the ICARUS architecture, borrows intuitions from earlier work on learning through problem solving, but its details rely on a new mechanism that bears little resem-

blance to previous techniques. Our work on learning teleoreactive logic programs is still in its early stages, but it appears to provide a novel and promising path to the acquisition of effective control systems through a combination of reasoning and experience.

Acknowledgements

This material is based on research sponsored by DARPA under agreement numbers HR0011-04-1-0008 and FA8750-05-2-0283 and by Grant IIS-0335353 from the National Science Foundation. The U. S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Discussions with Nima Asgharbeygi, Kirstin Cummings, Glenn Iba, Negin Nejati, David Nicholas, Seth Rogers, and Stephanie Sage contributed to the ideas we have presented in this paper.

References

Bacchus, F. AIPS'00 planning competition. *AI Magazine, 22*, 47–56, 2001.

Benson, S. Induction learning of reactive action models. *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 47–54. San Francisco: Morgan Kaufmann, 1995.

Choi, D., Kaufman, M., Langley, P., Nejati, N., and Shapiro, D. An architecture for persistent reactive behavior. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, pp. 988–995. New York: ACM Press, 2004.

Choi, D., and Langley, P. Learning teleoreactive logic programs from problem solving. *Proceedings of the Fifteenth International Conference on Inductive Logic Programming*, pp. 51–68. Bonn, Germany: Springer, 2005.

Clocksin, W. F., and Mellish, C. S. *Programming in* PROLOG. Berlin: Springer-Verlag, 1981.

Erol, K., Hendler, J., and Nau, D. S. HTN planning: Complexity and expressivity. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 1123–1128. Seattle: MIT Press, 1994.

Estlin, T. A., and Mooney, R. J. Learning to improve both efficiency and quality of planning. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 1227–1232. Nagoya, Japan, 1997.

Fern, A., Yoon, S. W., and Givan, R. Learning domain-specific control knowledge from random walks. *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling*, pp. 191–199. Whistler, BC: AAAI Press, 2004.

Iba, G. A. A heuristic approach to the discovery of macro-operators. *Machine Learning, 3*, 285–317, 1989.

Jones, R. M., and Langley, P. A constrained architecture for learning and problem solving. *Computational Intelligence, 21*, 480–502, 2005.

Khardon, R. Learning action strategies for planning domains. *Artificial Intelligence, 113*, 125–148, 1999.

Laird, J. E., Rosenbloom, P. S., and Newell, A. Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, *1*, 11–46, 1986.

Langley, P., and Rogers, S. Cumulative learning of hierarchical skills. *Proceedings of the Third International Conference on Development and Learning*. San Diego, CA, 2004.

Marsella, S., and Schmidt, C. F. A method for biasing the learning of nonterminal reduction rules. In S. Minton (Ed.), *Machine learning methods for planning*. San Mateo, CA: Morgan Kaufmann, 1993.

Minton, S. N. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363–391, 1990.

Mooney, R. J. The effect of rule use on the utility of explanation-based learning. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 725–730. Detroit: Morgan Kaufmann, 1989.

Nau, D., Cao, Y., Lotem, A., and Muñoz-Avila, H. SHOP: Simple hierarchical ordered planner. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 968–973. Stockholm: Morgan Kaufmann, 1999.

Neches, R., Langley, P., and Klahr, D. Learning, development, and production systems. In D. Klahr, P. Langley, and R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press, 1987.

Newell, A., and Simon, H. A. GPS, A program that simulates human thought. In H. Billing (Ed.), *Lernende automaten*. Munich: Oldenbourg KG. Reprinted in E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought*. New York: McGraw-Hill, 1961.

Nilsson, N. Teleoreactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*, 139–158, 1994.

Olsson, R. Inductive functional programming using incremental program transformation. *Artificial Intelligence*, *74*, 55–83, 1995.

Reddy, C., and Tadepalli, P. Learning goal-decomposition rules using exercises. *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 278–286. San Francisco: Morgan Kaufmann, 1997.

Ruby, D., and Kibler, D. SteppingStone: An empirical and analytical evaluation. *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 527–532. Menlo Park, CA: AAAI Press, 1991.

Sammut, C. Automatic construction of reactive control systems using symbolic machine learning. *Knowledge Engineering Review*, *11*, 27–42, 1996.

Schmid, U. A cognitive model of learning by doing. *Models and human reasoning – Festschrift für Bernd Mahr*. Berlin: Wissenschaft & Technik Verlag, 2005.

Shavlik, J. W. Acquiring recursive concepts with explanation-based learning. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 688–693. Detroit, MI: Morgan Kaufmann, 1989.

Sleeman, D., Langley, P., and Mitchell, T. Learning from solution paths: An approach to the credit assignment problem. *AI Magazine*, *3*, 48–52, 1982.

Sutton, R. S. and Barto, A. G. *Reinforcement learning*. Cambridge, MA: MIT Press, 1998.

Veloso, M. M., and Carbonell, J. G. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, *10*, 249–278, 1993.

Zelle, J. M., and Mooney, R. J. Combining FOIL and EBG to speed up logic programs. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1106–1111. Chambery, France: Morgan Kaufmann, 1993.

Journal of Machine Learning Research 7 (2006) 519–549 Submitted 6/05; Revised 11/05; Published 3/06

Learning Coordinate Covariances via Gradients

Sayan Mukherjee SAYAN@STAT.DUKE.EDU
Institute of Statistics and Decision Sciences
Institute for Genome Sciences and Policy
Department of Computer Science
Duke University
Durham, NC 27708, USA

Ding-Xuan Zhou MAZHOU@CITYU.EDU.HK
Department of Mathematics
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China

Editor: John Shawe-Taylor

Abstract

We introduce an algorithm that learns gradients from samples in the supervised learning framework. An error analysis is given for the convergence of the gradient estimated by the algorithm to the true gradient. The utility of the algorithm for the problem of variable selection as well as determining variable covariance is illustrated on simulated data as well as two gene expression data sets. For square loss we provide a very efficient implementation with respect to both memory and time.

Keywords: Tikhnonov regularization, variable selection, reproducing kernel Hilbert space, generalization bounds

1. Introduction

The advent of data sets with many variables or coordinates in the biological and physical sciences has driven the use of a variety of machine learning approaches based on Tikhonov regularization or global shrinkage such as support vector machines (SVMs) (Vapnik, 1998) and regularized least square classification (Poggio and Girosi, 1990). These algorithms have been very successful in both classification and regression problems. However, in a number of applications the classical questions from statistical linear modelling of which variables are most relevant to the prediction and how the coordinates vary with respect to each other have been revived. In the context of high dimensional data with few examples, the "large p, small n" paradigm (West, 2003), this leads to foundational questions in constructing and interpreting statistical models. Since statistical models based on shrinkage or regularization (Vapnik, 1998; West, 2003) have had success in the framework of both classification and regression, we formulate the problem of learning coordinate covariation and relevance in this framework.

We first describe the Tikhonov regularization method for classification and regression in order to define notation and basic concepts. We then introduce an algorithm that learns gradients of a function. We also motivate the algorithm and give an intuition of how the gradient can be used to learn coordinate covariation and relevance.

1.1 Classification and Regression

Classification and regression problems can be addressed in the framework of learning or estimating functions from a hypothesis space given sample values. An efficient learning method is the Tikhonov regularization scheme. Let X be a compact metric space and the hypothesis space, $\mathcal{H}$, be a set of functions $X \to Y \subset \mathbb{R}$. If we assign a penalty functional $\Omega : \mathcal{H} \to \mathbb{R}_+$ on $\mathcal{H}$ and choose a loss function $V : \mathbb{R}^2 \to \mathbb{R}_+$, the Tikhonov regularization scheme in $\mathcal{H}$ associated with (V,Ω) is defined for a sample $\mathbf{z} = \{(x_i,y_i)\}_{i=1}^m \in (X \times Y)^m$ and $\lambda > 0$ as

$$f_{\mathbf{z}}^V = \arg\min_{f\in\mathcal{H}} \Big\{ \frac{1}{m}\sum_{i=1}^m V(y_i, f(x_i)) + \lambda\Omega(f) \Big\}. \tag{1}$$

The efficiency of learning algorithms of type (1) in machine learning can be seen when $\mathcal{H}$ takes the special choice of a reproducing kernel Hilbert space generated by a Mercer kernel.

Let $K : X \times X \to \mathbb{R}$ be continuous, symmetric and positive semidefinite, i.e., for any finite set of distinct points $\{x_1,\cdots,x_m\} \subset X$, the matrix $(K(x_i,x_j))_{i,j=1}^m$ is positive semidefinite. Such a function is called a *Mercer kernel*.

The *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}_K$ associated with the Mercer kernel K is defined (see Aronszajn (1950)) to be the completion of the linear span of the set of functions $\{K_x := K(x,\cdot) : x \in X\}$ with the inner product $\langle\cdot,\cdot\rangle_K$ satisfying $\langle K_x, K_y\rangle_K = K(x,y)$. The reproducing property of $\mathcal{H}_K$ is

$$\langle K_x, f\rangle_K = f(x), \qquad \forall x \in X, f \in \mathcal{H}_K. \tag{2}$$

If $\mathcal{H} = \mathcal{H}_K$ and $\Omega(f) = \|f\|_K^2$ in (1), the reproducing property (2) tells us that

$$f_{\mathbf{z}}^V = \sum_{i=1}^m c_i K_{x_i}$$

and the coefficients $\{c_i\}_{i=1}^m$ can be found by solving an optimization problem in $\mathbb{R}^m$.

Assume that ρ is a probability distribution on $Z := X \times Y$ and $\mathbf{z} = \{(x_i,y_i)\}_{i=1}^m \in Z^m$ is a random sample independently drawn according to ρ.

When the loss function is the least-square loss $V(y,t) = (y-t)^2$, the algorithm (1) is least-square regression and the objective is to learn the regression function

$$f_\rho(x) = \int_Y y d\rho(y|x), \qquad x \in X \tag{3}$$

from the random sample $\mathbf{z}$. Here $\rho(\cdot|x)$ is the conditional distribution of ρ at x. Denote ρ_X as the marginal distribution of ρ on X and $L^2_{\rho_X}$ as the L^2 space with the metric $\|f\|_\rho := (\int_X |f(x)|^2 d\rho_X)^{1/2}$. There has been a vast literature (e.g. (Evgeniou et al., 2000; Zhang, 2003; Vito et al., 2005; Smale and Zhou, 2006b)) in learning theory showing for this least-square regression algorithm the convergence of $f_{\mathbf{z}}^V$ to f_ρ in the metric $\|\cdot\|_\rho$ under the assumption that f_ρ lies in the closure of $\mathcal{H}_K$ and $\lambda = \lambda(m) \to 0$ as $m \to \infty$.

For the (binary) classification purpose, we take $Y = \{1,-1\}$. A real valued function $f : X \to \mathbb{R}$ induces a classifier $\mathrm{sgn}(f) : X \to Y$. In this case, one uses a (convex) loss function $\phi : \mathbb{R} \to \mathbb{R}_+$ to measure the empirical error $\phi(t)$, $t = yf(x)$, when $\mathrm{sgn}(f(x))$ is applied to predict $y \in Y$. Examples of such a convex loss function ϕ include the logistic loss

$$\phi(t) = \log\big(1+e^{-t}\big), \qquad t \in \mathbb{R} \tag{4}$$

and the hinge loss $\phi(t) = \max\{0, 1-t\}$. For $V(y, f(x)) = \phi(t)$ in (1) extensive investigation in learning theory (e.g. (Cortes and Vapnik, 1995; Evgeniou et al., 2000; Schoelkopf and Smola, 2001; Vapnik, 1998; Wu and Zhou, 2005)) has shown that $\mathrm{sgn}(f_{\mathbf{z}}^V)$ converges to the Bayes rule $\mathrm{sgn}(f_\rho)$ with respect to the misclassification error:

$$\mathcal{R}(\mathrm{sgn}(f)) = \mathrm{Prob}\{\mathrm{sgn}(f(x)) \neq y\}.$$

1.2 Learning the Gradient

In this paper we are interested in learning the gradient of f_ρ from the function sample values. Let $X \subset \mathbb{R}^n$. Denote $x = (x^1, x^2, \ldots, x^n)^T \in \mathbb{R}^n$. The gradient of f_ρ is the vector of functions (if the partial derivatives exist)

$$\nabla f_\rho = \left(\frac{\partial f_\rho}{\partial x^1}, \ldots, \frac{\partial f_\rho}{\partial x^n} \right)^T. \tag{5}$$

The relevance of learning the gradient with respect to the problems of variable selection and estimating coordinate covariation is that the gradient provides the following information:
(a) variable selection: the norm of a partial derivative $\| \frac{\partial f_\rho}{\partial x^i} \|$ indicates the relevance of this variable, since a small norm implies a small change in the function f_ρ with respect to the i-th coordinate,
(b) coordinate covariation: the inner product between partial derivatives $\left\langle \frac{\partial f_\rho}{\partial x^i}, \frac{\partial f_\rho}{\partial x^j} \right\rangle$ indicates the covariance of the i-th and j-th coordinates with respect to variation in f_ρ.

We now motivate the derivation of our gradient learning algorithm. Taylor expanding a function $g(u)$ around the point x gives us

$$g(u) \approx g(x) + \int_{\Delta x \in \Gamma_x} \langle \nabla g, \Delta x \rangle,$$

where the inner product and a neighborhood Γ_x of x are determined according to what is natural for different settings. For example, in the manifold setting we know the marginal ρ_X is concentrated on a manifold $\mathcal{M}$ and it is natural to formulate the following expansion

$$g(u) \approx g(x) + \int_{\Delta x \in \mathcal{M}_x} \langle \nabla_{\mathcal{M}} g, \Delta x \rangle,$$

where $\Delta x \in \mathcal{M}_x$ are points on the manifold around x with respect to the intrinsic distance on the manifold and the inner product is L^2 over the manifold (Belkin and Niyogi, 2004). In the graph setting we are given a sparse sample on the manifold which can be thought of as vertices of a graph and the distance between the points is the weight matrix of the graph. A natural formulation in this setting is to set Γ_x to be vertices connected to x and the inner product as the weight matrix. Minimizing the empirical error (with regularization) between $g(u)$ and its expansion $g(x) + \int_{\Delta x \in \Gamma_x} \langle \nabla g, \Delta x \rangle \approx g(x) + \nabla g(x) \cdot (u - x)$ for $u \approx x$ results in various learning algorithms.

For regression the algorithm to learn gradients will use least-square loss to minimize the error of the Taylor expansion at sample points. To learn vectors of functions we use the hypothesis space $\mathcal{H}_K^n$ which is an n-fold of $\mathcal{H}_K$: each $\vec{f} \in \mathcal{H}_K^n$ can be written as a column vector of functions $\vec{f} = (f_1, f_2, \ldots, f_n)^T$ with $f_\ell \in \mathcal{H}_K$. Define $\langle \vec{f}, \vec{h} \rangle_K = \sum_{\ell=1}^n \langle f_\ell, h_\ell \rangle_K$. Then $\| \vec{f} \|_K^2 = \sum_{\ell=1}^n \| f_\ell \|_K^2$. The empirical error on sample points $x = x_i, u = x_j$ will be measured by the square loss

$$\big(g(u) - g(x) - \nabla g(x) \cdot (u - x) \big)^2 = \big(y_i - y_j + \vec{f}(x_i) \cdot (x_j - x_i) \big)^2.$$

The restriction $u \approx x$ will be enforced by weights: $w_{i,j} = w_{i,j}^{(s)} > 0$ corresponding to (x_i, x_j) with the requirement that $w_{i,j}^{(s)} \to 0$ as $|x_i - x_j|/s \to \infty$. For $x = (x^1, x^2, \ldots, x^n)^T \in \mathbb{R}^n$, we denote $|x| = \left(\sum_{j=1}^n (x^j)^2\right)^{1/2}$.

One possible choice of weights is given by a Gaussian with variance s. Let $w = w_s$ be the function on $\mathbb{R}^n$ given by $w(x) = \frac{1}{s^{n+2}} e^{-\frac{|x|^2}{2s^2}}$. Then this choice of weights is

$$w_{i,j} = w_{i,j}^{(s)} = \frac{1}{s^{n+2}} e^{-\frac{|x_i - x_j|^2}{2s^2}} = w(x_i - x_j), \qquad i, j = 1, \ldots, m. \tag{6}$$

For regression we define the algorithm by the following optimization problem with weights being arbitrary positive numbers $w_{i,j} = w_{i,j}^{(s)}$ which depend on an index $s > 0$.

Definition 1 *The least-square type learning scheme is defined for the sample $\mathbf{z} \in Z^m$ as*

$$\vec{f}_{\mathbf{z},\lambda} := \arg\min_{\vec{f} \in \mathcal{H}_K^n} \left\{ \frac{1}{m^2} \sum_{i,j=1}^m w_{i,j}^{(s)} \left(y_i - y_j + \vec{f}(x_i) \cdot (x_j - x_i) \right)^2 + \lambda \| \vec{f} \|_K^2 \right\}, \tag{7}$$

where λ, s are two positive constants called the regularization parameters.

A similar algorithm can be defined for classification with a convex loss function $\phi(\cdot)$ like the hinge or logistic loss.

Definition 2 *The regularization scheme for classification is defined for the sample $\mathbf{z} \in Z^m$ as*

$$\vec{f}_{\mathbf{z},\lambda} = \arg\min_{\vec{f} \in \mathcal{H}_K^n} \left\{ \frac{1}{m^2} \sum_{i,j=1}^m w_{i,j}^{(s)} \phi\left(y_i \left(y_j + \vec{f}(x_i) \cdot (x_i - x_j) \right) \right) + \lambda \| \vec{f} \|_K^2 \right\}. \tag{8}$$

Remark 3 *Some algorithms for computing numerical derivative by means of partition were introduced in Wahba and Wendelberger (1980). They work well in low dimensional spaces. In high dimensional spaces, partition is difficult. Our method can be regarded as an algorithm for numerical derivatives in high dimensional spaces.*

At first thought, a natural approach to computing partial derivatives would be to estimate the regression function and then compute partial derivatives. The problem with this approach is that the partial derivatives are no longer in the RKHS of the regression function. This leaves us with the problem of not having a norm or computable metric to work with. The advantage of our method is the derived functions are already approximations of the partial derivatives and they have RKHS inner products which are computed in the estimation process. The inner products reflect the nature of the measure, which is often on a low dimensional manifold embedded in a large dimensional space.

The hypothesis space $\mathcal{H}_K^n$ in the optimization problem (7) may be replaced by some other space of vector-valued functions (Micchelli and Pontil, 2005) in order to learn the gradients.

Remark 4 *Estimation of coordinate covariation is not possible in standard regression models that allow for variable selection such as: recursive feature elimination (RFE) (Guyon et al., 2002), least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996), and basis pursuits denoising (Chen et al., 1999).*

1.3 Overview

In Sections 2 and 3, we shall derive linear systems for solving the optimization problem (7). In particular, when $m << n$, an efficient algorithm will be provided.

The regularization parameters in (7) depend on m: $\lambda = \lambda(m)$, $s = s(m)$ and generally $\lambda(m), s(m) \to 0$ as m becomes large. In Section 4, we show for a Gaussian weight function (6) how a particular choice of the two regularization parameters leads to rates of convergence of our estimate of the gradient to the true gradient, $\vec{f}_{\mathbf{z},\lambda}$ to ∇f_ρ.

The utility of the algorithm is demonstrated in Section 5 in applications to simulated data as well as gene expression data. We close with a brief discussion in Section 6.

2. Representer Theorem

The optimization problem defining the least-square algorithm (7) can be solved as a linear system of equations. Denote $\mathbb{R}^{p\times q}$ as the space of $p \times q$ matrices, I_n the $n \times n$ identity matrix, and $\text{diag}\{B_1, B_2, \cdots, B_m\}$ the $m \times m$ block diagonal matrix with each $B_i \in \mathbb{R}^{n\times n}$. To save space, we express an mn column vector with blocks $\{c_i \in \mathbb{R}^n\}$ by the following abuse of notion $c = (c_1, c_2, \ldots, c_m)^T$.

The following theorem is an analog of the standard representer theorem (Schoelkopf and Smola, 2001) that states the minimizer of the optimization problem defined by (7) has the form

$$\vec{f}_{\mathbf{z},\lambda} = \sum_{i=1}^{m} c_{i,\mathbf{z}} K_{x_i} \tag{9}$$

with $c_{\mathbf{z}} = (c_{1,\mathbf{z}}, \ldots, c_{m,\mathbf{z}})^T \in \mathbb{R}^{mn}$.

Theorem 5 *For $i = 1, \ldots, m$, let B_i*

$$B_i = \sum_{j=1}^{m} w_{i,j}(x_j - x_i)(x_j - x_i)^T \in \mathbb{R}^{n\times n}, \quad Y_i = \sum_{j=1}^{m} w_{i,j}(y_j - y_i)(x_j - x_i) \in \mathbb{R}^n. \tag{10}$$

Then $\vec{f}_{\mathbf{z},\lambda} = \sum_{i=1}^{m} c_{i,\mathbf{z}} K_{x_i}$ with $c_{\mathbf{z}} = (c_{1,\mathbf{z}}, \ldots, c_{m,\mathbf{z}})^T \in \mathbb{R}^{mn}$ satisfying the linear system

$$\left\{ m^2\lambda I_{mn} + diag\{B_1, B_2, \cdots, B_m\}\left[K(x_i, x_j) I_n\right]_{i,j=1}^{m} \right\} c = (Y_1, Y_2, \ldots, Y_m)^T. \tag{11}$$

Proof By projecting onto the span of $\{K_{x_i}\}_{i=1}^m$ the reproducing property (2) ensures that $\vec{f}_{\mathbf{z},\lambda} = \sum_{i=1}^m c_{i,\mathbf{z}} K_{x_i}$, with $c_{i,\mathbf{z}} \in \mathbb{R}^n$ for each i. Note that $x \cdot v = \sum_{i=1}^n x^i v^i = x^T v$ for $x, v \in \mathbb{R}^n$. To find $\{c_{i,\mathbf{z}}\}$, we consider $\vec{f} = \sum_{i=1}^m c_i K_{x_i} \in \mathcal{H}_K^n$ with $c_i \in \mathbb{R}^n$. Then

$$\vec{f}(x_i) \cdot (x_j - x_i) = \sum_{p=1}^{m} K(x_p, x_i) c_p \cdot (x_j - x_i) = \sum_{p=1}^{m} K(x_p, x_i)(x_j - x_i)^T c_p$$

and

$$\|\vec{f}\|_K^2 = \sum_{i,j=1}^{m} K(x_i, x_j) c_i \cdot c_j.$$

Define the **empirical error** $\mathcal{E}_{\mathbf{z}}$ as

$$\mathcal{E}_{\mathbf{z}}(\vec{f}) = \frac{1}{m^2} \sum_{i,j=1}^{m} w_{i,j}^{(s)} \left(y_i - y_j + \vec{f}(x_i) \cdot (x_j - x_i) \right)^2.$$

It is a function of mn variables $\{c_q^k : 1 \le q \le m, 1 \le k \le n\}$ where the q-th coefficient $c_q \in \mathbb{R}^n$ of $\vec{f}$ is expressed as $(c_q^k)_{k=1}^n = (c_q^1, \ldots, c_q^n)^T$. For $q \in \{1, \ldots, m\}$, $k \in \{1, \ldots, n\}$,

$$\frac{\partial}{\partial c_q^k}\left\{\mathcal{E}_{\mathbf{z}}(\vec{f}) + \lambda \|\vec{f}\|_K^2\right\} = 2\lambda \sum_{i=1}^m K(x_q, x_i) c_i^k$$
$$+ \frac{2}{m^2} \sum_{i,j=1}^m w_{i,j} \left(y_i - y_j + \sum_{p=1}^m K(x_p, x_i)(x_j - x_i)^T c_p \right) K(x_q, x_i)(x_j^k - x_i^k).$$

Notice from (2) that for $g, h \in \operatorname{span}\{K_{x_i}\}_{i=1}^m$, $g(x_i) - h(x_i) = 0$ for $i = 1, \ldots, m$ implies that $g - h$ is orthogonal to each K_{x_i}, and hence $g - h = 0$. Then we know that $\vec{f}_{\mathbf{z},\lambda} = \sum_{i=1}^m \tilde{c}_{i,\mathbf{z}} K_{x_i}$ where $\tilde{c}_{\mathbf{z}} = \{\tilde{c}_{i,\mathbf{z}}\}_{i=1}^m$ is the solution to the linear system

$$\lambda c_i + \frac{1}{m^2} \sum_{j=1}^m w_{i,j} \left(y_i - y_j + \sum_{p=1}^m K(x_p, x_i)(x_j - x_i)^T c_p \right) (x_j - x_i) = 0, \; i = 1, \ldots, m.$$

Since $(x_j - x_i)^T c_p$ is a scalar, $[(x_j - x_i)^T c_p](x_j - x_i) = (x_j - x_i)(x_j - x_i)^T c_p$. So the above system can be expressed as

$$B_i \sum_{p=1}^m K(x_i, x_p) c_p + m^2 \lambda c_i = Y_i, \; i = 1, \ldots, m. \tag{12}$$

This is exactly the system in (11). ■

Remark 6 *One might consider solving the optimization problem (7) by finding each component of $\vec{f}_{\mathbf{z},\lambda}$ separately. However, to find $(\vec{f}_{\mathbf{z},\lambda})_\ell$ by minimizing over $f \in \mathcal{H}_K$, one needs to replace $y_i - y_j$ by $y_i - y_j + \sum_{k \ne \ell} (\vec{f}_{\mathbf{z},\lambda})_k (x_i)(x_j^k - x_i^k)$. So the optimization problems for components of $\vec{f}_{\mathbf{z},\lambda}$ are not completely separable. It would be interesting to have a separable method for (7).*

3. Reducing the Matrix Size

In some applications of variable selection, the number n of variables is much larger than the sample size m. In such a situation, the system (11) for implementing the learning algorithm (7) is not satisfactory, since the size of the linear system (11) is $(mn) \times (mn)$.

Observe that each term in the summation defining B_i in (10) is a rank one matrix. Hence the rank of the $n \times n$ matrix B_i is at most m for each i. This raises the expectation of reducing the matrix size in the linear system (11). In this section, we show how to reduce this size to $(\mathcal{S}m) \times (\mathcal{S}m)$ with $\mathcal{S} \le m - 1$. Moreover, an approximation algorithm will be introduced which is often implemented with $\mathcal{S} << m$.

We use the well known approach of singular value decomposition. It may be applied to the coefficient matrix of (11) to reduce the matrix size. Here we prefer to apply the approach to a matrix involving the data only, leaving us flexibility for the weights $w_{i,j}$.

Consider the matrix involving the data $\mathbf{x}$ given by

$$M_{\mathbf{x}} = [x_1 - x_m, x_2 - x_m, \ldots, x_{m-1} - x_m, x_m - x_m] \in \mathbb{R}^{n \times m}. \tag{13}$$

Assume the rank of $M_{\mathbf{x}}$ is d. Then $d \le \min\{m - 1, n\}$ since the last column of the matrix is zero. The theory of singular value decomposition tells us that there exists an $n \times n$ orthogonal matrix

$V = [V_1, V_2, \ldots, V_n]$ and a $m \times m$ orthogonal matrix $U = [U_1, U_2, \ldots, U_m]$ such that

$$M_{\mathbf{x}} = V\Sigma U^T = [V_1\ V_2\ \cdots\ V_n] \begin{bmatrix} \mathrm{diag}\{\sigma_1, \sigma_2, \cdots, \sigma_d\} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \\ \vdots \\ U_m^T \end{bmatrix}. \tag{14}$$

Here $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > \sigma_{d+1} = \ldots = \sigma_{\min\{m,n\}} = 0$ are the singular values of $M_{\mathbf{x}}$. The matrix Σ is $n \times m$ and has entries zero except that $(\Sigma)_{i,i} = \sigma_i$ for $i = 1, \ldots, d$. From expression (14), we see that

$$M_{\mathbf{x}} = \sum_{\ell=1}^{d} \sigma_\ell V_\ell U_\ell^T.$$

Note that $U_\ell^T = [U_\ell^1, \ldots, U_\ell^m]$. The j-th column of $M_{\mathbf{x}}$ equals $x_j - x_m = \sum_{\ell=1}^{d} \sigma_\ell V_\ell U_\ell^j$ and

$$x_j - x_i = \sum_{\ell=1}^{d} \sigma_\ell \left(U_\ell^j - U_\ell^i\right) V_\ell. \tag{15}$$

It follows that $Y_i = \sum_{j=1}^{m} w_{i,j}(y_j - y_i) \sum_{\ell=1}^{d} \sigma_\ell \left(U_\ell^j - U_\ell^i\right) V_\ell$ and

$$B_i = \sum_{j=1}^{m} w_{i,j} \sum_{\ell=1}^{d} \sum_{p=1}^{d} \sigma_\ell \sigma_p \left(U_\ell^j - U_\ell^i\right) \left(U_p^j - U_p^i\right) V_\ell V_p^T. \tag{16}$$

Now we can reduce the matrix size by solving an approximation to the linear system derived from the singular values. A strong correlation among the vectors $\{x_i\}$ would result in a large number of small singular values. If we ignore the small singular values $\sigma_{S+1}, \ldots, \sigma_d$, the error is proportional to σ_{S+1}. This follows from the idea of low-rank approximations in singular value decomposition. The following theorem quantifies the above statement.

Theorem 7 *Assume $|y| \leq M$ almost surely. Denote $\kappa = \sup_{x \in X} \sqrt{K(x,x)}$. Let $1 \leq S \leq d$. Set*

$$\mathcal{B}_i = \sum_{j=1}^{m} w_{i,j} \left[\sigma_\ell \sigma_p \left(U_\ell^j - U_\ell^i\right) \left(U_p^j - U_p^i\right)\right]_{\ell,p=1}^{S} \in \mathbb{R}^{S \times S}, \qquad i = 1, \ldots, m \tag{17}$$

and

$$\mathcal{Y}_i = \sum_{j=1}^{m} w_{i,j}(y_j - y_i) \left[\sigma_\ell \left(U_\ell^j - U_\ell^i\right)\right]_{\ell=1}^{S} \in \mathbb{R}^{S}, \qquad i = 1, \ldots, m. \tag{18}$$

Solve the linear system

$$\left\{m^2 \lambda I_{mS} + diag\{\mathcal{B}_1, \cdots, \mathcal{B}_m\} \left[K(x_i, x_j) I_S\right]_{i,j=1}^{m}\right\} \hat{b} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_m)^T. \tag{19}$$

The solution $\hat{b}_{\mathbf{z}} = (\hat{b}_{1,\mathbf{z}}, \ldots, \hat{b}_{m,\mathbf{z}})^T \in \mathbb{R}^{mS}$ gives an approximation $\vec{f}_{\mathbf{z},\lambda,S} = \sum_{i=1}^{m} b_{i,\mathbf{z}} K_{x_i}$ with $b_{i,\mathbf{z}} = \sum_{\ell=1}^{S} \hat{b}_{i,\mathbf{z}}^{\ell} V_\ell$. The error between $b_{\mathbf{z}} = (b_{i,\mathbf{z}})_{i=1}^{m}$ and $c_{\mathbf{z}}$ can be bounded as

$$\left|b_{\mathbf{z}} - c_{\mathbf{z}}\right|_{\ell^2(\mathbb{R}^{mn})} \leq \frac{4M\sigma_{S+1}}{m^2\lambda} \left\{\sqrt{d\Delta_m} + \frac{2\kappa^2\sigma_1^2}{m\lambda}\sqrt{S}\Delta_m\right\}, \tag{20}$$

where $\Delta_m := \max_{1 \leq i \leq m} \left(\sum_{j=1}^{m} w_{i,j}\right)^2 + \sum_{i,j=1}^{m} \left(w_{i,j}\right)^2$.

See Appendix B for the proof.

If we set $\mathcal{S} = d$, we can solve for $\vec{f}_{\mathbf{z},\lambda}$ exactly with a linear system of reduced size. This is stated in the following corollary.

Corollary 8 *Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0$ be all the positive singular values of $M_{\mathbf{x}}$ and U, V be the orthogonal matrices in (14). Then $\vec{f}_{\mathbf{z},\lambda} = \sum_{i=1}^{m} \{\sum_{\ell=1}^{d} \widetilde{c}_{i,\mathbf{z}}^{\ell} V_\ell\} K_{x_i}$ where $\widetilde{c}_{\mathbf{z}}$ satisfies (19) with $\mathcal{B}_i$ and $\mathcal{Y}_i$ given by (17) and (18) with $\mathcal{S} = d$, respectively.*

Theorem 7 provides a theoretical foundation for the following approximation algorithm with reduced matrix size that accurately approximates $\vec{f}_{\mathbf{z},\lambda}$ by $\vec{f}_{\mathbf{z},\lambda,\mathcal{S}}$.

3.1 Reduced Matrix Size Algorithm

The following is an outline of the reduced matrix algorithm. See appendix C for Matlab® code implementing this algorithm.

Algorithm 1: Approximation algorithm with reduced matrix size to approximate $\vec{f}_{\mathbf{z},\lambda}$

---

input : inputs $(x_i)_{i=1}^m$, labels $(y_i)_{i=1}^m$, kernel K, weights $(w_{i,j})$, eigenvalue threshold $\varepsilon > 0$

return: coefficients $(b_{i,\mathbf{z}})_{i=1}^m$

$M_{\mathbf{x}} = [x_1 - x_m, x_2 - x_m, \ldots, x_{m-1} - x_m, x_m - x_m] \in \mathbb{R}^{n \times m}$;
Given $M_{\mathbf{x}}$ compute the singular value decomposition (14) with orthogonal matrices U, V and singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\mathcal{S}} > \varepsilon$;
$t_j = \left(\sigma_1 U_1^j, \ldots, \sigma_{\mathcal{S}} U_{\mathcal{S}}^j\right)^T \in \mathbb{R}^{\mathcal{S}}$ for $1 \leq j \leq m$;
$\mathcal{B}_i = \sum_{j=1}^{m} w_{i,j}(t_j - t_i)(t_j - t_i)^T$ for $1 \leq i \leq m$;
$\mathcal{Y}_i = \sum_{j=1}^{m} w_{i,j}(y_j - y_i)(t_j - t_i)$ for $1 \leq i \leq m$;

$$\left[\widetilde{K}\right] = \begin{bmatrix} \mathcal{B}_1 K(x_1,x_1) & \mathcal{B}_1 K(x_1,x_2) & \cdots & \mathcal{B}_1 K(x_1,x_m) \\ \mathcal{B}_2 K(x_2,x_1) & \mathcal{B}_2 K(x_2,x_2) & \cdots & \mathcal{B}_2 K(x_2,x_m) \\ \vdots & \ddots & \cdots & \vdots \\ \mathcal{B}_m K(x_m,x_1) & \mathcal{B}_m K(x_m,x_2) & \cdots & \mathcal{B}_m K(x_m,x_m) \end{bmatrix}, \qquad \vec{\mathcal{Y}} = \begin{bmatrix} \mathcal{Y}_1 \\ \mathcal{Y}_2 \\ \vdots \\ \mathcal{Y}_m \end{bmatrix}$$

$\hat{b}_{\mathbf{z}} = (\hat{b}_{1,\mathbf{z}}, \ldots, \hat{b}_{m,\mathbf{z}})^T \in \mathbb{R}^{m\mathcal{S}}$ where

$$\left\{ m^2 \lambda I_{m\mathcal{S}} + \left[\widetilde{K}\right] \right\} \hat{b}_{\mathbf{z}} = \vec{\mathcal{Y}} \tag{21}$$

$b_{i,\mathbf{z}} = \sum_{\ell=1}^{\mathcal{S}} \hat{b}_{i,\mathbf{z}}^{\ell} V_\ell$ and $\vec{f}_{\mathbf{z},\lambda,\mathcal{S}} = \sum_{i=1}^{m} b_{i,\mathbf{z}} K_{x_i}$ is an approximation of $\vec{f}_{\mathbf{z},\lambda}$;
return $(b_{i,\mathbf{z}})_{i=1}^m$

---

4. Error Analysis

In what follows we use Gaussian weights (equation (6)) and estimate error bounds. We show that $\vec{f}_{\mathbf{z},\lambda} \to \nabla f_\rho$ as $m \to \infty$ for suitable choices of the regularization parameters going to zero, $\lambda = \lambda(m) \to 0, s = s(m) \to 0$. Since we are learning gradients, some regularity conditions on both the marginal distribution and the density are required. The following case illustrates the idea (this case corresponds to the realizable setting in the PAC learning paradigm and will be a corollary of the error analysis that follows).

Proposition 9 *Assume $|y| \le M$ almost surely. Suppose that for some $0 < \tau \le 2/3, c_\rho > 0$, the marginal distribution ρ_X satisfies*

$$\rho_X\left(\{x \in X : \inf_{u \in \mathbb{R}^n \setminus X} |u - x| \le s\}\right) \le c_\rho^2 s^{4\tau}, \qquad \forall s > 0, \tag{22}$$

and the density $p(x)$ of $d\rho_X(x)$ exists and satisfies

$$\sup_{x \in X} p(x) \le c_\rho, \quad |p(x) - p(v)| \le c_\rho |v - x|^\tau, \qquad \forall v, x \in X. \tag{23}$$

Choose $\lambda = \lambda(m) = m^{-\frac{\tau}{n+2+3\tau}}$ and $s = s(m) = (\kappa c_\rho)^{\frac{2}{\tau}} m^{-\frac{1}{n+2+3\tau}}$. If $\nabla f_\rho \in \mathcal{H}_K^n$ and the kernel K is C^3, then there is a constant $C_{\rho,K}$ such that for any $0 < \delta < 1$ and $m \ge 1$, with confidence $1 - \delta$, we have

$$\|\vec{f}_{\mathbf{z},\lambda} - \nabla f_\rho\|_\rho \le C_{\rho,K} \log\left(\frac{2}{\delta}\right)\left(\frac{1}{m}\right)^{-\frac{\tau}{2(n+2+3\tau)}}. \tag{24}$$

The condition (23) means the density of the marginal distribution is Hölder τ. The condition (22) is about the behavior of ρ_X near the boundary of X. They are natural assumptions for learning gradients of the regression function. When the boundary is piecewise smooth, (23) implies (22).

The idea behind the proof for the convergence of the gradient consists of simultaneously controlling a sample or estimation error term and a regularization or approximation error term. The first term, the sample error, is bounded using a concentration inequality since it is a function of the sample, $\mathbf{z}$. The second term, the regularization error, does not depend on the sample and we use functional analysis to bound this quantity.

4.1 Sample Error

First we estimate the sample error by means of the sampling operator introduced in Smale and Zhou (2004, 2006b,a).

Definition 10 *The* **sampling operator** *$S_\mathbf{x} : \mathcal{H}_K^n \to \mathbb{R}^{mn}$ associated with a discrete subset $\mathbf{x} = \{x_i\}_{i=1}^m$ of X is defined by*

$$S_\mathbf{x}(\vec{f}) = \left(\vec{f}(x_i)\right)_{i=1}^m = \left(\vec{f}(x_1), \ldots, \vec{f}(x_m)\right)^T.$$

The adjoint of the sampling operator, $S_\mathbf{x}^T : \mathbb{R}^{mn} \to \mathcal{H}_K^n$, is given by

$$S_\mathbf{x}^T c = \sum_{i=1}^m c_i K_{x_i}, \qquad c = (c_i)_{i=1}^m = (c_1, \ldots, c_m)^T \in \mathbb{R}^{mn}.$$

Denote $D_{\mathbf{x}} = \text{diag}\{B_1, B_2, \cdots, B_m\}$ and $\vec{Y} = (Y_1, Y_2, \ldots, Y_m)^T$.

Consider equation (12) satisfied by $c_{\mathbf{z}}$. The quantity $\sum_{p=1}^m K(x_i, x_p) c_{p,\mathbf{z}}$ equals $\vec{f}_{\mathbf{z},\lambda}(x_i)$. Then (12) implies $\left(S_{\mathbf{x}}^T D_{\mathbf{x}} S_{\mathbf{x}} + m^2 \lambda I\right) \vec{f}_{\mathbf{z},\lambda} = S_{\mathbf{x}}^T \vec{Y}$. Therefore,

$$\vec{f}_{\mathbf{z},\lambda} = \left(\frac{1}{m^2} S_{\mathbf{x}}^T D_{\mathbf{x}} S_{\mathbf{x}} + \lambda I\right)^{-1} \frac{1}{m^2} S_{\mathbf{x}}^T \vec{Y}. \tag{25}$$

We introduce an s-generalization error corresponding to the empirical error as follows.

Definition 11 *The s-**generalization error** $\mathcal{E} = \mathcal{E}_s$ is defined for vectors of functions as*

$$\mathcal{E}(\vec{f}) = \int_Z \int_Z w(x-u) \left(y - v + \vec{f}(x) \cdot (u-x)\right)^2 d\rho(x,y) d\rho(u,v).$$

If we denote $\sigma_s^2 = \int_Z \int_Z w(x-u)(y - f_\rho(x))^2 \, d\rho(x,y) d\rho(u,v)$, then

$$\mathcal{E}(\vec{f}) = 2\sigma_s^2 + \int_X \int_X w(x-u) \left[f_\rho(x) - f_\rho(u) + \vec{f}(x) \cdot (u-x)\right]^2 d\rho_X(x) d\rho_X(u). \tag{26}$$

A data independent limit of $\vec{f}_{\mathbf{z},\lambda}$ is

$$\vec{f}_\lambda = \arg \min_{\vec{f} \in \mathcal{H}_K^n} \left\{\mathcal{E}(\vec{f}) + \lambda \|\vec{f}\|_K^2\right\}. \tag{27}$$

Taking the functional derivatives, we know from (26) that $\vec{f}_\lambda$ can be expressed in terms of the following integral operator on the space $\left(L^2_{\rho_X}\right)^n$ with norm $\|\vec{f}\|_\rho = \left(\|f_\ell\|_\rho^2\right)^{1/2}$.

Proposition 12 *Let $L_{K,s} : \left(L^2_{\rho_X}\right)^n \to \left(L^2_{\rho_X}\right)^n$ be the integral operator defined by*

$$L_{K,s} \vec{f} = \int_X \int_X w(x-u)(u-x) K_x (u-x)^T \vec{f}(x) \, d\rho_X(x) d\rho_X(u). \tag{28}$$

It is a positive operator on $\left(L^2_{\rho_X}\right)^n$ and

$$\vec{f}_\lambda = \left(L_{K,s} + \lambda I\right)^{-1} \vec{f}_{\rho,s}. \tag{29}$$

where

$$\vec{f}_{\rho,s} := \int_X \int_X w(x-u)(u-x) K_x \left(f_\rho(u) - f_\rho(x)\right) d\rho_X(x) d\rho_X(u). \tag{30}$$

The operator $L_{K,s}$ has its range in $\mathcal{H}_K^n$. It can also be regarded as a positive operator on $\mathcal{H}_K^n$. We shall use the same notion for the operators on these two different domains.

To bound the sample error $\vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda$, we shall introduce a McDiarmid-Bernstein type probability inequality for vector-valued random variables.

Proposition 13 *Let $\mathbf{z} = \{z_i\}_{i=1}^m$ be i.i.d. draws from a probability distribution ρ on Z, $(H, \|\cdot\|)$ be a Hilbert space, and $F : Z^m \to H$ be measurable. If there is $\widetilde{M} \geq 0$ such that $\|F(\mathbf{z}) - E_{z_i}(F(\mathbf{z}))\| \leq \widetilde{M}$ for each $1 \leq i \leq m$ and almost every $\mathbf{z} \in Z^m$, then for every $\varepsilon > 0$,*

$$Prob_{\mathbf{z}\in Z^m}\{\|F(\mathbf{z}) - E_{\mathbf{z}}(F(\mathbf{z}))\| \geq \varepsilon\} \leq 2\exp\left\{-\frac{\varepsilon^2}{2(\widetilde{M}\varepsilon + \sigma^2)}\right\}, \tag{31}$$

where $\sigma^2 := \sum_{i=1}^m \sup_{\mathbf{z}\setminus\{z_i\}\in Z^{m-1}} E_{z_i}\{\|F(\mathbf{z}) - E_{z_i}(F(\mathbf{z}))\|^2\}$. For any $0 < \delta < 1$, with confidence $1-\delta$, there holds

$$\|F(\mathbf{z}) - E_{\mathbf{z}}(F(\mathbf{z}))\| \leq 2\log\frac{2}{\delta}\{\widetilde{M} + \sqrt{\sigma^2}\}.$$

Proof Apply Theorem 3.3 of Pinelis (1994) (see Appendix A) to the finite sequence $\{f_j = E_{z_m,\ldots,z_j}F - E_{z_m,\ldots,z_1}F : j = 1,2,\ldots,m+1\}$. So $f_1 = 0$ and $f_{m+1} = F(\mathbf{z}) - E_{\mathbf{z}}(F(\mathbf{z}))$. Note that $\|f_j - f_{j-1}\| \leq \widetilde{M}$ almost surely and $\sum_{j=2}^{m+1} E_{z_{j-1}}\|f_j - f_{j-1}\|^2 \leq \sigma^2$. The conditions of Theorem 3.3 of Pinelis (1994) hold with $B^2 = \sigma^2, \Gamma = \widetilde{M}$. As pointed out in a correction of Pinelis (1999), the probability should be $2\exp\{-\frac{r^2}{r\Gamma + B^2 + B\sqrt{B^2+2r\Gamma}}\}$, which is bounded by $2\exp\{-\frac{r^2}{2(r\Gamma+B^2)}\}$. Inequality (31) follows from the theorem.

Choose ε such that $\frac{\varepsilon^2}{2\widetilde{M}\varepsilon + 2\sigma^2} = \log\frac{2}{\delta}$. That is, ε satisfies

$$\varepsilon^2 = 2\widetilde{M}\log\frac{2}{\delta}\varepsilon + 2\sigma^2\log\frac{2}{\delta}.$$

Therefore, with confidence at least $1-\delta$, we have

$$\|F(\mathbf{z}) - E_{\mathbf{z}}(F(\mathbf{z}))\| \leq \varepsilon \leq 2\widetilde{M}\log\frac{2}{\delta} + \sqrt{2\sigma^2\log\frac{2}{\delta}} \leq 2\log\frac{2}{\delta}\{\widetilde{M} + \sqrt{\sigma^2}\}.$$

This proves the proposition. ■

Now we can give the main result on the sample error $\|\vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda\|_K$. Denote the diameter of X as $\mathrm{Diam}(X) = \max_{x,t\in X}|x-t|$ and the moments of the Gaussian as

$$J_p := \int_{\mathbb{R}^n} e^{-\frac{|x|^2}{2}}|x|^p dx, \qquad p \geq 0.$$

In the following sample error estimates, the bounds are valid for any m, λ, and s, though they yield reasonable learning rates only for suitable choices of $\lambda = \lambda(m)$ and $s = s(m)$ which we state in Proposition 9.

Theorem 14 *Assume $|y| \leq M$ almost surely.*

1. *For any $0 < \delta < 1$, with confidence $1-\delta$, we have*

$$\|\vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda\|_K \leq \frac{16\kappa Diam(X)\log(2/\delta)}{\sqrt{m}\lambda s^{n+2}}\left\{2M + \kappa Diam(X)\|\vec{f}_\lambda\|_K\right\} + \frac{1}{m}\|\vec{f}_\lambda\|_K. \tag{32}$$

2. *If the density $p(x)$ of $d\rho_X(x)$ exists and satisfies $\sup_{x\in X} p(x) \leq c_\rho$, then for any $0 < s \leq 1$, with confidence $1-\delta$, there holds*

$$\begin{aligned}\|\vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda\|_K \quad \leq \quad & \frac{8\kappa\log(2/\delta)}{\sqrt{m}\lambda s^{1+n/2}}\left(\sqrt{c_\rho} + \frac{Diam(X)}{\sqrt{m}s^{1+n/2}}\right) \\ & \left(2M\sqrt{J_2} + \kappa(Diam(X) + \sqrt{J_4})\|\vec{f}_\lambda\|_K\right) + \frac{1}{m}\|\vec{f}_\lambda\|_K.\end{aligned} \tag{33}$$

Proof By (25), we have

$$\vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda = \big(\frac{1}{m^2} S_{\mathbf{x}}^T D_{\mathbf{x}} S_{\mathbf{x}} + \lambda I\big)^{-1} \bigg\{ \frac{1}{m^2} S_{\mathbf{x}}^T \vec{Y} - \frac{1}{m^2} S_{\mathbf{x}}^T D_{\mathbf{x}} S_{\mathbf{x}} \vec{f}_\lambda - \lambda \vec{f}_\lambda \bigg\}.$$

Define a vector-valued function $F : Z^m \to \mathcal{H}_K^n$ by

$$F(\mathbf{z}) = \frac{1}{m^2} S_{\mathbf{x}}^T \vec{Y} - \frac{1}{m^2} S_{\mathbf{x}}^T D_{\mathbf{x}} S_{\mathbf{x}} \vec{f}_\lambda.$$

That is,

$$F(\mathbf{z}) = \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m w_{i,j}(x_j - x_i) K_{x_i}(y_j - y_i) - \frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m w_{i,j}(x_j - x_i) K_{x_i}(x_j - x_i)^T \vec{f}_\lambda(x_i).$$

By independence, the expected value of $F(\mathbf{z})$ equals

$$\begin{aligned} & \frac{1}{m^2} \sum_{i=1}^m E_{\mathbf{z}_i} \sum_{j \neq i} E_{\mathbf{z}_j} \bigg\{ w_{i,j}(x_j - x_i) K_{x_i} \big[(y_j - y_i) - (x_j - x_i)^T \vec{f}_\lambda(x_i) \big] \bigg\} \\ = \quad & \frac{m-1}{m^2} \sum_{i=1}^m E_{\mathbf{z}_i} \bigg\{ \int_X w(x_i - u)(u - x_i) K_{x_i} \big[(f_\rho(u) - y_i) - (u - x_i)^T \vec{f}_\lambda(x_i) \big] d\rho_X(u) \bigg\}. \end{aligned}$$

It follows that

$$E_{\mathbf{z}}(F(\mathbf{z})) = \frac{m-1}{m} \vec{f}_{\rho,s} - \frac{m-1}{m} L_{K,s} \vec{f}_\lambda.$$

By (29), $L_{K,s} \vec{f}_\lambda + \lambda \vec{f}_\lambda = \vec{f}_{\rho,s}$. Hence $\lambda \vec{f}_\lambda = \vec{f}_{\rho,s} - L_{K,s} \vec{f}_\lambda = \frac{m}{m-1} E_{\mathbf{z}}(F(\mathbf{z}))$. Therefore,

$$\| \vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda \|_K \leq \frac{1}{\lambda} \| F(\mathbf{z}) - \frac{m}{m-1} E_{\mathbf{z}}(F(\mathbf{z})) \|_K \leq \frac{1}{\lambda} \| F(\mathbf{z}) - E_{\mathbf{z}}(F(\mathbf{z})) \|_K + \frac{1}{m} \| \vec{f}_\lambda \|_K.$$

The reproducing property (2) together with the upper bound κ implies

$$\| f \|_\rho \leq \| f \|_\infty \leq \kappa \| f \|_K, \qquad \forall f \in \mathcal{H}_K. \tag{34}$$

Then we apply Proposition 13 to the function $F(\mathbf{z})$ to get our error bound.

Let $i \in \{1, \ldots, m\}$. We know that $F(\mathbf{z}) - E_{z_i}(F(\mathbf{z}))$ equals

$$\begin{aligned} & \frac{1}{m^2} \sum_{j \neq i} w(x_i - x_j)(x_j - x_i) \bigg\{ K_{x_i} \big[y_j - y_i - (x_j - x_i)^T \vec{f}_\lambda(x_i) \big] \\ & \qquad + K_{x_j} \big[y_j - y_i - (x_j - x_i)^T \vec{f}_\lambda(x_j) \big] \bigg\} \\ & - \frac{1}{m^2} \sum_{j \neq i} \int_X w(x - x_j)(x_j - x) \bigg\{ K_x \big[y_j - f_\rho(x) - (x_j - x)^T \vec{f}_\lambda(x) \big] \\ & \qquad + K_{x_j} \big[y_j - f_\rho(x) - (x_j - x)^T \vec{f}_\lambda(x_j) \big] \bigg\} d\rho_X(x). \end{aligned}$$

Using (34) for $\vec{f}_\lambda$ and $|x - x_j| \leq \mathrm{Diam}(X)$ for any $x \in X$, we see that

$$\| F(\mathbf{z}) - E_{z_i}(F(\mathbf{z})) \|_K \leq \widetilde{M} = \frac{4\kappa \mathrm{Diam}(X)}{m s^{n+2}} \bigg\{ 2M + \kappa \mathrm{Diam}(X) \| \vec{f}_\lambda \|_K \bigg\}.$$

1. We first prove (32). Apply the trivial bound $\sigma^2 \le m\widetilde{M}^2$. Then Proposition 13 tells us that for any $0<\delta<1$, with confidence $1-\delta$, there holds

$$\|F(\mathbf{z})-E_{\mathbf{z}}(F(\mathbf{z})\|_K \le 2\log\frac{2}{\delta}\{\widetilde{M}+\sqrt{m}\widetilde{M}\} \le 4\log\frac{2}{\delta}\sqrt{m}\widetilde{M}.$$

This proves (32).

2. To prove (33) we need to improve on our estimate of the variance σ^2, we bound $\|F(\mathbf{z})-E_{z_i}(F(\mathbf{z}))\|_K$ by

$$\frac{1}{m^2}\sum_{j\neq i} w(x_i-x_j)|x_j-x_i|\left\{2\kappa 2M+2|x_j-x_i|\kappa^2\|\vec{f}_\lambda\|_K\right\}$$
$$+\frac{1}{m^2}\sum_{j\neq i}\int_X w(x-x_j)|x_j-x|\left\{2\kappa 2M+2|x_j-x|\kappa^2\|\vec{f}_\lambda\|_K\right\}d\rho_X(x).$$

It follows that $\left(E_{z_i}(\|F(\mathbf{z})-E_{z_i}(F(\mathbf{z}))\|_K^2)\right)^{1/2}$ is bounded by

$$\frac{2}{m^2}\sum_{j\neq i}\left\{\int_X\left(w(x-x_j)\right)^2|x_j-x|^2\left\{4\kappa M+2|x_j-x|\kappa^2\|\vec{f}_\lambda\|_K\right\}^2 d\rho_X(x)\right\}^{1/2}$$
$$\le \frac{2}{m^2}\sum_{j\neq i}\left\{\int_X s^{-2(n+2)}e^{-\frac{|x-x_j|^2}{s^2}}|x_j-x|^2\left\{4\kappa M\right\}^2 c_\rho dx\right\}^{1/2}$$
$$+\frac{2}{m^2}\sum_{j\neq i}\left\{\int_X s^{-2(n+2)}e^{-\frac{|x-x_j|^2}{s^2}}|x_j-x|^4(2\kappa^2)^2\|\vec{f}_\lambda\|_K^2 c_\rho dx\right\}^{1/2}.$$

Here we have used the assumption $d\rho_X(x)=p(x)dx$ with $p(x)\le c_\rho$. Bounding the above integrals by those on the whole space $\mathbb{R}^n$, we see from the definition of the moments M_r that

$$E_{z_i}(\|F(\mathbf{z})-E_{z_i}(F(\mathbf{z}))\|_K^2) \le \frac{2(m-1)}{m^2}\left\{4\kappa M\sqrt{c_\rho}\sqrt{\frac{J_2}{s^{n+2}2^{1+n/2}}}+2\kappa^2\|\vec{f}_\lambda\|_K\sqrt{c_\rho}\sqrt{\frac{J_4}{s^n 2^{2+n/2}}}\right\}.$$

It follows then that for $s\le 1$

$$\sigma^2 \le \frac{16c_\rho\kappa^2}{ms^{n+2}}\left\{2^{1/4}M\sqrt{J_2}+\kappa\|\vec{f}_\lambda\|_K\sqrt{J_4}\right\}^2.$$

The second statement (inequality (33)) follows from Proposition 13. ■

4.2 Regularization Error

In this subsection, we shall bound the regularization error $\|\vec{f}_\lambda-\nabla f_\rho\|$ by a functional analysis approach. To illustrate the idea, we state the result for a special case when $\nabla f_\rho \in \mathcal{H}_K^n$. It is a corollary of Theorem 17 and Theorem 19 with $r=1/2$.

Proposition 15 *Assume (22) and (23). Denote $V_\rho=\int_X(p(x))^2dx>0$. Suppose that $\nabla f_\rho\in\mathcal{H}_K^n$ and for some $c'_\rho>0$,*

$$|f_\rho(u)-f_\rho(x)-\nabla f_\rho(x)\cdot(u-x)| \le c'_\rho|u-x|^2,\ \forall\, u,x\in X. \tag{35}$$

Then for any $\lambda > 0$ *and* $0 < s \le \min\{\{2\kappa^2 c_\rho (M_{2+\tau} + J_4 + c_\rho J_2)\}^{1/\tau} \lambda^{1/\tau}, 1\}$, *there holds*

$$\|\vec{f}_\lambda - \nabla f_\rho\|_\rho \le (\kappa^2 c'_\rho J_3)\frac{s}{\lambda} + \{2(V_\rho n(2\pi)^{n/2})^{-1/2}\|\nabla f_\rho\|_K\}\sqrt{\lambda}.$$

To estimate the regularization error, we need to consider the convergence of $L_{k,s}$ as $s \to 0$.

Lemma 16 *Assume that for some* $0 < \tau < 1$, *conditions (22) and (23) hold. Then* $V_\rho \le c_\rho$ *and for any* $0 < s \le 1$ *we have*

$$\|L_{K,s} - V_\rho n(2\pi)^{n/2} L_K\|_{\mathcal{H}_K^n \to \mathcal{H}_K^n} \le s^\tau \kappa^2 c_\rho (M_{2+\tau} + J_4 + c_\rho J_2), \tag{36}$$

where L_K *is a positive operator on* $\mathcal{H}_K^n$ *defined by*

$$L_K \vec{f} = \int_X K_x \vec{f}(x) \frac{p(x)}{V_\rho} d\rho_X(x). \tag{37}$$

The operator L_K *is also a positive operator on* $(L^2_{\rho_X})^n$ *satisfying*

$$\|L_{K,s} - V_\rho n(2\pi)^{n/2} L_K\|_{(L^2_{\rho_X})^n \to (L^2_{\rho_X})^n} \le s^\tau \kappa^2 c_\rho (M_{2+\tau} + J_4 + c_\rho J_2), \qquad \forall\, 0 < s \le 1. \tag{38}$$

Proof Let $\vec{f} \in (L^2_{\rho_X})^2$. Denote

$$\vec{g} = \int_X \left\{ \int_X w(x-u)(u-x) K_x (u-x)^T du \right\} p(x) \vec{f}(x)\, d\rho_X(x).$$

Then by (23) and the Cauchy-Schwartz inequality we see that $\|L_{K,s}\vec{f} - \vec{g}\|_K$ is bounded by

$$\int_X \left\{ \int_X \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left|\frac{u-x}{s}\right|^2 \|K_x\|_K c_\rho |u-x|^\tau du \right\} |\vec{f}(x)|\, d\rho_X(x) \le s^\tau \kappa c_\rho M_{2+\tau} \|\vec{f}\|_\rho.$$

Observe that $n(2\pi)^{n/2} = J_2$ and $\int_{\mathbb{R}^n} w(u-x)(u^i - x^i)(u^j - x^j) du = 0$ when $i \ne j$. Then $\int_{\mathbb{R}^n} \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left(\frac{u-x}{s}\right)\left(\frac{u-x}{s}\right)^T du = J_2 I_n$. Hence

$$V_\rho n(2\pi)^{n/2} L_K \vec{f} = \int_X \left\{ \int_{\mathbb{R}^n} \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left(\frac{u-x}{s}\right)\left(\frac{u-x}{s}\right)^T du \right\} p(x) K_x \vec{f}(x) d\rho_X(x).$$

It follows that

$$\begin{aligned} \|\vec{g} - V_\rho n(2\pi)^{n/2} L_K \vec{f}\|_K &= \left\| \int_X \left\{ \int_{\mathbb{R}^n \setminus X} \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left(\frac{u-x}{s}\right)\left(\frac{u-x}{s}\right)^T du \right\} p(x) K_x \vec{f}(x) d\rho_X(x) \right\|_K \\ &\le \int_X \left\{ \int_{\mathbb{R}^n \setminus X} \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left|\frac{u-x}{s}\right|^2 du \right\} \kappa |\vec{f}(x)| p(x) d\rho_X(x). \end{aligned}$$

Separate the domain X into $X_s := \{x \in X : \inf_{u \in \mathbb{R}^n \setminus X} |u-x| \le \sqrt{s}\}$, consisting of those points whose distance to the boundary is at most $\sqrt{s}$, and its complement $X \setminus X_s$.

If $x \in X \setminus X_s$, any $u \in \mathbb{R}^n \setminus X$ satisfies $|u-x| \ge \sqrt{s}$ and thereby $1 \le s\left|\frac{u-x}{s}\right|^2$. Hence

$$\int_{\mathbb{R}^n \setminus X} \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left|\frac{u-x}{s}\right|^2 du \le s \int_{\mathbb{R}^n \setminus X} \frac{1}{s^n} e^{-\frac{|u-x|^2}{2s^2}} \left|\frac{u-x}{s}\right|^4 du \le s J_4.$$

It follows from (23) that

$$\int_{X\setminus X_s}\left\{\int_{\mathbb{R}^n\setminus X}\frac{1}{s^n}e^{-\frac{|u-x|^2}{2s^2}}\Big|\frac{u-x}{s}\Big|^2du\right\}\kappa|\vec{f}(x)|p(x)d\rho_X(x)\le s\kappa c_\rho M_4\int_{X\setminus X_s}|\vec{f}(x)|d\rho_X(x)$$

which is bounded by $s\kappa c_\rho M_4\|\vec{f}\|_\rho$.

For the subset X_s, we use the Cauchy-Schwartz inequality and (23) to obtain

$$\int_{X_s}\left\{\int_{\mathbb{R}^n\setminus X}\frac{1}{s^n}e^{-\frac{|u-x|^2}{2s^2}}\Big|\frac{u-x}{s}\Big|^2du\right\}\kappa|\vec{f}(x)|p(x)d\rho_X(x)\le\int_{X_s}\kappa c_\rho M_2|\vec{f}(x)|d\rho_X(x).$$

This is bounded by $\kappa c_\rho M_2\sqrt{\rho_X(X_s)}\|\vec{f}\|_\rho$. By (22), $\rho_X(X_s)\le c_\rho^2 s^{2\tau}$. Thus, for $0<s\le 1$,

$$\|\vec{g}-n(2\pi)^{n/2}L_K\vec{f}\|_K\le s^\tau\kappa c_\rho\big(M_4+c_\rho J_2\big)\|\vec{f}\|_\rho.$$

Combine the above two estimates. There holds for any $0<s\le 1$

$$\|L_{K,s}\vec{f}-V_\rho n(2\pi)^{n/2}L_K\vec{f}\|_K\le s^\tau\kappa c_\rho\big(M_{2+\tau}+J_4+c_\rho J_2\big)\|\vec{f}\|_\rho$$

which proves (38) by (34). When $\vec{f}\in\mathcal{H}_K^n$, we have from (34) again that

$$\|L_{K,s}\vec{f}-V_\rho n(2\pi)^{n/2}L_K\vec{f}\|_K\le s^\tau\kappa^2 c_\rho\big(M_{2+\tau}+J_4+c_\rho J_2\big)\|\vec{f}\|_K.$$

This verifies (36) and proves the lemma. ■

The measure $\frac{p(x)}{V_\rho}d\rho_X$ is probability one on X. So we know (see Cucker and Smale (2001)) that the operator L_K can be used to define the reproducing kernel Hilbert space: Let L_K^r be the r-th power of the positive operator L_K on $(L^2_{\rho_X})^n$ having range in $\mathcal{H}_K^n$. Then $\mathcal{H}_K^n$ is the range of $L_K^{1/2}$: $\|\vec{f}\|_\rho=\|L_K^{1/2}\vec{f}\|_K$ for any $\vec{f}\in(L^2_{\rho_X})^n$.

Theorem 17 *Under the assumption (35), we have*

$$\|\vec{f}_\lambda-\nabla f_\rho+\lambda\big(L_{K,s}+\lambda I\big)^{-1}\nabla f_\rho\|_K\le\frac{s}{\lambda}\kappa c'_\rho J_3.$$

Proof By (29), we find that

$$\vec{f}_\lambda-\nabla f_\rho+\lambda\big(L_{K,s}+\lambda I\big)^{-1}\nabla f_\rho=\big(L_{K,s}+\lambda I\big)^{-1}\left\{\vec{f}_{\rho,s}-L_{K,s}\nabla f_\rho\right\}.$$

Then

$$\|\vec{f}_\lambda-\nabla f_\rho+\lambda\big(L_{K,s}+\lambda I\big)^{-1}\nabla f_\rho\|_K\le\|\big(L_{K,s}+\lambda I\big)^{-1}\|_{\mathcal{H}_K^n\to\mathcal{H}_K^n}\|\vec{f}_{\rho,s}-L_{K,s}\nabla f_\rho\|_K$$

which is bounded by $\frac{1}{\lambda}\|\vec{f}_{\rho,s}-L_{K,s}\nabla f_\rho\|_K$. Using (35) on the integral

$$\vec{f}_{\rho,s}-L_{K,s}\nabla f_\rho=\int_X\int_X w(x-u)(u-x)K_x\left\{f_\rho(u)-f_\rho(x)-(u-x)^T\nabla f_\rho(x)\right\}d\rho_X(x)d\rho_X(u),$$

we know that

$$\|\vec{f}_{\rho,s}-L_{K,s}\nabla f_\rho\|_K\le\int_X\int_X w(x-u)|u-x|\|K_x\|_K c'_\rho|u-x|^2\,d\rho_X(x)d\rho_X(u)\le s\kappa c'_\rho J_3.$$

This proves the theorem. ■

Finally, we need to study $\lambda\big(L_{K,s}+\lambda I\big)^{-1}\nabla f_\rho$ in order to estimate the error $\|\vec{f}_\lambda-\nabla f_\rho\|$.

Lemma 18 *Assume (22) and (23). Denote* $c_\rho'' = \left(2\kappa^2 c_\rho\left(M_{2+\tau}+J_4+c_\rho J_2\right)\right)^{1/\tau}$. *Then*

$$\|\left(L_{K,s}+\lambda I\right)^{-1}\vec{f}\| \le 2\|\left(V_\rho n(2\pi)^{n/2}L_K+\lambda I\right)^{-1}\vec{f}\|, \qquad \forall 0<s\le \min\left\{c_\rho''\lambda^{1/\tau},1\right\},$$

where $\vec{f}$ *is either in the space* $\mathcal{H}_K^n$ *or in* $(L^2_{\rho_X})^n$, *and* $\|\cdot\|$ *is the corresponding norm.*

Proof Write $\left(L_{K,s}+\lambda I\right)^{-1}\vec{f} = \left\{\left[V_\rho n(2\pi)^{n/2}L_K+\lambda I\right]-\left[n(2\pi)^{n/2}L_K-L_{K,s}\right]\right\}^{-1}\vec{f}$ as

$$\left\{I-\left[V_\rho n(2\pi)^{n/2}L_K+\lambda I\right]^{-1}\left[V_\rho n(2\pi)^{n/2}L_K-L_{K,s}\right]\right\}^{-1}\left[V_\rho n(2\pi)^{n/2}L_K+\lambda I\right]^{-1}\vec{f}.$$

This in connection with Lemma 16 implies

$$\left\|\left(L_{K,s}+\lambda I\right)^{-1}\vec{f}\right\| \le \left\{1-\frac{1}{\lambda}s^\tau\kappa^2 c_\rho\left(M_{2+\tau}+J_4+c_\rho J_2\right)\right\}^{-1}\left\|\left[V_\rho n(2\pi)^{n/2}L_K+\lambda I\right]^{-1}\vec{f}\right\|.$$

This verifies the lemma. ■

Lemma 18 yields the convergence of $\|\lambda\left(L_{K,s}+\lambda I\right)^{-1}\nabla f_\rho\|$. The convergence rates require some conditions on ∇f_ρ relative to the pair $(L^2_{\rho_X},\mathcal{H}_K)$. The assumption we shall use is $\|L_K^{-r}\nabla f_\rho\|_\rho < \infty$. It means that ∇f_ρ lies in the range of L_K^r. In particular, in the case $r=1/2$, the condition $\|L_K^{-1/2}\nabla f_\rho\|_\rho<\infty$ means $\nabla f_\rho\in\mathcal{H}_K^n$. For more examples about this condition, see Smale and Zhou (2006a).

Theorem 19 *Assume (22), (23), and (35). Let* $0<s\le\min\left\{c_\rho''\lambda^{1/\tau},1\right\}$. *If* $\|L_K^{-r}\nabla f_\rho\|_\rho<\infty$ *for some* $0<r\le 1$, *then*

$$\|\lambda\left(L_{K,s}+\lambda I\right)^{-1}\nabla f_\rho\|_\rho \le 2\lambda^r\left(V_\rho n(2\pi)^{n/2}\right)^{-r}\|L_K^{-r}\nabla f_\rho\|_\rho, \qquad \forall\lambda>0.$$

If moreover $r\ge 1/2$, *then we have for any* $\lambda>0$,

$$\|\lambda\left(L_{K,s}+\lambda I\right)^{-1}\nabla f_\rho\|_K \le 2\lambda^{r-1/2}\left(V_\rho n(2\pi)^{n/2}\right)^{-r}\|L_K^{-r}\nabla f_\rho\|_\rho.$$

In the general situation, we can see that $\|\lambda\left(L_{K,s}+\lambda I\right)^{-1}\nabla f_\rho\|_\rho\to 0$ as $\lambda\to 0$, provided that $\mathcal{H}_K$ is dense in $L^2_{\rho_X}$ (Smale and Zhou, 2003). This can be seen from the following convergence estimate.

Proposition 20 *Assume (22), (23), and (35). Then*

$$\|\lambda\left(L_{K,s}+\lambda I\right)^{-1}\nabla f_\rho\|_\rho \le 2\mathcal{K}\left(\nabla f_\rho,\frac{\sqrt{\lambda}}{V_\rho n(2\pi)^{n/2}}\right), \qquad \forall\, 0<s\le\min\left\{c_\rho''\lambda^{1/\tau},1\right\},$$

where $\mathcal{K}(\vec{f},t)$ *is the K-functional of the pair* $\left((L^2_{\rho_X})^n,\mathcal{H}_K^n\right)$ *defined as*

$$\mathcal{K}(\vec{f},t)=\inf_{\vec{g}\in\mathcal{H}_K^n}\left\{\|\vec{f}-\vec{g}\|_\rho+t\|\vec{g}\|_K\right\}, \qquad t>0. \tag{39}$$

The proof of Proposition 9 shows how our error analysis can be applied.

Proof of Proposition 9. Since the kernel K is C^3 and $\nabla f_\rho \in \mathcal{H}_K^n$, we know from Zhou (2003) that $\frac{\partial f_\rho}{\partial x^i}$ is C^1 for each i. It follows that f_ρ is C^2 and condition (35) is satisfied for some constant $c'_\rho > 0$.

Since $\lambda = \big(1/m\big)^\gamma$ with $\gamma = \frac{\tau}{n+2+3\tau}$ and $s = (\kappa c_\rho)^{2/\tau}\lambda^{1/\tau}$, we see from the fact $J_2 > 1$ that for $m \geq (\kappa c_\rho)^{2(n+2+3\tau)/\tau}$, the restriction $0 < s \leq \min\big\{\big\{2\kappa^2 c_\rho\big(M_{2+\tau} + J_4 + c_\rho J_2\big)\big\}^{1/\tau}\lambda^{1/\tau}, 1\big\}$ in Proposition 15 and Lemma 18 is satisfied. Then by Proposition 15, since $\frac{1}{\tau} - 1 \geq \frac{1}{2}$, we have for some constant $C_\rho > 0$ that

$$\|\vec{f}_\lambda - \nabla f_\rho\|_\rho \leq C_\rho\Big(\frac{s}{\lambda} + \sqrt{\lambda}\Big) \leq C_\rho(1 + (\kappa c_\rho)^{2/\tau})\left(\frac{1}{m}\right)^{\frac{\gamma}{2}}.$$

Applying Lemma 18, we know that

$$\|\lambda\big(L_{K,s} + \lambda I\big)^{-1}\nabla f_\rho\|_K \leq 2\|\lambda\big(V_\rho n(2\pi)^{n/2}L_K + \lambda I\big)^{-1}\nabla f_\rho\|_K \leq 2\|\nabla f_\rho\|_K.$$

This in connection with Theorem 17 implies that

$$\|\vec{f}_\lambda\|_K \leq \|\nabla f_\rho\|_K + 2\|\nabla f_\rho\|_K + \frac{s}{\lambda}\kappa c'_\rho J_3 \leq 3\|\nabla f_\rho\|_K + (\kappa c_\rho)^{2/\tau}\kappa c'_\rho J_3.$$

Finally, we apply (33) of Theorem 14 and know that for a constant $C'_\rho > 0$, with confidence $1-\delta$,

$$\|\vec{f}_{\mathbf{z},\lambda} - \vec{f}_\lambda\|_K \leq C'_\rho\left\{\frac{\log(2/\delta)}{\sqrt{m}\lambda s^{1+n/2}} + \frac{1}{m}\right\} \leq C'_\rho\log(2/\delta)\left\{\left(\frac{1}{m}\right)^{\frac{1}{2}-\gamma-\frac{\gamma}{2}(1+\frac{n}{2})}(\kappa c_\rho)^{-\frac{2}{\tau}(1+\frac{n}{2})} + \frac{1}{m}\right\}.$$

which is bounded by $C''_\rho\log\big(\frac{2}{\delta}\big)\big(\frac{1}{m}\big)^{-\frac{\tau}{2(n+2+3\tau)}}$ with a constant C''_ρ. This is true for $m \geq (\kappa c_\rho)^{2(n+2+3\tau)/\tau}$. Replacing the constant C''_ρ by a new one enables us to bound errors for the finitely many terms with $m < (\kappa c_\rho)^{2(n+2+3\tau)/\tau}$. Thus Proposition 9 is proved. ■

5. Simulated Data and Gene Expression Data

In this section we apply the least-squares gradient algorithm (7) to the variable selection and variable covariance problems. Our idea is to rank the importance of variables according to the norm of their partial derivatives $\|\frac{\partial f_\rho}{\partial x^\ell}\|$, since a small norm implies small changes on the function with respect to this variable. By our error analysis, we expect $\vec{f}_{\mathbf{z},\lambda} \approx \nabla f_\rho$. So we shall use the norms of the components of $\vec{f}_{\mathbf{z},\lambda}$ to rank the variables.

Definition 21 *The relative magnitude of the norm for the variables is defined as*

$$s_\ell^\rho = \frac{\|\big(\vec{f}_{\mathbf{z},\lambda}\big)_\ell\|_K}{\big(\sum_{j=1}^n \|\big(\vec{f}_{\mathbf{z},\lambda}\big)_j\|_K^2\big)^{1/2}}, \qquad \ell = 1,\ldots,n.$$

In the same way, we can study coordinate covariances by the variance of an empirical matrix.

Definition 22 *The* **empirical gradient matrix** *(EGM), $F_{\mathbf{z}}$, is the $n \times m$ matrix whose columns are $\vec{f}_{\mathbf{z},\lambda}(x_j)$ with $j = 1,\ldots,m$. The* **empirical covariance matrix** *(ECM), $\Xi_{\mathbf{z}}$, is the $n \times n$ matrix of inner products of the directional derivative of two coordinates*

$$Cov(\vec{f}_{\mathbf{z},\lambda}) := \Big[\langle\big(\vec{f}_{\mathbf{z},\lambda}\big)_p, \big(\vec{f}_{\mathbf{z},\lambda}\big)_q\rangle_K\Big]_{p,q=1}^n = \sum_{i,j=1}^m c_{i,\mathbf{z}}c_{j,\mathbf{z}}^T K(x_i,x_j).$$

The ECM gives us the covariance between the coordinates while the EGM gives us information as how the variables differ over different sections of the space.

We apply our idea to three data sets. The first data set is an artificial one which we use to illustrate the procedure. The second is a cancer classification problem that has been well studied and serves as further confirmation of the utility of the method. The third data set provides a gold standard as to relevant variables.

5.1 Artificial Data

We construct a function in an $n = 80$ dimensional space which consists of three linear functions over different partitions of the space. We generate 30 samples as follows:

1. For samples $\{x_i\}_{i=1}^{10}$

$$x^j \sim \mathcal{N}(1,\sigma_x), \text{ for } j = 1,\ldots,10; \qquad x^j \sim \mathcal{N}(0,\sigma_x), \text{ for } j = 11,\ldots,80.$$

2. For samples $\{x_i\}_{i=11}^{20}$

$$x^j \sim \mathcal{N}(1,\sigma_x), \text{ for } j = 11,\ldots,20; \qquad x^j \sim \mathcal{N}(0,\sigma_x), \text{ for } j = 1,\ldots,10,21,\ldots,80.$$

3. For samples $\{x_i\}_{i=21}^{30}$

$$x^j \sim \mathcal{N}(1,\sigma_x), \text{ for } j = 41,\ldots,50; \qquad x^j \sim \mathcal{N}(0,\sigma_x), \text{ for } j = 1,\ldots,40,51,\ldots,80.$$

A draw of this x matrix is shown in figure (1a). Three vectors with support over different dimensions were constructed as follows:

$$\begin{aligned} w_1 &= 2 + .5\sin(2\pi i/10) \text{ for } i = 1,...,10 \text{ and } 0 \text{ otherwise,} \\ w_2 &= -2 - .5\sin(2\pi i/10) \text{ for } i = 11,...,20 \text{ and } 0 \text{ otherwise,} \\ w_3 &= -2 - .5\sin(2\pi i/10) \text{ for } i = 41,...,50 \text{ and } 0 \text{ otherwise.} \end{aligned}$$

The values for $\{y_i\}_{i=1}^{30}$ were given by the following linear equations

1. For samples $\{y_i\}_{i=1}^{10}$

$$y_i = x_i \cdot w_1 + \mathcal{N}(0,\sigma_y),$$

2. For samples $\{y_i\}_{i=11}^{20}$

$$y_i = x_i \cdot w_2 + \mathcal{N}(0,\sigma_y),$$

3. For samples $\{y_i\}_{i=21}^{30}$

$$y_i = x_i \cdot w_3 + \mathcal{N}(0,\sigma_y).$$

A draw of the y values is shown in figure (1b).

In figure (1c) we plot the norm of each component of the estimate of the gradient, $\{\|(\vec{f}_{\mathbf{z},\lambda})_\ell\|_K\}_{\ell=1}^{80}$ for $\sigma_x = .05$ and $\sigma_y = .30$. The norm of each component gives an indication of the importance of a variable and variables with small norms can be eliminated. Note that the coordinates with nonzero norm are the ones we expect, $\ell = 1,\ldots,20,41,\ldots,50$.

Perhaps more interesting is that we can evaluate the gradient at each sample $\{x_i\}_{i=1}^{m}$. This leads to an estimate of the covariation of the variables. In figure (1d) we plot the EGM, while the ECM is displayed in figure (1e). The blocking structure of the ECM indicates the coordinates that covary.

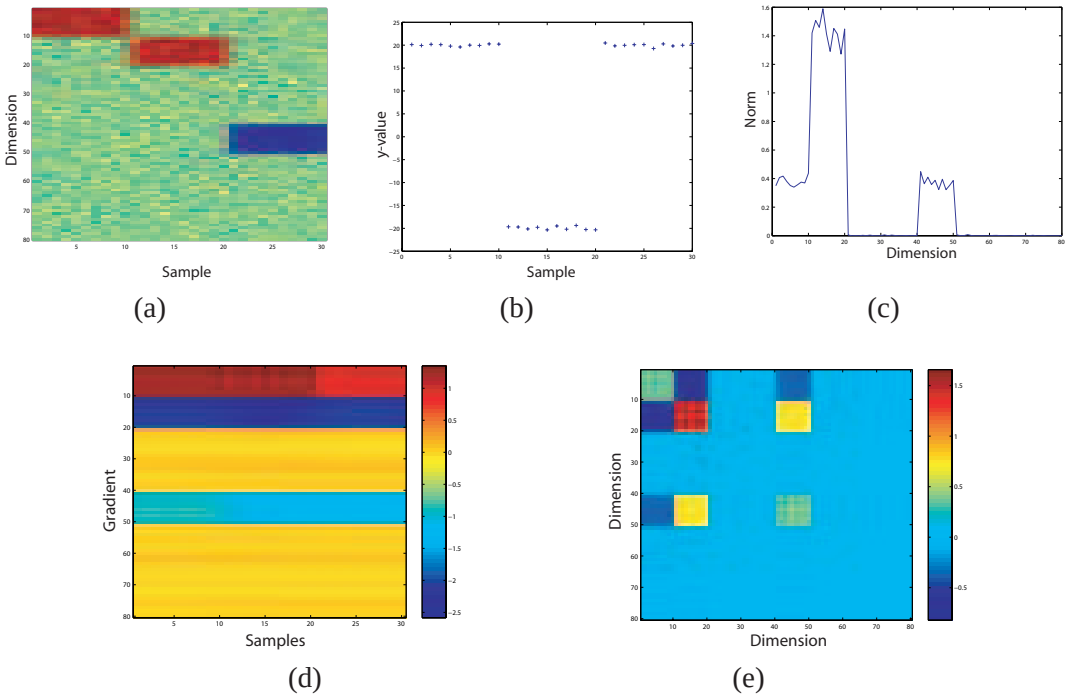


Figure 1: a) The data matrix x where each sample corresponds to a column, b) the vector of y values generated by sampling the function, c) the RKHS norm for each dimension, d) an estimate of the gradient at each sample, the samples correspond to columns, e) the empirical covariance matrix.

5.2 Gene Expression Data

In computational biology, specifically in the subfield of gene expression analysis variable selection and estimation of covariation is of fundamental importance. Microarray technologies enable experimenters to measure the expression level of thousands of genes, the entire genome, at once. The expression level of a gene is proportional to the number of copies of mRNA transcribed by that gene. This readout of gene expression is considered a proxy of the state of the cell. The goals of gene expression analysis include using the expression level of the genes to predict classes, for example tissue morphology or treatment outcome, or real-valued quantities such as toxicity or sensitivity. Fundamental to understanding the biology giving rise to the outcome or toxicity is determining which genes are most relevant for the prediction.

5.2.1 LEUKEMIA CLASSIFICATION

We apply our procedure to a well studied expression data set. The data set is a result of a study using expression data to discriminate acute myeloid leukemia (AML) from acute lymphoblastic leukemia (ALL) (Golub et al., 1999; Slonim et al., 2000) and estimating the genes most relevant to this discrimination. The data set contains 48 samples of AML and 25 samples of ALL. Expression levels of $n = 7,129$ genes and expressed sequence tags (ESTs) were measured via an oligonucleotide microarray for each sample. This data set was split into a training set of 38 samples and a test set of 35 samples.

Various variable selection algorithms have been applied to this data set by using the training set specified in Golub et al. (1999) to select variables and build a classification model and then compute the classification error on the test set. We estimate $\vec{f}_{\mathbf{z},\lambda}$ from the training data and then select the $\mathcal{S}$ variables with the largest s_ℓ^ρ. We then use a linear Support Vector Machine (SVM) to build a classification model and compute the accuracy on the test set. Table 1 reports test errors for various values of $\mathcal{S}$. The classification accuracy is very similar to other feature selection algorithms such as recursive feature elimination (RFE) (Guyon et al., 2002; Lee et al., 2004) and radius-margin bound (RMB) (Chapelle et al., 2002) both of which were developed specifically for SVMs.

genes (S)	5	55	105	155	205	255	305	355	405	455
test errors	1	3	2	1	1	1	1	1	1	1

Table 1: Number of errors in classification for various values of $\mathcal{S}$ using the genes corresponding to dimensions with the largest norms. A linear SVM was used for classification.

In figure (2a-d) we plot the relative magnitude sequence s_ℓ^ρ for the genes. On this data set the decay in the ranked scores $s_{(\ell)}^\rho$ is steeper than that for most statistics that have been previously used on this data. To illustrate this we compared the gradient score to the Fisher score Slonim et al. (2000) for each gene

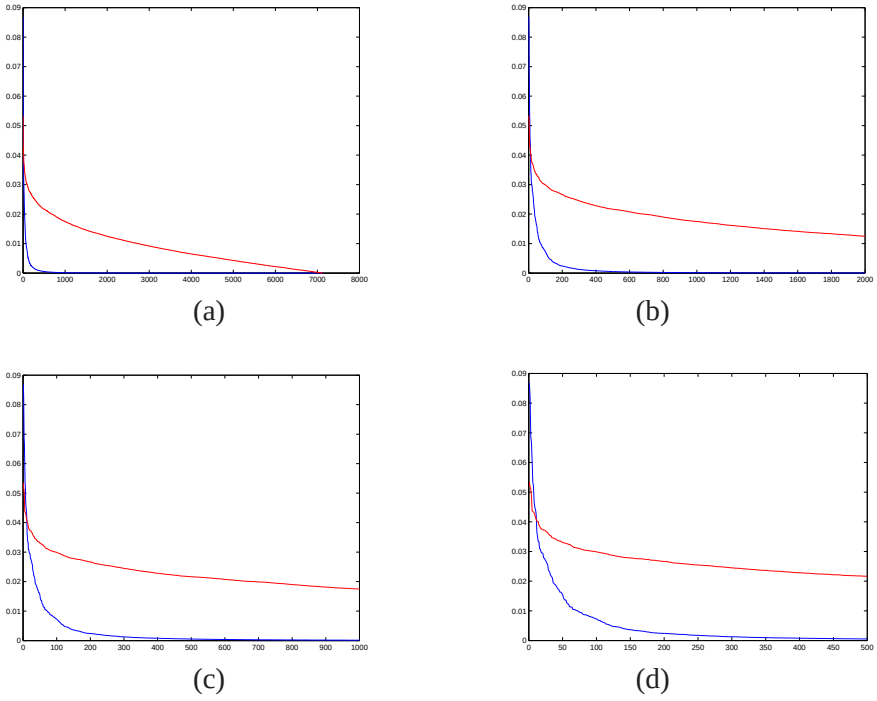
$$t_\ell = \frac{|\hat{\mu}_\ell^{\text{AML}} - \hat{\mu}_\ell^{\text{ALL}}|}{\hat{\sigma}_\ell^{\text{AML}} + \hat{\sigma}_\ell^{\text{ALL}}},$$

where $\hat{\mu}_\ell^{\text{AML}}$ is the mean expression level for the AML samples in the ℓ-th gene, $\hat{\mu}_\ell^{\text{ALL}}$ is the mean expression level for the ALL samples in the ℓ-th gene, $\hat{\sigma}_\ell^{\text{AML}}$ is the standard deviation of the expression level for the AML samples in the ℓ-th gene, and $\hat{\sigma}_\ell^{\text{ALL}}$ is the standard deviation of the expression

level for the ALL samples in the ℓ-th gene. We then normalize these scores

$$s_\ell^F = \frac{t_\ell}{\left(\sum_{p=1}^n t_p^2\right)^{1/2}}.$$

Figure (2a-d) displays the relative decay of $s_{(\ell)}^\rho$ and $s_{(\ell)}^F$ over various numbers of dimensions. In all plots it is apparent that the decay rate of $s_{(\ell)}^\rho$ is much steeper. Plotting the decay of the elements for the normalized hyperplane $\frac{w^0}{\|w^0\|}$ that is the solution of a linear SVM results in a plot much more like that of the Fisher score than the gradient statistic. Whether and how this steepness (sparsity) has an implication on the generalization error is an open question.



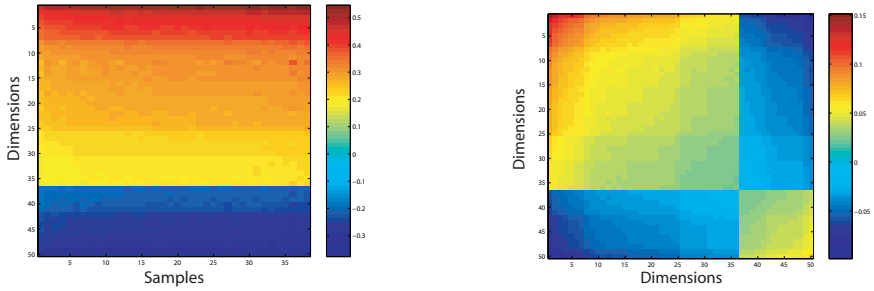
(a) (b) (c) (d)

Figure 2: The decay of $s_{(\ell)}^\rho$ (blue) and $s_{(\ell)}^F$ (red) over: a) all the genes/dimensions, b) the top 2000 genes/dimensions, c) the top 1000 genes/dimensions, d) the top 500 genes/dimensions.

We can also examine the EGM and the ECM. The EGM in this case is a $7,129 \times 38$ matrix and the ECM is $7,129 \times 7,129$ matrix. We plot the EGM in the space of the dimensions corresponding to the top 50 norms ordered by a clustering metric in figure (3a). The covariation in the coordinates is plotted for the top 50 dimensions ordered in the same way as the EGM (see figure (3b)). The blocking structure of the matrix gives us coordinate covariance.

5.2.2 GENDER: "A GOLD STANDARD"

In this section we assess the accuracy of the algorithm with respect to a data set for which a priori biological knowledge gives us a set of important variables. This serves as a gold standard.

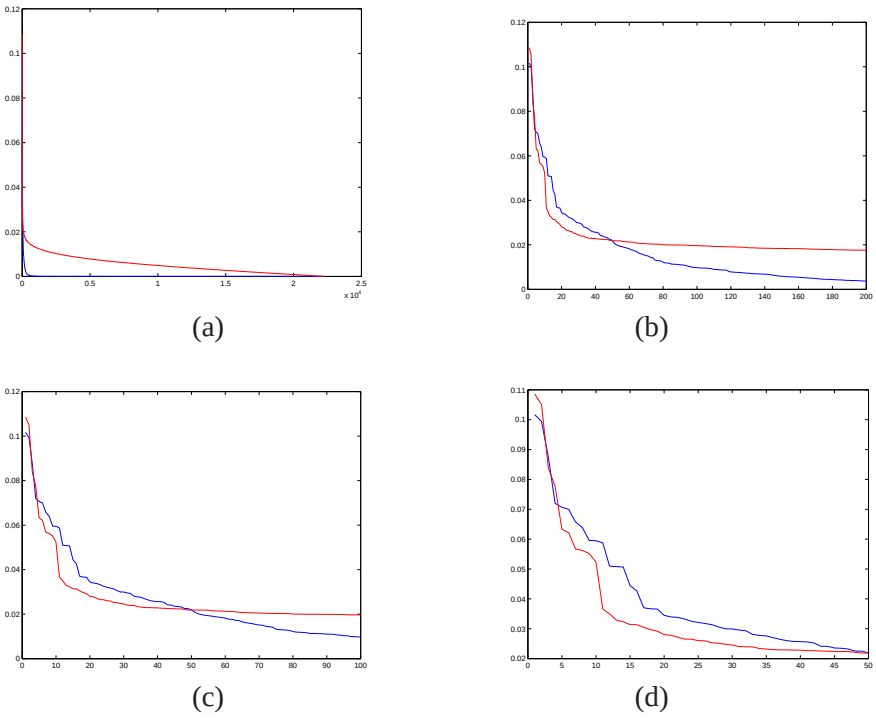


(a) (b)

Figure 3: The a) EGM for the top 50 dimensions ordered by clustering the EGM and b) the ECM for the top 50 dimensions ordered in the same way.

We examine a gene expression data set with 15 male and 17 females samples from lymphblastoid cell lines (unpublished). Expression levels of $n = 22,283$ probes corresponding to genes and expressed sequence tags (ESTs) were measured via an oligonucleotide microarray for each sample.

In figure (4a-d) we plot the relative magnitude sequence s_ℓ^ρ for the genes as compared to those of the relative Fisher score s_ℓ^F and we see again the quicker decay for the gradient norms.



(a) (b)

(c) (d)

Figure 4: The decay of $s_{(\ell)}^\rho$ (blue) and $s_{(\ell)}^F$ (red) over: a) all the genes/dimensions, b) the top 200 genes/dimensions, c) the top 100 genes/dimensions, d) the top 50 genes/dimensions.

From a priori biological knowledge we would predict that the most discriminative genes for gender would be those on the Y chromosome as well as genes on the X chromosome known to

escape X inactivation. The reason that all the genes on the X chromosome would not be expected to be discriminative is due to dosage compensation in expression which takes compensates for the fact that women have two X chromosomes and men have one. The mechanism for this compensation is X inactivation. However, there are genes known to escape X inactivation and these should be differentially expressed. We obtained a list of such genes by combining lists reported in two sources (Carrel et al., 1999; Disteche et al., 2002). There were 35 probes in the X inactivation set and 66 probes corresponding to genes on the Y chromosome.

An important caveat is that while these 101 probes would be expected to be differentially expressed they would not all be expected to rank at the top of a list of genes that are differentially expressed. This is due to the fact that in the cell line or tissue of question there may be other genes that are more strongly differentially expressed due to local conditions. This is why the term gold standard is quoted.

We first used a standard variation filter (Slonim et al., 2000) which reduced the number of probes to about $12,000$. This data set was then standardized (the expression values for each gene was recentered and scaled to be zero mean and standard deviation of one). We then iteratively ran our procedure 20 times, each time removing the bottom 10% of the probes. We found that 16 of the 101 probes appeared in the top ranked 500 probes. Ranking by the Fisher score we found 22 of the top 101 probes in the top ranked 500 probes. Using the logistic loss may result in more like the Fisher score since it is a more appropriate model for classification. Both results are significant with respect to a hypergeometric distribution as the model for the null hypothesis. However, the assumptions of independence in the model which gives rise to the hypergeometric distribution are completely inappropriate in this problem (the probes tend to be strongly correlated). There are statistical tests that account for the correlations but this topic is beyond the scope of this paper (Sweet-Cordero et al., 2005; Subramanian et al., 2005).

6. Discussion

We introduce an algorithm that learns gradients from samples of function values and show its relevance to variable selection. An error analysis is given for the convergence of the estimated gradient to the true gradient. This method also places the problem of variable selection into the powerful framework of Tikhonov regularization. There are many extensions and refinements to this method which we discuss below:

1. Logistic regression model: In Definition 2 we state an algorithm for classification. As many applications of this method are for classification problems it is important to implement a reduced matrix version of this algorithm as was done for regression by Algorithm 1. In addition, an error analysis for the classification setting is also necessary.

2. Fully Bayesian model: The Tikhonov regularization framework coupled with the use of an RKHS allows us to implement a fully Bayesian version of the procedure in the context of Bayesian radial basis (RB) models Liao et al. (2005); Liao (2005). The Bayesian RB framework can be extended to develop a proper probability model for the gradient learning problem. The optimization procedures 1 and 2 would be replaced by Markov Chain Monte-carlo methods and the full posterior rather than the maximum a posteriori estimate would be computed. A very useful result of this is that in addition to the point estimates for the gradient we would also be able to compute confidence intervals.

3. Intrinsic dimension: In Proposition 9 the rate of convergence of the gradient has the form of $O(m^{-1/n})$ which can be extremely slow if n is large. However, in most data sets and when variable selection is meaningful the data are concentrated on a much lower dimensional manifold embedded in the high dimensional space. In this setting an analysis that replaces the ambient dimension n with the intrinsic dimension of the manifold $n_{\mathcal{M}}$ would be of great interest.

4. Semi-supervised setting: Intrinsic properties of the manifold X can be further studied by unlabelled data. This is one of the motivations of semi-supervised learning. In many applications, it is much easier to obtain unlabelled data with a larger sample size $u >> m$. For our purpose, unlabelled data $\mathbf{x} = (x_i)_{i=m+1}^{m+u}$ can be used to reduce the dimension or correlation. Since we learn the gradient by $\vec{f}$, it is natural to use the unlabelled data to control the approximate norm of $\vec{f}$ in some Sobolev spaces and introduce a semi-supervised learning algorithm as

$$\begin{aligned}\vec{f}_{\mathbf{z},\mathbf{x},\lambda,\mu} &= \arg\min_{\vec{f}\in\mathcal{H}_K^n}\left\{\frac{1}{m^2}\sum_{i,j=1}^m w_{i,j}^{(s)}\left(y_i - y_j + \vec{f}(x_i)\cdot(x_j - x_i)\right)^2\right. \\ &\qquad \left. + \frac{\mu}{(m+u)^2}\sum_{i,j=1}^{m+u} W_{i,j}|\vec{f}(x_i) - \vec{f}(x_j)|^2_{\ell^2(\mathbb{R}^n)} + \lambda\|\vec{f}\|_K^2\right\}, \end{aligned} \tag{40}$$

where $\{W_{i,j}\}$ are edge weights in the data adjacency graph, μ is another regularization parameter and often satisfies $\lambda = o(\mu)$.

Acknowledgments

We would like to thank André Elisseeff and Misha Belkin for useful discussions. We would like to thank Aravind Subramanian for help with the X inactivation sets. We would like to acknowledge support for this project from the National Science Foundation and from the University Grants Council of Hong Kong (Project No. CityU 103303) and the Institute for Genome Sciences & Policy at Duke.

Appendix A

Let $\mathcal{S}(X)$ denote the class of all sequences $f = (f_0, f_1, ...)$ of Bochner-integrable random vectors in X with $f_0 \equiv 0$, defined on a probability space. Let $\mathcal{M}(X)$ denote the class of all sequences $f_j \in \mathcal{S}(X)$ that are martingales. The following theorem can be found in (Pinelis, 1994) as Theorem 3.3 with a correction made in Pinelis (1999). Note that Hilbert spaces are $(2,D)$-smooth Banach spaces with $D = 1$.

Theorem 23 *(Pinelis, 1994) Suppose that $f \in \mathcal{M}(X)$, X is a $(2,D)$-smooth separable Banach space and*

$$\left\|\sum_{j=1}^{\infty} E_{j-1}\|f_j - f_{j-1}\|^m\right\|_{\infty} \leq m!\Gamma^{m-2}B^2/(2D^2)$$

for some $\Gamma > 0, B > 0$ and $m = 2,3,...$ Then for all $r \geq 0$,

$$Prob(\sup_j \|f_j\| \geq r) \leq 2\exp\left(-\frac{r^2}{\Gamma r + B^2 + B\sqrt{B^2 + 2\Gamma r}}\right).$$

Appendix B

We give the proof for Theorem 7.
Proof We divide our approximation in three steps.

Step 1. Approximate $c_{\mathbf{z}}$ by $\widetilde{c}_{\mathbf{z}}$ which is defined by

$$\widetilde{c}_{\mathbf{z}} = \left(m^2\lambda I_{mn} + \mathrm{diag}\{B_1,\cdots,B_m\}\left[K(x_i,x_j)I_n\right]_{i,j=1}^m\right)^{-1}(\widetilde{\mathcal{Y}}_1,\ldots,\widetilde{\mathcal{Y}}_m)^T,$$

where $\widetilde{\mathcal{Y}}_i = \sum_{j=1}^m w_{i,j}(y_j - y_i)\sum_{\ell=1}^{\mathcal{S}}\sigma_\ell\left(U_\ell^j - U_\ell^i\right)V_\ell$. For each i,

$$\left|\widetilde{\mathcal{Y}}_i - Y_i\right|_{\ell^2(\mathbb{R}^n)} \quad \le \quad \sum_{j=1}^m w_{i,j}2M\sigma_{\mathcal{S}+1}\left(\sum_{\ell=\mathcal{S}+1}^d\left(U_\ell^j - U_\ell^i\right)^2\right)^{1/2}.$$

Since the matrix U is orthogonal, we know that $\sum_{\ell=1}^m\left(U_\ell^j\right)^2 = 1$ for each j and $\sum_{j=1}^m\left(U_\ell^j\right)^2 = 1$ for each ℓ. By the Schartz inequality, $\left|\widetilde{\mathcal{Y}}_i - Y_i\right|^2_{\ell^2(\mathbb{R}^n)}$ is bounded by

$$(4M\sigma_{\mathcal{S}+1})^2\left\{\sum_{j=1}^m\left(w_{i,j}\right)^2\cdot\sum_{j=1}^m\sum_{\ell=\mathcal{S}+1}^d\left(U_\ell^j\right)^2 + \left(\sum_{j=1}^m w_{i,j}\right)^2\sum_{\ell=\mathcal{S}+1}^d\left(U_\ell^i\right)^2\right\}.$$

It follows that

$$\left|c_{\mathbf{z}} - \widetilde{c}_{\mathbf{z}}\right|_{\ell^2(\mathbb{R}^{mn})} \le \frac{1}{m^2\lambda}\left\{\sum_{i=1}^m\left|\widetilde{\mathcal{Y}}_i - Y_i\right|^2_{\ell^2(\mathbb{R}^n)}\right\}^{1/2} \le \frac{4M\sigma_{\mathcal{S}+1}\sqrt{d-\mathcal{S}}}{m^2\lambda}\sqrt{\Delta_m}.$$

Step 2. Approximate $\widetilde{c}_{\mathbf{z}}$ by $\widetilde{b}_{\mathbf{z}}$ which is defined by

$$\widetilde{b}_{\mathbf{z}} = \left(m^2\lambda I_{mn} + \mathrm{diag}\{\widetilde{B}_1,\cdots,\widetilde{B}_m\}\left[K(x_i,x_j)I_n\right]_{i,j=1}^m\right)^{-1}(\widetilde{\mathcal{Y}}_1,\ldots,\widetilde{\mathcal{Y}}_m)^T,$$

where

$$\widetilde{B}_i = \sum_{j=1}^m w_{i,j}\sum_{\ell=1}^{\mathcal{S}}\sum_{p=1}^{\mathcal{S}}\sigma_\ell\sigma_p\left(U_\ell^j - U_\ell^i\right)\left(U_p^j - U_p^i\right)V_\ell V_p^T.$$

For $b\in\mathbb{R}^n$, the vector $\left(B_i - \widetilde{B}_i\right)b$ equals

$$\left\{\sum_{\ell=\mathcal{S}+1}^d\sum_{p=1}^{\mathcal{S}} + \sum_{\ell=1}^d\sum_{p=\mathcal{S}+1}^d\right\}\sigma_\ell\sigma_p\sum_{j=1}^m w_{i,j}\left(U_\ell^j - U_\ell^i\right)\left(U_p^j - U_p^i\right)(V_p^T b)V_\ell.$$

By the Schwartz inequality, the $\ell^2(\mathbb{R}^n)$ norm of the first term above is bounded by

$$\begin{aligned}&\sum_{j=1}^m w_{i,j}\sigma_{\mathcal{S}+1}\left\{\sum_{\ell=\mathcal{S}+1}^d\left(U_\ell^j - U_\ell^i\right)^2\left(\sum_{p=1}^{\mathcal{S}}\sigma_p\left(U_p^j - U_p^i\right)(V_p^T b)\right)^2\right\}^{1/2}\\ \le\ &\sum_{j=1}^m w_{i,j}\sigma_{\mathcal{S}+1}\left\{\sum_{\ell=\mathcal{S}+1}^d\left(U_\ell^j - U_\ell^i\right)^2\right\}^{1/2}|b|_{\ell^2(\mathbb{R}^n)}\left\{\sum_{p=1}^{\mathcal{S}}\sigma_p^2\left(U_p^j - U_p^i\right)^2\right\}^{1/2}.\end{aligned}$$

This is at most $2\sigma_{S+1}\sigma_1|b|_{\ell^2(\mathbb{R}^n)}\sum_{j=1}^m w_{i,j}$. The $\ell^2(\mathbb{R}^n)$ norm of the second term in the expression of $\left(B_i-\widetilde{B}_i\right)b$ can be bounded in the same way and we thus have

$$\left\|\left(B_i-\widetilde{B}_i\right)b\right\|_{\ell^2(\mathbb{R}^n)}\leq 4\sigma_{S+1}\sigma_1|b|_{\ell^2(\mathbb{R}^n)}\sum_{j=1}^m w_{i,j}.$$

Then we have the following estimate for the operator norm of the difference of the diagonal operators

$$\left\|\mathrm{diag}\{B_1,\cdots,B_m\}-\mathrm{diag}\{\widetilde{B}_1,\cdots,\widetilde{B}_m\}\right\|\leq 4\sigma_{S+1}\sigma_1\max_{1\leq i\leq m}\sum_{j=1}^m w_{i,j}.$$

It follows that

$$\begin{aligned}&\left\|\mathrm{diag}\{B_1,\cdots,B_m\}\left[K(x_i,x_j)I_n\right]_{i,j=1}^m-\mathrm{diag}\{\widetilde{B}_1,\cdots,\widetilde{B}_m\}\left[K(x_i,x_j)I_n\right]_{i,j=1}^m\right\|\\&\leq 4\kappa^2 m\sigma_{S+1}\sigma_1\max_{1\leq i\leq m}\sum_{j=1}^m w_{i,j}.\end{aligned}$$

Notice that for two invertible operators L_1,L_2 on a Hilbert space, there holds

$$L_1^{-1}-L_2^{-1}=L_1^{-1}(L_2-L_1)L_2^{-1}.$$

Hence

$$\|L_1^{-1}-L_2^{-1}\|\leq\|L_1^{-1}\|\,\|L_2-L_1\|\,\|L_2^{-1}\|.$$

Applying this to our setting, we have

$$\left|\widetilde{b}_{\mathbf{z}}-\widetilde{c}_{\mathbf{z}}\right|_{\ell^2(\mathbb{R}^{mn})}\leq\frac{4\kappa^2 m\sigma_{S+1}\sigma_1}{(m^2\lambda)^2}\left\{\max_{1\leq i\leq m}\sum_j w_{i,j}\right\}\left\|(\widetilde{\mathcal{Y}}_1,\ldots,\widetilde{\mathcal{Y}}_m)^T\right\|_{\ell^2(\mathbb{R}^{mn})}.$$

For each i, we have

$$\left|\widetilde{\mathcal{Y}}_i\right|_{\ell^2(\mathbb{R}^n)}\leq 2M\sum_{j=1}^m w_{i,j}\left\{\left(\sum_{\ell=1}^S\sigma_\ell^2\left(U_\ell^j\right)^2\right)^{1/2}+\left(\sum_{\ell=1}^S\sigma_\ell^2\left(U_\ell^i\right)^2\right)^{1/2}\right\}.$$

It follows that

$$\left|\widetilde{b}_{\mathbf{z}}-\widetilde{c}_{\mathbf{z}}\right|_{\ell^2(\mathbb{R}^{mn})}\leq\frac{8M\kappa^2 m\sigma_{S+1}\sigma_1^2\sqrt{S}}{(m^2\lambda)^2}\Delta_m.$$

Step 3. Find the coefficients $\widetilde{b}_{\mathbf{z}}$. The linear system it satisfies is

$$m^2\lambda\widetilde{b}_{i,\mathbf{z}}+\sum_{q=1}^m\sum_{j=1}^m w_{i,j}\sum_{\ell=1}^S\sum_{p=1}^S\sigma_\ell\sigma_p\left(U_\ell^j-U_\ell^i\right)\left(U_p^j-U_p^i\right)V_\ell V_p^T K(x_i,x_q)\widetilde{b}_{q,\mathbf{z}}=\widetilde{\mathcal{Y}}_i,$$

where $i=1,\ldots,m$. Since $\widetilde{\mathcal{Y}}_i$ lies in $\mathrm{span}\{V_\ell\}_{\ell=1}^S$, we know that each $\widetilde{b}_{i,\mathbf{z}}$ also lies in this subspace of $\mathbb{R}^n$. That is, there is a vector $b_{i,\mathbf{z}}^*\in\mathbb{R}^S$ such that

$$\widetilde{b}_{i,\mathbf{z}}=\sum_{\ell=1}^S b_{i,\mathbf{z}}^{*\ell}V_\ell,\qquad i=1,\ldots,m.$$

Substituting this expression into the linear system for $\widetilde{b}_{\mathbf{z}}$, we know that $b^*_{\mathbf{z}}$ can be solved by the linear system

$$m^2\lambda b^{*\ell}_{i,\mathbf{z}} + \sum_{q=1}^{m}\sum_{j=1}^{m} w_{i,j}\sum_{p=1}^{s}\sigma_\ell\sigma_p\left(U^j_\ell - U^i_\ell\right)\left(U^j_p - U^i_p\right)K(x_i,x_q)b^{*p}_{q,\mathbf{z}}$$
$$= \sum_{j=1}^{m} w_{i,j}(y_j - y_i)\sigma_\ell\left(U^j_\ell - U^i_\ell\right), \qquad 1 \le \ell \le s, 1 \le i \le m.$$

This is exactly the linear system (19). Therefore, $\hat{b}_{i,\mathbf{z}} = b^*_{i,\mathbf{z}}$ for each i and $\widetilde{b}_{\mathbf{z}} = b_{\mathbf{z}}$ is the desired coefficients for the function $\vec{f}_{\mathbf{z},\lambda,s}$. ■

Appendix C

The following is Matlab® code that implements algorithm (1), the approximation algorithm with reduced matrix size. The code could be made more efficient by exploiting the vector nature of Matlab. However, we include the version with loops for transparency.

```
% a matrix x that is dim by m where m is the number of samples
% a vector y that is m by 1
% eps is a constraint on the ratio of the top s eigenvalues to the sum over
%         all eigenvalues
% lambda is the regularization constant
% sigma is the variance of the weight matrix computed automatically from the
%         data
% F is the gradient evaluated at each sample again a dim by m matrix
% nrm is the RKHS norm for each dimension


function [F,nrm,sigma] =
solveder(x,y,lambda,eps)

[dim,m] = size(x);

% this subroutine computes distances between all pairs and sets sigma to the
%         median
a = zeros(m,m);
 for i=1:m
   for j=1:m
       a(i,j) = norm(x(:,i)-x(:,j));
   end
 end
 sigma = median(median(a));

% this subroutine computes the weight matrix
a = zeros(m,m);
```

```
 for i=1:m
   for j=1:m
       a(i,j) = (1/(sigma*sqrt(2*pi)))*exp(-norm(x(:,i)-x(:,j))^2/(2*sigma^2));
   end
 end

% the kernel matrix is computed will add nonlinear version
K = zeros(m,m); K = transpose(x)*x;

% constructs the matrix of differences between all points
M = zeros(dim,m); for i=1:m
     M(:,i) = x(:,i)-x(:,m);
end


 % computes the eigenvalues and eigenvectors of M^t M
 % and keeps s eigenvectors as specified by eps
 d = eig(K);
 W = transpose(M)*M;
 [V,d] = eig(W);
 d = diag(d);
 vals = cumsum(d);
 inds = find(vals/vals(m) < eps);
 s = m-max(inds);

 % since matlab indexes eigenvalues from smallest to largest we reverse
 U = zeros(m,m);
 dp = zeros(m,1);
 for i=1:m
    U(:,m-i) = V(:,i);
    dp(i) = d(m-i);
 end

 % projects of the paired differences into the subspace of the s eigenfunctions
t = zeros(s,m); for i=1:m
     t(:,i) = sqrt(dp(1:s)).*transpose(U(i,1:s));
 end

 Ktilde = zeros(m*s,m*s);
 ytilde = zeros(m*s,1);

 % computes the Ktilde matrix and the vector script Y
 for i=1:m
    Bmat = zeros(s,s);
    yv = zeros(s,1);
```

```
    for j=1:m
        Bmat = Bmat+a(i,j)* (t(:,j)-t(:,i))*(transpose(t(:,j)-t(:,i)));
        yv = yv + a(i,j)*(y(j)-y(i))*(t(:,j)-t(:,i));
    end

    ytilde((i-1)*s+1:i*s,1) = yv;

    for j=1:m
        Ktilde((i-1)*s+1:i*s,(j-1)*s+1:j*s) = K(i,j)*Bmat;
    end
end


% solves the linear system for coefficients c
I = eye(m*s);
c = (m^2*lambda*I+Ktilde)\ytilde;

% uwraps the coefficients into a vector for each sample
Cmat = zeros(dim,m);
for i = 1:m
   vec=zeros(dim,1);
    for j=1:s
        vec = vec+(c((i-1)*s+j,1)/sqrt(dp(j,1)))*M*U(:,j);
    end
    Cmat(:,i) = vec;
end

% computes the gradient for each sample
F = zeros(dim,m);
F = Cmat*K;

%computes the norm for each dimension
nrm = zeros(dim,1);
for i=1:dim
  nrm(i) = Cmat(i,:)*K*transpose(Cmat(i,:));
end
```

References

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 686:337–404, 1950.

M. Belkin and P. Niyogi. Semi-Supervised Learning on Riemannian Manifolds. *Machine Learning*, 56(1-3):209–239, 2004.

I. Carrel, A. Cottle, K. Coglin, and H. Willard. A first-generation X-incativation profile of the human X chromosome. *Proc. Natl. Acad. Sci. USA*, 96:14440–14444, 1999.

O. Chapelle, V. N. Vapnik, O. Bousquet, and S. Mukherjee. Choosing Multiple Parameters for Support Vector Machines. *Machine Learning*, 46(1-3):131–159, 2002.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

C. Cortes and V. N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39: 1–49, 2001.

C. Disteche, G. Flippova, and K. Tsuchiya. Escape from X inactivation. *Cytogenet. Genome Res.*, 99:35–43, 2002.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization Networks and Support Vector Machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: theory and applications to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Society*, 99:67–81, 2004.

M. Liao. *Bayesian estimation of gene expression index and Bayesian kernel models*. PhD thesis, Duke University, Durham, NC, 2005.

M. Liao, F. Liang, S. Mukherjee, and M. West. Bayesian kernel regression and radial basis function models. Preprint, 2005.

C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural Computation*, 17: 177–204, 2005.

I. Pinelis. Optimum bounds for the distributions of martingales in Banach spaces. *Ann. Probab.*, 22:1679–1706, 1994.

I. Pinelis. Correction: "Optimum bounds for the distributions of martingales in Banach spaces". *Ann. Probab.*, 27:2119, 1999.

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

B. Schoelkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

D. K. Slonim, P. Tamayo, J. P. Mesirov, T. R. Golub, and E. S. Lander. Class prediction and discovery using gene expression data. In *Proc. of the 4th Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 263–272, 2000.

S. Smale and D. X. Zhou. Learning theory estimates via integral operators and their approximations. *Constr. Approx.*, 24, 2006a.

S. Smale and D. X. Zhou. Shannon sampling II. Connections to learning theory. *Appl. Comput. Harmonic Anal.*, 19:285–302, 2006b.

S. Smale and D. X. Zhou. Shannon sampling and function reconstruction from point values. *Bull. Amer. Math. Soc.*, 41:279–305, 2004.

S. Smale and D. X. Zhou. Estimating the approximation error in learning theory. *Anal. Appl.*, 1: 17–41, 2003.

A. Subramanian, P. Tamayo, VK. Mootha, S. Mukherjee, BL. Ebert, MA. Gillette, A. Paulovich, SL. Pomeroy, TR. Golub, ES. Lander, and JP. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A*, 2005.

A. Sweet-Cordero, S. Mukherjee, A. Subramanian, H. You, J. J. Roix, C. Ladd-Acosta, J. P. Mesirov, T. R. Golub, and T. Jacks. An oncogenic KRAS2 expression signature identified by cross-species gene-expression analysis. *Nature Genetics*, 37:48–55, 2005.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J Royal Stat Soc B*, 58(1):267–288, 1996.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

E. De Vito, A. Caponnetto, and L. Rosasco. Model selection for regularized least-squares algorithm in learning. *Foundat. Comput. Math.*, 5:59–85, 2005.

G. Wahba and J. Wendelberger. Some new mathematical methods for variational objective analysis using splines and cross-validation. *Monthly Weather Rev.*, 108:1122–1145, 1980.

M. West. Bayesian factor regression models in the "large p, small n" paradigm. In J. M. Bernardo et al., editor, *Bayesian Statistics 7*, pages 723–732. Oxford, 2003.

Q. Wu and D. X. Zhou. Support vector machine classifiers: linear programming versus quadratic programming. *Neural Computation*, 17:1160–1187, 2005.

T. Zhang. Leave-one-out bounds for kernel methods. *Neural Computation*, 15(6):1397–1437, 2003.

D. X. Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE Trans. Inform. Theory*, 49:1743–1752, 2003.

Journal of Machine Learning Research 7 (2006) 551–585 Submitted 5/05; Published 3/06

Online Passive-Aggressive Algorithms

Koby Crammer[*] CRAMMER@CIS.UPENN.EDU
Ofer Dekel OFERD@CS.HUJI.AC.IL
Joseph Keshet JKESHET@CS.HUJI.AC.IL
Shai Shalev-Shwartz SHAIS@CS.HUJI.AC.IL
Yoram Singer[†] SINGER@CS.HUJI.AC.IL
School of Computer Science and Engineering
The Hebrew University
Jerusalem, 91904, Israel

Editor: Manfred K. Warmuth

Abstract

We present a family of margin based online learning algorithms for various prediction tasks. In particular we derive and analyze algorithms for binary and multiclass categorization, regression, uniclass prediction and sequence prediction. The update steps of our different algorithms are all based on analytical solutions to simple constrained optimization problems. This unified view allows us to prove worst-case loss bounds for the different algorithms and for the various decision problems based on a single lemma. Our bounds on the cumulative loss of the algorithms are relative to the smallest loss that can be attained by any fixed hypothesis, and as such are applicable to both realizable and unrealizable settings. We demonstrate some of the merits of the proposed algorithms in a series of experiments with synthetic and real data sets.

1. Introduction

In this paper we describe and analyze several online learning tasks through the same algorithmic prism. We first introduce a simple online algorithm which we call Passive-Aggressive (PA) for online binary classification (see also (Herbster, 2001)). We then propose two alternative modifications to the PA algorithm which improve the algorithm's ability to cope with noise. We provide a unified analysis for the three variants. Building on this unified view, we show how to generalize the binary setting to various learning tasks, ranging from regression to sequence prediction.

The setting we focus on is that of online learning. In the online setting, a learning algorithm observes instances in a sequential manner. After each observation, the algorithm predicts an outcome. This outcome can be as simple as a yes/no $(+/-)$ decision, as in the case of binary classification problems, and as complex as a string over a large alphabet. Once the algorithm has made a prediction, it receives feedback indicating the correct outcome. Then, the online algorithm may modify its prediction mechanism, presumably improving the chances of making an accurate prediction on subsequent rounds. Online algorithms are typically simple to implement and their analysis often provides tight bounds on their performance (see for instance Kivinen and Warmuth (1997)).

∗. Current affiliation: Department of Computer and Information Science, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104, USA.
†. Current affiliation: Google, Moutain View, CA 94043, USA.

Our learning algorithms use hypotheses from the set of linear predictors. While this class may seem restrictive, the pioneering work of Vapnik (1998) and colleagues demonstrates that by using Mercer kernels one can employ highly non-linear predictors and still entertain all the formal properties and simplicity of linear predictors. For concreteness, our presentation and analysis are confined to the linear case which is often referred to as the primal version (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002). As in other constructions of linear kernel machines, our paradigm also builds on the notion of margin.

Binary classification is the first setting we discuss in the paper. In this setting each instance is represented by a vector and the prediction mechanism is based on a hyperplane which divides the instance space into two half-spaces. The margin of an example is proportional to the distance between the instance and the hyperplane. The PA algorithm utilizes the margin to modify the current classifier. The update of the classifier is performed by solving a constrained optimization problem: we would like the new classifier to remain as close as possible to the current one while achieving at least a unit margin on the most recent example. Forcing a unit margin might turn out to be too aggressive in the presence of noise. Therefore, we also describe two versions of our algorithm which cast a tradeoff between the desired margin and the proximity to the current classifier.

The above formalism is motivated by the work of Warmuth and colleagues for deriving online algorithms (see for instance (Kivinen and Warmuth, 1997) and the references therein). Furthermore, an analogous optimization problem arises in support vector machines (SVM) for classification (Vapnik, 1998). Indeed, the core of our construction can be viewed as finding a support vector machine based on a single example while replacing the norm constraint of SVM with a proximity constraint to the current classifier. The benefit of this approach is two fold. First, we get a closed form solution for the next classifier. Second, we are able to provide a unified analysis of the cumulative loss for various online algorithms used to solve different decision problems. Specifically, we derive and analyze versions for regression problems, uniclass prediction, multiclass problems, and sequence prediction tasks.

Our analysis is in the realm of relative loss bounds. In this framework, the cumulative loss suffered by an online algorithm is compared to the loss suffered by a fixed hypothesis that may be chosen in hindsight. Our proof techniques are surprisingly simple and the proofs are fairly short and easy to follow. We build on numerous previous results and views. The mere idea of deriving an update as a result of a constrained optimization problem compromising of two opposing terms, has been largely advocated by Littlestone, Warmuth, Kivinen and colleagues (Littlestone, 1989; Kivinen and Warmuth, 1997). Online margin-based prediction algorithms are also quite prevalent. The roots of many of the papers date back to the Perceptron algorithm (Agmon, 1954; Rosenblatt, 1958; Novikoff, 1962). More modern examples include the ROMMA algorithm of Li and Long (2002), Gentile's ALMA algorithm (Gentile, 2001), the MIRA algorithm (Crammer and Singer, 2003b), and the NORMA algorithm (Kivinen et al., 2002). The MIRA algorithm is closely related to the work presented in this paper, and specifically, the MIRA algorithm for binary classification is identical to our basic PA algorithm. However, MIRA was designed for *separable* binary and multiclass problems whereas our algorithms also apply to nonseparable problems. Furthermore, the loss bounds derived in Crammer and Singer (2003b) are inferior and less general than the bounds derived in this paper. The NORMA algorithm also shares a similar view of classification problems. Rather than projecting the current hypothesis onto the set of constraints induced by the most recent example, NORMA's update rule is based on a stochastic gradient approach (Kivinen et al., 2002). Of all the work on online learning algorithms, the work by Herbster (2001) is probably the closest to the work

presented here. Herbster describes and analyzes a projection algorithm that, like MIRA, is essentially the same as the basic PA algorithm for the separable case. We surpass MIRA and Herbster's algorithm by providing bounds for both the separable and the nonseparable settings using a unified analysis. As mentioned above we also extend the algorithmic framework and the analysis to more complex decision problems.

The paper is organized as follows. In Sec. 2 we formally introduce the binary classification problem and in the next section we derive three variants of an online learning algorithm for this setting. The three variants of our algorithm are then analyzed in Sec. 4. We next show how to modify these algorithms to solve regression problems (Sec. 5) and uniclass prediction problems (Sec. 6). We then shift gears to discuss and analyze more complex decision problems. Specifically, in Sec. 7 we describe a generalization of the algorithms to multiclass problems and further extend the algorithms to cope with sequence prediction problems (Sec. 9). We describe experimental results with binary and multiclass problems in Sec. 10 and conclude with a discussion of future directions in Sec. 11.

2. Problem Setting

As mentioned above, the paper describes and analyzes several online learning tasks through the same algorithmic prism. We begin with binary classification which serves as the main building block for the remainder of the paper. Online binary classification takes place in a sequence of rounds. On each round the algorithm observes an instance and predicts its label to be either $+1$ or -1. After the prediction is made, the true label is revealed and the algorithm suffers an *instantaneous loss* which reflects the degree to which its prediction was wrong. At the end of each round, the algorithm uses the newly obtained instance-label pair to improve its prediction rule for the rounds to come.

We denote the instance presented to the algorithm on round t by $\mathbf{x}_t$, and for concreteness we assume that it is a vector in $\mathbb{R}^n$. We assume that $\mathbf{x}_t$ is associated with a unique label $y_t \in \{+1,-1\}$. We refer to each instance-label pair $(\mathbf{x}_t, y_t)$ as an *example*. The algorithms discussed in this paper make predictions using a classification function which they maintain in their internal memory and update from round to round. We restrict our discussion to classification functions based on a vector of weights $\mathbf{w} \in \mathbb{R}^n$, which take the form $\mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$. The magnitude $|\mathbf{w} \cdot \mathbf{x}|$ is interpreted as the degree of confidence in this prediction. The task of the algorithm is therefore to incrementally learn the weight vector $\mathbf{w}$. We denote by $\mathbf{w}_t$ the weight vector used by the algorithm on round t, and refer to the term $y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ as the (signed) *margin* attained on round t. Whenever the margin is a positive number then $\mathrm{sign}(\mathbf{w}_t \cdot \mathbf{x}_t) = y_t$ and the algorithm has made a correct prediction. However, we are not satisfied by a positive margin value and would additionally like the algorithm to predict with high confidence. Therefore, the algorithm's goal is to achieve a margin of at least 1 as often as possible. On rounds where the algorithm attains a margin less than 1 it suffers an instantaneous loss. This loss is defined by the following *hinge-loss* function,

$$\ell\big(\mathbf{w};(\mathbf{x},y)\big) \;=\; \begin{cases} 0 & y(\mathbf{w}\cdot\mathbf{x}) \geq 1 \\ 1 - y(\mathbf{w}\cdot\mathbf{x}) & \text{otherwise} \end{cases} \quad . \tag{1}$$

Whenever the margin exceeds 1, the loss equals zero. Otherwise, it equals the difference between the margin value and 1. We note in passing that the choice of 1 as the margin threshold below which a loss is suffered is rather arbitrary. In Sec. 5 we generalize the hinge-loss function in the context of regression problems, by letting the threshold be a user-defined parameter. We abbreviate the loss

suffered on round t by ℓ_t, that is, $\ell_t = \ell\big(\mathbf{w}_t;(\mathbf{x}_t,y_t)\big)$. The algorithms presented in this paper will be shown to attain a small *cumulative squared loss* over a given sequence of examples. In other words, we will prove different bounds on $\sum_{t=1}^{T}\ell_t^2$, where T is the length of the sequence. Notice that whenever a prediction mistake is made then $\ell_t^2 \geq 1$ and therefore a bound on the cumulative squared loss also bounds the number of prediction mistakes made over the sequence of examples.

3. Binary Classification Algorithms

In the previous section we described a general setting for binary classification. To obtain a concrete algorithm we must determine how to initialize the weight vector $\mathbf{w}_1$ and we must define the update rule used to modify the weight vector at the end of each round. In this section we present three variants of an online learning algorithm for binary classification. The pseudo-code for the three variants is given in Fig. 1. The vector $\mathbf{w}_1$ is initialized to $(0,\ldots,0)$ for all three variants, however each variant employs a different update rule. We focus first on the simplest of the three, which on round t sets the new weight vector $\mathbf{w}_{t+1}$ to be the solution to the following constrained optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) = 0. \tag{2}$$

Geometrically, $\mathbf{w}_{t+1}$ is set to be the projection of $\mathbf{w}_t$ onto the half-space of vectors which attain a hinge-loss of zero on the current example. The resulting algorithm is *passive* whenever the hinge-loss is zero, that is, $\mathbf{w}_{t+1} = \mathbf{w}_t$ whenever $\ell_t = 0$. In contrast, on those rounds where the loss is positive, the algorithm *aggressively* forces $\mathbf{w}_{t+1}$ to satisfy the constraint $\ell(\mathbf{w}_{t+1};(\mathbf{x}_t,y_t)) = 0$ regardless of the step-size required. We therefore name the algorithm *Passive-Aggressive* or *PA* for short.

The motivation for this update stems from the work of Helmbold et al. (Helmbold et al., 1999) who formalized the trade-off between the amount of progress made on each round and the amount of information retained from previous rounds. On one hand, our update requires $\mathbf{w}_{t+1}$ to correctly classify the current example with a sufficiently high margin and thus progress is made. On the other hand, $\mathbf{w}_{t+1}$ must stay as close as possible to $\mathbf{w}_t$, thus retaining the information learned on previous rounds.

The solution to the optimization problem in Eq. (2) has a simple closed form solution,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t \;\text{ where }\; \tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}. \tag{3}$$

We now show how this update is derived using standard tools from convex analysis (see for instance (Boyd and Vandenberghe, 2004)). If $\ell_t = 0$ then $\mathbf{w}_t$ itself satisfies the constraint in Eq. (2) and is clearly the optimal solution. We therefore concentrate on the case where $\ell_t > 0$. First, we define the Lagrangian of the optimization problem in Eq. (2) to be,

$$\mathcal{L}(\mathbf{w},\tau) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \;+\; \tau\big(1-y_t(\mathbf{w}\cdot\mathbf{x}_t)\big), \tag{4}$$

where $\tau \geq 0$ is a Lagrange multiplier. The optimization problem in Eq. (2) has a convex objective function and a single feasible affine constraint. These are sufficient conditions for Slater's condition to hold therefore finding the problem's optimum is equivalent to satisfying the Karush-Khun-Tucker

INPUT: aggressiveness parameter $C > 0$
INITIALIZE: $\mathbf{w}_1 = (0, \ldots, 0)$
For $t = 1, 2, \ldots$

- receive instance: $\mathbf{x}_t \in \mathbb{R}^n$
- predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
- receive correct label: $y_t \in \{-1, +1\}$
- suffer loss: $\ell_t = \max\{0\ ,\ 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\}$
- update:
  1. set:
     $$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2} \qquad \text{(PA)}$$
     $$\tau_t = \min\left\{C\ ,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right\} \qquad \text{(PA-I)}$$
     $$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} \qquad \text{(PA-II)}$$
  2. update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$

Figure 1: Three variants of the Passive-Aggressive algorithm for binary classification.

conditions (Boyd and Vandenberghe, 2004). Setting the partial derivatives of $\mathcal{L}$ with respect to the elements of $\mathbf{w}$ to zero gives,

$$0 \ = \ \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \tau) \ = \ \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t \qquad \Longrightarrow \qquad \mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t . \tag{5}$$

Plugging the above back into Eq. (4) we get,

$$\mathcal{L}(\tau) \ = \ -\frac{1}{2}\tau^2 \|\mathbf{x}_t\|^2 \ + \ \tau\big(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\big).$$

Taking the derivative of $\mathcal{L}(\tau)$ with respect to τ and setting it to zero, we get,

$$0 \ = \ \frac{\partial \mathcal{L}(\tau)}{\partial \tau} \ = \ -\tau \|\mathbf{x}_t\|^2 \ + \ \big(1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\big) \qquad \Longrightarrow \qquad \tau = \frac{1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}.$$

Since we assumed that $\ell_t > 0$ then $\ell_t = 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)$. In summary, we can state a unified update for the case where $\ell_t = 0$ and the case where $\ell_t > 0$ by setting $\tau_t = \ell_t / \|\mathbf{x}_t\|^2$.

As discussed above, the PA algorithm employs an aggressive update strategy by modifying the weight vector by as much as needed to satisfy the constraint imposed by the current example. In certain real-life situations this strategy may also result in undesirable consequences. Consider for instance the common phenomenon of label noise. A mislabeled example may cause the PA algorithm to drastically change its weight vector in the wrong direction. A single mislabeled example can lead to several prediction mistakes on subsequent rounds. To cope with such problems, we present two variations on the PA update that employ gentler update strategies. We adopt the technique previously used to derive soft-margin classifiers (Vapnik, 1998) and introduce a non-negative slack variable ξ into the optimization problem defined in Eq. (2). This variable can be introduced in two different ways. First, we consider the update where the objective function scales linearly with

ξ, namely,

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) \le \xi \text{ and } \xi \ge 0. \tag{6}$$

Here C is a positive parameter which controls the influence of the slack term on the objective function. Specifically, we will show that larger values of C imply a more aggressive update step and we therefore refer to C as the *aggressiveness parameter* of the algorithm. We term the algorithm which results from this update *PA-I* .

Alternatively, we can have the objective function scale quadratically with ξ, resulting in the following constrained optimization problem,

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi^2 \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) \le \xi. \tag{7}$$

Note that the constraint $\xi \ge 0$ which appears in Eq. (6) is no longer necessary since ξ^2 is always non-negative. We term the algorithm which results from this update *PA-II* . As with PA-I , C is a positive parameter which governs the degree to which the update of PA-II is aggressive. The updates of PA-I and PA-II also share the simple closed form $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$, where

$$\tau_t = \min\left\{ C\ ,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2} \right\} \quad \text{(PA-I)} \qquad \text{or} \qquad \tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} \quad \text{(PA-II)}. \tag{8}$$

A detailed derivation of the PA-I and PA-II updates is provided in Appendix A. It is worth noting that the PA-II update is equivalent to increasing the dimension of each $\mathbf{x}_t$ from n to $n+T$, setting $x_{n+t} = \sqrt{1/2C}$, setting the remaining $T-1$ new coordinates to zero, and then using the simple PA update. This technique was previously used to derive noise-tolerant online algorithms in (Klasner and Simon, 1995; Freund and Schapire, 1999). We do not use this observation explicitly in this paper, since it does not lead to a tighter analysis.

Up until now, we have restricted our discussion to linear predictors of the form $\text{sign}(\mathbf{w}\cdot\mathbf{x})$. We can easily generalize any of the algorithms presented in this section using Mercer kernels. Simply note that for all three PA variants,

$$\mathbf{w}_t = \sum_{i=1}^{t-1} \tau_t y_t \mathbf{x}_t,$$

and therefore,

$$\mathbf{w}_t \cdot \mathbf{x}_t = \sum_{i=1}^{t-1} \tau_t y_t (\mathbf{x}_i \cdot \mathbf{x}_t).$$

The inner product on the right hand side of the above can be replaced with a general Mercer kernel $K(\mathbf{x}_i,\mathbf{x}_t)$ without otherwise changing our derivation. Additionally, the formal analysis presented in the next section also holds for any kernel operator.

4. Analysis

In this section we prove *relative* loss bounds for the three variants of the PA algorithm presented in the previous section. Specifically, most of the theorems in this section relate the cumulative squared loss attained by our algorithms on any sequence of examples with the loss attained by an arbitrary

fixed classification function of the form $\text{sign}(\mathbf{u} \cdot \mathbf{x})$ on the same sequence. As previously mentioned, the cumulative squared hinge loss upper bounds the number of prediction mistakes. Our bounds essentially prove that, for any sequence of examples, our algorithms cannot do much worse than the best fixed predictor chosen in hindsight.

To simplify the presentation we use two abbreviations throughout this paper. As before we denote by ℓ_t the instantaneous loss suffered by our algorithm on round t. In addition, we denote by $\ell_t^\star$ the loss suffered by the arbitrary fixed predictor to which we are comparing our performance. Formally, let $\mathbf{u}$ be an arbitrary vector in $\mathbb{R}^n$, and define

$$\ell_t = \ell\big(\mathbf{w}_t; (\mathbf{x}_t, y_t)\big) \qquad \text{and} \qquad \ell_t^\star = \ell\big(\mathbf{u}; (\mathbf{x}_t, y_t)\big). \tag{9}$$

We begin with a technical lemma which facilitates the proofs in this section. With this lemma handy, we then derive loss and mistake bounds for the variants of the PA algorithm presented in the previous section.

Lemma 1 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$ and $y_t \in \{+1, -1\}$ for all t. Let τ_t be as defined by either of the three PA variants given in Fig. 1. Then using the notation given in Eq. (9), the following bound holds for any $\mathbf{u} \in \mathbb{R}^n$,*

$$\sum_{t=1}^{T} \tau_t \left(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 - 2\ell_t^\star\right) \;\leq\; \|\mathbf{u}\|^2.$$

Proof Define Δ_t to be $\|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$. We prove the lemma by summing Δ_t over all t in $1, \ldots, T$ and bounding this sum from above and below. First note that $\sum_t \Delta_t$ is a telescopic sum which collapses to,

$$\begin{aligned} \sum_{t=1}^{T} \Delta_t &= \sum_{t=1}^{T} \left(\|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2\right) \\ &= \|\mathbf{w}_1 - \mathbf{u}\|^2 - \|\mathbf{w}_{T+1} - \mathbf{u}\|^2. \end{aligned}$$

Using the facts that $\mathbf{w}_1$ is defined to be the zero vector and that $\|\mathbf{w}_{T+1} - \mathbf{u}\|^2$ is non-negative, we can upper bound the right-hand side of the above by $\|\mathbf{u}\|^2$ and conclude that,

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2. \tag{10}$$

We now turn to bounding Δ_t from below. If the minimum margin requirement is not violated on round t, i.e. $\ell_t = 0$, then $\tau_t = 0$ and therefore $\Delta_t = 0$. We can therefore focus only on rounds for which $\ell_t > 0$. Using the definition $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \tau_t \mathbf{x}_t$, we can write Δ_t as,

$$\begin{aligned} \Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2 \\ &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + y_t \tau_t \mathbf{x}_t\|^2 \\ &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left(\|\mathbf{w}_t - \mathbf{u}\|^2 + 2\tau_t y_t (\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t + \tau_t^2 \|\mathbf{x}_t\|^2\right) \\ &= -2\tau_t y_t (\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t - \tau_t^2 \|\mathbf{x}_t\|^2. \end{aligned} \tag{11}$$

Since we assumed that $\ell_t > 0$ then $\ell_t = 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)$ or alternatively $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) = 1 - \ell_t$. In addition, the definition of the hinge loss implies that $\ell_t^\star \geq 1 - y_t(\mathbf{u} \cdot \mathbf{x}_t)$, hence $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq 1 - \ell_t^\star$. Using these two facts back in Eq. (11) gives,

$$\begin{aligned} \Delta_t &\geq 2\tau_t \left((1 - \ell_t^\star) - (1 - \ell_t) \right) - \tau_t^2 \|\mathbf{x}_t\|^2 \\ &= \tau_t \left(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 - 2\ell_t^\star \right). \end{aligned} \tag{12}$$

Summing Δ_t over all t and comparing the lower bound of Eq. (12) with the upper bound in Eq. (10) proves the lemma. ■

We first prove a loss bound for the PA algorithm in the separable case. This bound was previously presented by Herbster (2001) and is analogous to the classic mistake bound for the Perceptron algorithm due to Novikoff (1962). We assume that there exists some $\mathbf{u} \in \mathbb{R}^n$ such that $y_t(\mathbf{u} \cdot \mathbf{x}_t) > 0$ for all $t \in \{1, \ldots, T\}$. Without loss of generality we can assume that $\mathbf{u}$ is scaled such that that $y_t(\mathbf{u} \cdot \mathbf{x}_t) \geq 1$ and therefore $\mathbf{u}$ attains a loss of zero on all T examples in the sequence. With the vector $\mathbf{u}$ at our disposal, we prove the following bound on the cumulative squared loss of PA .

Theorem 2 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Assume that there exists a vector $\mathbf{u}$ such that $\ell_t^\star = 0$ for all t. Then, the cumulative squared loss of PA on this sequence of examples is bounded by,*

$$\sum_{t=1}^{T} \ell_t^2 \ \leq \ \|\mathbf{u}\|^2 R^2.$$

Proof Since $\ell_t^\star = 0$ for all t, Lemma 1 implies that,

$$\sum_{t=1}^{T} \tau_t \left(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 \right) \ \leq \ \|\mathbf{u}\|^2. \tag{13}$$

Using the definition of τ_t for the PA algorithm in the left-hand side of the above gives,

$$\sum_{t=1}^{T} \ell_t^2 / \|\mathbf{x}_t\|^2 \ \leq \ \|\mathbf{u}\|^2.$$

Now using the fact that $\|\mathbf{x}_t\|^2 \leq R^2$ for all t, we get,

$$\sum_{t=1}^{T} \ell_t^2 / R^2 \ \leq \ \|\mathbf{u}\|^2.$$

Multiplying both sides of this inequality by R^2 gives the desired bound. ■

The remaining bounds we prove in this section do not depend on a separability assumption. In contrast to the assumptions of Thm. 2, the vector $\mathbf{u}$ which appears in the theorems below is an arbitrary vector in $\mathbb{R}^n$ and not necessarily a perfect separator. The first of the following theorems bounds the cumulative squared loss attained by the PA algorithm in the special case where all of

the instances in the input sequence are normalized so that $\|\mathbf{x}_t\|^2 = 1$. Although this assumption is somewhat restrictive, it is often the case in many practical applications of classification that the instances are normalized. For instance, certain kernel operators, such as the Gaussian kernel, imply that all input instances have a unit norm. See for example (Cristianini and Shawe-Taylor, 2000).

Theorem 3 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| = 1$ for all t. Then for any vector $\mathbf{u} \in \mathbb{R}^n$ the cumulative squared loss of PA on this sequence of examples is bounded from above by,*

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; \left(\|\mathbf{u}\| + 2\sqrt{\textstyle\sum_{t=1}^{T} (\ell_t^\star)^2} \right)^2 .$$

Proof In the special case where $\|\mathbf{x}_t\|^2 = 1$, τ_t and ℓ_t are equal. Therefore, Lemma 1 gives us that,

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; \|\mathbf{u}\|^2 + 2 \sum_{t=1}^{T} \ell_t \ell_t^\star .$$

Using the Cauchy-Schwartz inequality to upper bound the right-hand side of the above inequality, and denoting

$$L_T = \sqrt{\textstyle\sum_{t=1}^{T} \ell_t^2} \quad \text{and} \quad U_T = \sqrt{\textstyle\sum_{t=1}^{T} (\ell_t^\star)^2}, \tag{14}$$

we get that $L_T^2 \leq \|\mathbf{u}\|^2 + 2 L_T U_T$. The largest value of L_T for which this inequality is satisfied is the larger of the two values for which this inequality holds with equality. That is, to obtain an upper bound on L_T we need to find the largest root of the second degree polynomial $L_T^2 - 2 U_T L_T - \|\mathbf{u}\|^2$, which is,

$$U_T + \sqrt{U_T^2 + \|\mathbf{u}\|^2}.$$

Using the fact that $\sqrt{\alpha + \beta} \leq \sqrt{\alpha} + \sqrt{\beta}$, we conclude that

$$L_T \;\leq\; \|\mathbf{u}\| + 2 U_T . \tag{15}$$

Taking the square of both sides of this inequality and plugging in the definitions of L_T and U_T from Eq. (14) gives the desired bound. ■

Next we turn to the analysis of PA-I . The following theorem does not provide a loss bound but rather a mistake bound for the PA-I algorithm. That is, we prove a direct bound on the number of times $y_t \neq \mathrm{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$ without using $\sum \ell_t^2$ as a proxy.

Theorem 4 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\| \leq R$ for all t. Then, for any vector $\mathbf{u} \in \mathbb{R}^n$, the number of prediction mistakes made by PA-I on this sequence of examples is bounded from above by,*

$$\max\left\{R^2, 1/C\right\} \left(\|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_t^\star \right),$$

where C is the aggressiveness parameter provided to PA-I (Fig. 1) .

Proof If PA-I makes a prediction mistake on round t then $\ell_t \geq 1$. Using our assumption that $\|\mathbf{x}_t\|^2 \leq R^2$ and the definition $\tau_t = \min\{\ell_t/\|\mathbf{x}_t\|^2, C\}$, we conclude that if a prediction mistake occurs then it holds that,

$$\min\{1/R^2, C\} \;\leq\; \tau_t \ell_t .$$

Let M denote the number of prediction mistakes made on the entire sequence. Since $\tau_t \ell_t$ is always non-negative, it holds that,

$$\min\{1/R^2, C\}\, M \;\leq\; \sum_{t=1}^{T} \tau_t \ell_t . \tag{16}$$

Again using the definition of τ_t, we know that $\tau_t \ell_t^\star \leq C \ell_t^\star$ and that $\tau_t \|\mathbf{x}_t\|^2 \leq \ell_t$. Plugging these two inequalities into Lemma 1 gives,

$$\sum_{t=1}^{T} \tau_t \ell_t \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_t^\star . \tag{17}$$

Combining Eq. (16) with Eq. (17), we conclude that,

$$\min\{1/R^2, C\}\, M \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_t^\star .$$

The theorem follows from multiplying both sides of the above by $\max\{R^2, 1/C\}$. ■

Finally, we turn to the analysis of PA-II . As before, the proof of the following theorem is based on Lemma 1.

Theorem 5 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_t)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \{+1, -1\}$ and $\|\mathbf{x}_t\|^2 \leq R^2$ for all t. Then for any vector $\mathbf{u} \in \mathbb{R}^n$ it holds that the cumulative squared loss of PA-II on this sequence of examples is bounded by,*

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; \left(R^2 + \frac{1}{2C}\right) \left(\|\mathbf{u}\|^2 \;+\; 2C \sum_{t=1}^{T} (\ell_t^\star)^2\right),$$

where C is the aggressiveness parameter provided to PA-II (Fig. 1) .

Proof Recall that Lemma 1 states that,

$$\|\mathbf{u}\|^2 \;\geq\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|^2 - 2\tau_t \ell_t^\star\right).$$

Defining $\alpha = 1/\sqrt{2C}$, we subtract the non-negative term $(\alpha\tau_t - \ell_t^\star/\alpha)^2$ from each summand on the right-hand side of the above inequality, to get

$$\begin{aligned}
\|\mathbf{u}\|^2 \;&\geq\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|^2 - 2\tau_t \ell_t^\star - (\alpha\tau_t - \ell_t^\star/\alpha)^2\right) \\
&=\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 \|\mathbf{x}_t\|^2 - 2\tau_t \ell_t^\star - \alpha^2\tau_t^2 + 2\tau_t \ell_t^\star - (\ell_t^\star)^2/\alpha^2\right) \\
&=\; \sum_{t=1}^{T} \left(2\tau_t \ell_t - \tau_t^2 (\|\mathbf{x}_t\|^2 + \alpha^2) - (\ell_t^\star)^2/\alpha^2\right).
\end{aligned}$$

Plugging in the definition of α, we obtain the following lower bound,

$$\|\mathbf{u}\|^2 \;\geq\; \sum_{t=1}^{T}\left(2\tau_t\ell_t - \tau_t^2\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right) - 2C(\ell_t^\star)^2\right).$$

Using the definition $\tau_t = \ell_t/(\|\mathbf{x}_t\|^2 + 1/(2C))$, we can rewrite the above as,

$$\|\mathbf{u}\|^2 \;\geq\; \sum_{t=1}^{T}\left(\frac{\ell_t^2}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}} - 2C(\ell_t^\star)^2\right).$$

Replacing $\|\mathbf{x}_t\|^2$ with its upper bound of R^2 and rearranging terms gives the desired bound. ■

We conclude this section with a brief comparison of our bounds to previously published bounds for the Perceptron algorithm. As mentioned above, the bound in Thm. 2 is equal to the bound of Novikoff (1962) for the Perceptron in the separable case. However, Thm. 2 bounds the cumulative squared hinge loss of PA , whereas Novikoff's bound is on the number of prediction mistakes. Gentile (2002) proved a mistake bound for the Perceptron in the nonseparable case which can be compared to our mistake bound for PA-I in Thm. 4. Using our notation from Thm. 4, Gentile bounds the number of mistakes made by the Perceptron by,

$$\tfrac{R^2\|\mathbf{u}\|^2}{2} \;+\; \textstyle\sum_{t=1}^{T}\ell_t^\star \;+\; \sqrt{R^2\|\mathbf{u}\|^2\textstyle\sum_{t=1}^{T}\ell_t^\star + \left(\tfrac{R^2\|\mathbf{u}\|^2}{2}\right)^2}.$$

At the price of a slightly loosening this bound, we can use the inequality $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$ to get the simpler bound,

$$R^2\|\mathbf{u}\|^2 \;+\; \textstyle\sum_{t=1}^{T}\ell_t^\star \;+\; R\|\mathbf{u}\|\sqrt{\textstyle\sum_{t=1}^{T}\ell_t^\star}.$$

With $C = 1/R^2$, our bound in Thm. 4 becomes,

$$R^2\|\mathbf{u}\|^2 \;+\; 2\sum_{t=1}^{T}\ell_t^\star.$$

Thus, our bound is inferior to Gentile's when $R\|\mathbf{u}\| < \sqrt{\sum_{t=1}^{T}\ell_t^\star}$, and even then by a factor of at most 2.

The loss bound for PA-II in Thm. 5 can be compared with the bound of Freund and Schapire (1999) for the Perceptron algorithm. Using the notation defined in Thm. 5, Freund and Schapire bound the number of incorrect predictions made by the Perceptron by,

$$\left(R\|\mathbf{u}\| + \sqrt{\textstyle\sum_{t=1}^{T}(\ell_t^\star)^2}\right)^2.$$

It can be easily verified that the bound for the PA-II algorithm given in Thm. 5 exactly equals the above bound of Freund and Schapire when C is set to $\|\mathbf{u}\|/(2R\sqrt{\sum_t(\ell_t^\star)^2})$. Moreover, this is the optimal choice of C. However, we bound the cumulative squared hinge-loss of PA-II whereas the bound of Freund and Schapire is on the number of mistakes.

5. Regression

In this section we show that the algorithms described in Sec. 3 can be modified to deal with online regression problems. In the regression setting, every instance $\mathbf{x}_t$ is associated with a real target value $y_t \in \mathbb{R}$, which the online algorithm tries to predict. On every round, the algorithm receives an instance $\mathbf{x}_t \in \mathbb{R}^n$ and predicts a target value $\hat{y}_t \in \mathbb{R}$ using its internal regression function. We focus on the class of linear regression functions, that is, $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$ where $\mathbf{w}_t$ is the incrementally learned vector. After making a prediction, the algorithm is given the true target value y_t and suffers an instantaneous loss. We use the ε-insensitive hinge loss function:

$$\ell_\varepsilon\big(\mathbf{w};(\mathbf{x},y)\big) \;=\; \begin{cases} 0 & |\mathbf{w}\cdot\mathbf{x}-y| \le \varepsilon \\ |\mathbf{w}\cdot\mathbf{x}-y|-\varepsilon & \text{otherwise} \end{cases}, \tag{18}$$

where ε is a positive parameter which controls the sensitivity to prediction mistakes. This loss is zero when the predicted target deviates from the true target by less than ε and otherwise grows linearly with $|\hat{y}_t - y_t|$. At the end of every round, the algorithm uses $\mathbf{w}_t$ and the example $(\mathbf{x}_t, y_t)$ to generate a new weight vector $\mathbf{w}_{t+1}$, which will be used to extend the prediction on the next round.

We now describe how the various PA algorithms from Sec. 3 can be adapted to learn regression problems. As in the case of classification, we initialize $\mathbf{w}_1$ to $(0,\ldots,0)$. On each round, the PA regression algorithm sets the new weight vector to be,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell_\varepsilon\big(\mathbf{w};(\mathbf{x}_t,y_t)\big) = 0, \tag{19}$$

In the binary classification setting, we gave the PA update the geometric interpretation of projecting $\mathbf{w}_t$ onto the linear half-space defined by the constraint $\ell\big(\mathbf{w};(\mathbf{x}_t,y_t)\big) = 0$. For regression problems, the set $\{\mathbf{w}\in\mathbb{R}^n : \ell_\varepsilon(\mathbf{w},\mathbf{z}_t) = 0\}$ is not a half-space but rather a hyper-slab of width 2ε. Geometrically, the PA algorithm for regression projects $\mathbf{w}_t$ onto this hyper-slab at the end of every round. Using the shorthand $\ell_t = \ell_\varepsilon(\mathbf{w}_t;(\mathbf{x}_t,y_t))$, the update given in Eq. (19) has a closed form solution similar to that of the classification PA algorithm of the previous section, namely,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \operatorname{sign}(y_t - \hat{y}_t)\tau_t\mathbf{x}_t \qquad \text{where} \qquad \tau_t = \ell_t/\|\mathbf{x}_t\|^2.$$

We can also obtain the PA-I and PA-II variants for online regression by introducing a slack variable into the optimization problem in Eq. (19), as we did for classification in Eq. (6) and Eq. (7). The closed form solution for these updates also comes out to be $\mathbf{w}_{t+1} = \mathbf{w}_t + \operatorname{sign}(y_t - \hat{y}_t)\tau_t\mathbf{x}_t$ where τ_t is defined as in Eq. (8). The derivations of these closed-form updates are almost identical to that of the classification problem in Sec. 3.

We now turn to the analysis of the three PA regression algorithms described above. We would like to show that the analysis given in Sec. 4 for the classification algorithms also holds for their regression counterparts. To do so, it suffices to show that Lemma 1 still holds for regression problems. After obtaining a regression version of Lemma 1, regression versions of Thm. 2 through Thm. 5 follow as immediate corollaries.

Lemma 6 *Let $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ be an arbitrary sequence of examples, where $\mathbf{x}_t\in\mathbb{R}^n$ and $y_t\in\mathbb{R}$ for all t. Let τ_t be as defined in either of the three PA variants for regression problems. Then using the notation given in Eq. (9), the following bound holds for any $\mathbf{u}\in\mathbb{R}^n$,*

$$\sum_{t=1}^{T}\tau_t\left(2\ell_t - \tau_t\|\mathbf{x}_t\|^2 - 2\ell_t^\star\right) \;\le\; \|\mathbf{u}\|^2.$$

Proof The proof of this lemma follows that of Lemma 1 and therefore subtleties which were discussed in detail in that proof are omitted here. Again, we use the definition

$$\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$$

and the same argument used in Lemma 1 implies that,

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2,$$

We focus our attention on bounding Δ_t from below on those rounds where $\Delta_t \neq 0$. Using the recursive definition of $\mathbf{w}_{t+1}$, we rewrite Δ_t as,

$$\begin{aligned} \Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + \text{sign}(y_t - \hat{y}_t)\tau_t \mathbf{x}_t\|^2 \\ &= -\text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{w}_t - \mathbf{u}) \cdot \mathbf{x}_t - \tau_t^2 \|\mathbf{x}_t\|^2 \end{aligned}$$

We now add and subtract the term $\text{sign}(y_t - \hat{y}_t)2\tau_t y_t$ from the right-hand side above to get the bound,

$$\Delta_t \;\geq\; -\text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{w}_t \cdot \mathbf{x}_t - y_t) + \text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{u} \cdot \mathbf{x}_t - y_t) - \tau_t^2 \|\mathbf{x}_t\|^2. \tag{20}$$

Since $\mathbf{w}_t \cdot \mathbf{x}_t = \hat{y}_t$, we have that $-\text{sign}(y_t - \hat{y}_t)(\mathbf{w}_t \cdot \mathbf{x}_t - y_t) = |\mathbf{w}_t \cdot \mathbf{x}_t - y_t|$. We only need to consider the case where $\Delta_t \neq 0$, so $\ell_t = |\mathbf{w}_t \cdot \mathbf{x}_t - y_t| - \varepsilon$ and we can rewrite the bound in Eq. (20) as,

$$\Delta_t \;\geq\; 2\tau_t(\ell_t + \varepsilon) + \text{sign}(y_t - \hat{y}_t)2\tau_t(\mathbf{u} \cdot \mathbf{x}_t - y_t) - \tau_t^2 \|\mathbf{x}_t\|^2.$$

We also know that $\text{sign}(y_t - \hat{y}_t)(\mathbf{u} \cdot \mathbf{x}_t - y_t) \geq -|\mathbf{u} \cdot \mathbf{x}_t - y_t|$ and that $-|\mathbf{u} \cdot \mathbf{x}_t - y_t| \geq -(\ell_t^\star + \varepsilon)$. This enables us to further bound,

$$\Delta_t \;\geq\; 2\tau_t(\ell_t + \varepsilon) - 2\tau_t(\ell_t^\star + \varepsilon) - \tau_t^2 \|\mathbf{x}_t\|^2 \;=\; \tau_t(2\ell_t - \tau_t \|\mathbf{x}_t\|^2 - 2\ell_t^\star).$$

Summing the above over all t and comparing to the upper bound discussed in the beginning of this proof proves the lemma. ■

6. Uniclass Prediction

In this section we present PA algorithms for the uniclass prediction problem. This task involves predicting a sequence of vectors $\mathbf{y}_1, \mathbf{y}_2, \cdots$ where $\mathbf{y}_t \in \mathbb{R}^n$. Uniclass prediction is fundamentally different than classification and regression as the algorithm makes predictions without first observing any external input (such as the instance $\mathbf{x}_t$). Specifically, the algorithm maintains in its memory a vector $\mathbf{w}_t \in \mathbb{R}^n$ and simply predicts the next element of the sequence to be $\mathbf{w}_t$. After extending this prediction, the next element in the sequence is revealed and an instantaneous loss is suffered. We measure loss using the following ε-insensitive loss function:

$$\ell_\varepsilon(\mathbf{w}; \mathbf{y}) \;=\; \begin{cases} 0 & \|\mathbf{w} - \mathbf{y}\| \leq \varepsilon \\ \|\mathbf{w} - \mathbf{y}\| - \varepsilon & \text{otherwise} \end{cases}. \tag{21}$$

As in the regression setting, ε is a positive user-defined parameter. If the prediction is within ε of the true sequence element then no loss is suffered. Otherwise the loss is proportional to the Euclidean

distance between the prediction and the true vector. At the end of each round $\mathbf{w}_t$ is updated in order to have a potentially more accurate prediction on where the next element in the sequence will fall. Equivalently, we can think of uniclass prediction as the task of finding a center-point $\mathbf{w}$ such that as many vectors in the sequence fall within a radius of ε from $\mathbf{w}$. At the end of this section we discuss a generalization of this problem, where the radius ε is also determined by the algorithm.

As before, we initialize $\mathbf{w}_1 = (0,\ldots,0)$. Beginning with the PA algorithm, we define the update for the uniclass prediction algorithm to be,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell_\varepsilon(\mathbf{w};\mathbf{y}_t) = 0, \tag{22}$$

Geometrically, $\mathbf{w}_{t+1}$ is set to be the projection of $\mathbf{w}_t$ onto a ball of radius ε about $\mathbf{y}_t$. We now show that the closed form solution of this optimization problem turns out to be,

$$\mathbf{w}_{t+1} \;=\; \left(1-\frac{\ell_t}{\|\mathbf{w}_t-\mathbf{y}_t\|}\right)\mathbf{w}_t \;+\; \left(\frac{\ell_t}{\|\mathbf{w}_t-\mathbf{y}_t\|}\right)\mathbf{y}_t. \tag{23}$$

First, we rewrite the above equation and express $\mathbf{w}_{t+1}$ by,

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \frac{\mathbf{y}_t-\mathbf{w}_t}{\|\mathbf{y}_t-\mathbf{w}_t\|}, \tag{24}$$

where $\tau_t = \ell_t$. In the Uniclass problem the KKT conditions are both sufficient and necessary for optimality. Therefore, we prove that Eq. (24) is the minimizer of Eq. (22) by verifying that the KKT conditions indeed hold. The Lagrangian of Eq. (22) is,

$$\mathcal{L}(\mathbf{w},\tau) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + \tau\left(\|\mathbf{w}-\mathbf{y}_t\|-\varepsilon\right), \tag{25}$$

where $\tau \geq 0$ is a Lagrange multiplier. Differentiating with respect to the elements of $\mathbf{w}$ and setting these partial derivatives to zero, we get the first KKT condition, stating that at the optimum $(\mathbf{w},\tau)$ must satisfy the equality,

$$0 \;=\; \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\tau) \;=\; \mathbf{w}-\mathbf{w}_t+\tau\frac{\mathbf{w}-\mathbf{y}_t}{\|\mathbf{w}-\mathbf{y}_t\|}. \tag{26}$$

In addition, an optimal solution must satisfy the conditions $\tau \geq 0$ and,

$$\tau\left(\|\mathbf{w}-\mathbf{y}_t\|-\varepsilon\right) \;=\; 0. \tag{27}$$

Clearly, $\tau_t \geq 0$. Therefore, to show that $\mathbf{w}_{t+1}$ is the optimum of Eq. (22) it suffices to prove that $(\mathbf{w}_{t+1},\tau_t)$ satisfies Eq. (26) and Eq. (27). These equalities trivially hold if $\ell_t = 0$ and therefore from now on we assume that $\ell_t > 0$. Plugging the values $\mathbf{w} = \mathbf{w}_{t+1}$ and $\tau = \tau_t$ in the right-hand side of Eq. (26) gives,

$$\mathbf{w}_{t+1}-\mathbf{w}_t+\tau_t\frac{\mathbf{w}_{t+1}-\mathbf{y}_t}{\|\mathbf{w}_{t+1}-\mathbf{y}_t\|} \;=\; \tau_t\left(\frac{\mathbf{y}_t-\mathbf{w}_t}{\|\mathbf{y}_t-\mathbf{w}_t\|}+\frac{\mathbf{w}_{t+1}-\mathbf{y}_t}{\|\mathbf{w}_{t+1}-\mathbf{y}_t\|}\right). \tag{28}$$

Note that,

$$\begin{aligned}\mathbf{w}_{t+1}-\mathbf{y}_t &= \mathbf{w}_t+\tau_t\frac{\mathbf{y}_t-\mathbf{w}_t}{\|\mathbf{y}_t-\mathbf{w}_t\|}-\mathbf{y}_t \;=\; (\mathbf{w}_t-\mathbf{y}_t)\left(1-\tau_t\frac{1}{\|\mathbf{y}_t-\mathbf{w}_t\|}\right)\\ &= \frac{\mathbf{w}_t-\mathbf{y}_t}{\|\mathbf{w}_t-\mathbf{y}_t\|}\left(\|\mathbf{w}_t-\mathbf{y}_t\|-\tau_t\right) \;=\; \frac{\varepsilon}{\|\mathbf{w}_t-\mathbf{y}_t\|}(\mathbf{w}_t-\mathbf{y}_t).\end{aligned} \tag{29}$$

Combining Eq. (29) with Eq. (28) we get that,

$$\mathbf{w}_{t+1} - \mathbf{w}_t + \tau_t \frac{\mathbf{w}_{t+1} - \mathbf{y}_t}{\|\mathbf{w}_{t+1} - \mathbf{y}_t\|} \;=\; 0,$$

and thus Eq. (26) holds for $(\mathbf{w}_{t+1}, \tau_t)$. Similarly,

$$\|\mathbf{w}_{t+1} - \mathbf{y}_t\| - \varepsilon \;=\; \varepsilon - \varepsilon \;=\; 0,$$

and thus Eq. (27) also holds. In summary, we have shown that the KKT optimality conditions hold for $(\mathbf{w}_{t+1}, \tau_t)$ and therefore Eq. (24) gives the desired closed-form update.

To obtain uniclass versions of PA-I and PA-II , we add a slack variable to the optimization problem in Eq. (22) in the same way as we did in Eq. (6) and Eq. (7) for the classification algorithms. Namely, the update for PA-I is defined by,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi \quad \text{s.t.} \quad \|\mathbf{w} - \mathbf{y}_t\| \leq \varepsilon + \xi, \quad \xi \geq 0, \tag{30}$$

and the update for PA-II is,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 + C\xi^2 \quad \text{s.t.} \quad \|\mathbf{w} - \mathbf{y}_t\| \leq \varepsilon + \xi.$$

The closed form for these updates can be derived using the same technique as we used for deriving the PA update. The final outcome is that both PA-I and PA-II share the form of update given in Eq. (24), with τ_t set to be,

$$\tau_t = \min\{\, C \,,\, \ell_t \,\} \quad \text{(PA-I)} \qquad \text{or} \qquad \tau_t = \frac{\ell_t}{1 + \frac{1}{2C}} \quad \text{(PA-II)}.$$

We can extend the analysis of the three PA variants from Sec. 4 to the case of uniclass prediction. We do so by proving a uniclass version of Lemma 1. After proving this lemma, we discuss an additional technical difficulty which needs to be addressed so that Thm. 2 through Thm. 5 carry over smoothly to the uniclass case.

Lemma 7 *Let $\mathbf{y}_1, \ldots, \mathbf{y}_T$ be an arbitrary sequence of vectors, where $\mathbf{y}_t \in \mathbb{R}^n$ for all t. Let τ_t be as defined in either of the three PA variants for uniclass prediction. Then using the notation given in Eq. (9), the following bound holds for any $\mathbf{u} \in \mathbb{R}^n$,*

$$\sum_{t=1}^{T} \tau_t \left(2\ell_t - \tau_t - 2\ell_t^\star\right) \;\leq\; \|\mathbf{u}\|^2.$$

Proof We prove this lemma in much the same way as we did Lemma 1. We again use the definition, $\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$, along with the fact stated in Eq. (10) that

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2.$$

We now focus our attention on bounding Δ_t from below on those rounds where $\Delta_t \neq 0$. Using the recursive definition of $\mathbf{w}_{t+1}$, we rewrite Δ_t as,

$$\begin{aligned}
\Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left\| \left(1 - \frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) \mathbf{w}_t + \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) \mathbf{y}_t - \mathbf{u} \right\|^2 \\
&= \|\mathbf{w}_t - \mathbf{u}\|^2 - \left\| (\mathbf{w}_t - \mathbf{u}) + \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{y}_t - \mathbf{w}_t) \right\|^2 \\
&= -2 \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{w}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t) - \tau_t^2 .
\end{aligned}$$

We now add and subtract $\mathbf{y}_t$ from the term $(\mathbf{w}_t - \mathbf{u})$ above to get,

$$\begin{aligned}
\Delta_t &= -2 \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{w}_t - \mathbf{y}_t + \mathbf{y}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t) - \tau_t^2 \\
&= 2\tau_t \|\mathbf{w}_t - \mathbf{y}_t\| - 2 \left(\frac{\tau_t}{\|\mathbf{w}_t - \mathbf{y}_t\|}\right) (\mathbf{y}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t) - \tau_t^2 .
\end{aligned}$$

Now, using the Cauchy-Schwartz inequality on the term $(\mathbf{y}_t - \mathbf{u}) \cdot (\mathbf{y}_t - \mathbf{w}_t)$, we can bound,

$$\Delta_t \;\geq\; 2\tau_t \|\mathbf{w}_t - \mathbf{y}_t\| - 2\tau_t \|\mathbf{y}_t - \mathbf{u}\| - \tau_t^2 .$$

We now add and subtract $2\tau_t \varepsilon$ from the right-hand side of the above, to get,

$$\Delta_t \;\geq\; 2\tau_t \left(\|\mathbf{w}_t - \mathbf{y}_t\| - \varepsilon\right) - 2\tau_t \left(\|\mathbf{y}_t - \mathbf{u}\| - \varepsilon\right) - \tau_t^2 .$$

Since we are dealing with the case where $\ell_t > 0$, it holds that $\ell_t = \|\mathbf{w}_t - \mathbf{y}_t\| - \varepsilon$. By definition, $\ell_t^\star \geq \|\mathbf{u} - \mathbf{y}_t\| - \varepsilon$. Using these two facts, we get,

$$\Delta_t \;\geq\; 2\tau_t \ell_t - 2\tau_t \ell_t^\star - \tau_t^2 .$$

Summing the above inequality over all t and comparing the result to the upper bound in Eq. (10) gives the bound stated in the lemma. ■

As mentioned above, there remains one more technical obstacle which stands in the way of applying Thm. 2 through Thm. 5 to the uniclass case. This difficulty stems from the fact $\mathbf{x}_t$ is not defined in the uniclass whereas the term $\|\mathbf{x}\|^2$ appears in the theorems. This issue is easily resolved by setting $\mathbf{x}_t$ in the uniclass case to be an arbitrary vector of a unit length, namely $\|\mathbf{x}_t\|^2 = 1$. This technical modification enables us to write τ_t as $\ell_t / \|\mathbf{x}_t\|^2$ in the uniclass PA algorithm, as in the classification case. Similarly, τ_t can be defined as in the classification case for PA-I and PA-II . Now Thm. 2 through Thm. 5 can be applied verbatim to the uniclass PA algorithms.

Learning the Radius of the Uniclass Predictor In the derivation above we made the simplifying assumption that ε, the radius of our uniclass predictor, is fixed beforehand and that the online algorithm can only move its center, $\mathbf{w}$. We now show that learning ε and $\mathbf{w}$ in parallel is no harder than learning $\mathbf{w}$ alone. We do so by using a simple reduction argument. For technical reasons, we still require an upper bound on ε, which we denote by B. Although B is specified ahead of time, it can

be arbitrarily large and does not appear in our analysis. Typically, we will think of B as being far greater than any conceivable value of ε. Our goal is now to incrementally find $\mathbf{w}_t$ and ε_t such that,

$$\|\mathbf{w}_t - \mathbf{y}_t\| \leq \varepsilon_t, \tag{31}$$

as often as possible. Additionally, we would like ε_t to stay relatively small, since an extremely large value of ε_t would solve the problem in a trivial way. We do so by reducing this problem to a different uniclass problem where the radius is fixed and where $\mathbf{y}_t$ is in $\mathbb{R}^{n+1}$. That is, by adding an additional dimension to the problem, we can learn ε using the same machinery developed for fixed-radius uniclass problems. The reduction stems from the observation that Eq. (31) can be written equivalently as,

$$\|\mathbf{w}_t - \mathbf{y}_t\|^2 + (B^2 - \varepsilon_t^2) \;\leq\; B^2. \tag{32}$$

If we were to concatenate a 0 to the end of every $\mathbf{y}_t$ (thus increasing its dimension to $n+1$) and if we considered the $n+1$'th coordinate of $\mathbf{w}_t$ to be equivalent to $\sqrt{B^2 - \varepsilon_t^2}$, then Eq. (32) simply becomes $\|\mathbf{w}_t - \mathbf{y}_t\|^2 \;\leq\; B^2$. Our problem has reduced to a fixed-radius uniclass problem where the radius is set to B. $w_{1,n+1}$ should be initialized to B, which is equivalent to initializing $\varepsilon_1 = 0$. On each round, ε_t can be extracted from $\mathbf{w}_t$ by,

$$\varepsilon_t \;=\; \sqrt{B^2 - w_{t,n+1}^2}.$$

Since $w_{t+1,n+1}$ is defined to be a convex combination of $w_{t,n+1}$ and $y_{t,n+1}$ (where the latter equals zero), then $w_{t,n+1}$ is bounded in $(0,B]$ for all t and can only decrease from round to round. This means that the radius ε_t is always well defined and can only increase with t. Since the radius is initialized to zero and is now one of the learned parameters, the algorithm has a natural tendency to favor small radii. Let $\mathbf{u}$ denote the center of a fixed uniclass predictor and let ε denote its radius. Then the reduction described above enables us to prove loss bounds similar to those presented in Sec. 4, with $\|\mathbf{u}\|^2$ replaced by $\|\mathbf{u}\|^2 + \varepsilon^2$.

7. Multiclass Problems

We now address more complex decision problems. We first adapt the binary classification algorithms described in Sec. 3 to the task of *multiclass multilabel* classification. In this setting, every instance is associated with a set of labels Y_t. For concreteness we assume that there are k different possible labels and denote the set of all possible labels by $\mathcal{Y} = \{1,\ldots,k\}$. For every instance $\mathbf{x}_t$, the set of relevant labels Y_t is therefore a subset of $\mathcal{Y}$. We say that label y is *relevant* to the instance $\mathbf{x}_t$ if $y \in Y_t$. This setting is often discussed in text categorization applications (see for instance (Schapire and Singer, 2000)) where $\mathbf{x}_t$ represents a document and Y_t is the set of topics which are relevant to the document and is chosen from a predefined collection of topics. The special case where there is only a *single* relevant topic for each instance is typically referred to as *multiclass single-label* classification or multiclass categorization for short. As discussed below, our adaptation of the PA variants to multiclass multilabel settings encompasses the single-label setting as a special case.

As in the previous sections, the algorithm receives instances $\mathbf{x}_1, \mathbf{x}_2, \ldots$ in a sequential manner where each $\mathbf{x}_t$ belongs to an instance space $\mathcal{X}$. Upon receiving an instance, the algorithm outputs a score for each of the k labels in $\mathcal{Y}$. That is, the algorithm's prediction is a vector in $\mathbb{R}^k$ where each element in the vector corresponds to the score assigned to the respective label. This form of prediction is often referred to as label ranking. Predicting a label ranking is more general and

flexible than predicting the set of relevant labels Y_t. Special purpose learning algorithms such as AdaBoost.MR (Schapire and Singer, 1998) and adaptations of support vector machines (Crammer and Singer, 2003a) have been devised for the task of label ranking. Here we describe a reduction from online label ranking to online binary classification that deems label ranking as simple as binary prediction. We note that in the case of multiclass single-label classification, the prediction of the algorithm is simply set to be the label with the highest score.

For a pair of labels $r,s \in \mathcal{Y}$, if the score assigned by the algorithm to label r is greater than the score assigned to label s, we say that label r is *ranked* higher than label s. The goal of the algorithm is to rank every relevant label above every irrelevant label. Assume that we are provided with a set of d features $\phi_1,\ldots,\phi_d$ where each feature ϕ_j is a mapping from $\mathcal{X}\times\mathcal{Y}$ to the reals. We denote by $\Phi(\mathbf{x},y) = (\phi_1(\mathbf{x},y),\ldots,\phi_d(\mathbf{x},y))$ the vector formed by concatenating the outputs of the features, when each feature is applied to the pair $(\mathbf{x},y)$. The label ranking function discussed in this section is parameterized by a weight vector, $\mathbf{w}\in\mathbb{R}^d$. On round t, the prediction of the algorithm is the k-dimensional vector,

$$\Big((\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,1)),\ldots,(\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,k))\Big).$$

We motivate our construction with an example from the domain of text categorization. We describe a variant of the *Term Frequency - Inverse Document Frequency* (TF-IDF) representation of documents (Rocchio, 1971; Salton and Buckley, 1988). Each feature ϕ_j corresponds to a different word, denoted μ_j. Given a corpus of documents S, for every $\mathbf{x}\in S$ and for every potential topic y, the feature $\phi_j(\mathbf{x},y)$ is defined to be,

$$\phi_j(\mathbf{x},y) \;=\; \mathrm{TF}(\mu_j,\mathbf{x})\cdot\log\left(\frac{|S|}{\mathrm{DF}(\mu_j,y)}\right),$$

where $\mathrm{TF}(\mu_j,\mathbf{x})$ is the number of times μ_j appears in $\mathbf{x}$ and $\mathrm{DF}(\mu_j,y)$ is the number of times μ_j appears in all of the documents in S which are *not* labeled by y. The value ϕ_j grows in proportion to the frequency of μ_j in the document $\mathbf{x}$ but is dampened if μ_j is a frequent word for topics other than y. In the context of this paper, the important point is that each feature is label-dependent.

After making its prediction (a ranking of the labels), the algorithm receives the correct set of relevant labels Y_t. We define the *margin* attained by the algorithm on round t for the example $(\mathbf{x}_t,Y_t)$ as,

$$\gamma\big(\mathbf{w}_t;(\mathbf{x}_t,Y_t)\big) \;=\; \min_{r\in Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,r) \;-\; \max_{s\notin Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,s).$$

This definition generalizes the definition of margin for binary classification and was employed by both single-label and multilabel learning algorithms for support vector machines (Vapnik, 1998; Weston and Watkins, 1999; Elisseeff and Weston, 2001; Crammer and Singer, 2003a). In words, the margin is the difference between the score of the lowest ranked relevant label and the score of the highest ranked irrelevant label. The margin is positive only if all of the relevant labels are ranked higher than all of the irrelevant labels. However, in the spirit of binary classification, we are not satisfied by a mere positive margin as we require the margin of every prediction to be at least 1. After receiving Y_t, we suffer an instantaneous loss defined by the following hinge-loss function,

$$\ell_{\mathrm{MC}}\big(\mathbf{w};(\mathbf{x},Y)\big) \;=\; \begin{cases} 0 & \gamma\big(\mathbf{w};(\mathbf{x},Y)\big)\geq 1 \\ 1-\gamma\big(\mathbf{w};(\mathbf{x},Y)\big) & \text{otherwise} \end{cases}. \tag{33}$$

As in the previous sections, we use ℓ_t as an abbreviation for $\ell_{\text{MC}}\big(\mathbf{w}_t;(\mathbf{x}_t,Y_t)\big)$. If an irrelevant label is ranked higher than a relevant label, then ℓ_t^2 attains a value greater than 1. Therefore, $\sum_{t=1}^{T}\ell_t^2$ upper bounds the number of multiclass prediction mistakes made on rounds 1 through T.

One way of updating the weight vector $\mathbf{w}_t$ is to mimic the derivation of the PA algorithm for binary classification defined in Sec. 3 and to set

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d}\frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell_{\text{MC}}(\mathbf{w};(\mathbf{x}_t,Y_t))=0. \tag{34}$$

Satisfying the single constraint in the optimization problem above is equivalent to satisfying the following set of linear constraints,

$$\forall r\in Y_t \quad \forall s\notin Y_t \qquad \mathbf{w}\cdot\Phi(\mathbf{x}_t,r)-\mathbf{w}\cdot\Phi(\mathbf{x}_t,s)\;\geq\;1. \tag{35}$$

However, instead of attempting to satisfy all of the $|\mathcal{Y}_t|\times(k-|\mathcal{Y}_t|)$ constraints above we focus only on the single constraint which is violated the most by $\mathbf{w}_t$. We show in the sequel that we can still prove a cumulative loss bound for this simplified version of the update. We note that satisfying all of these constraints simultaneously leads to the online algorithm presented in (Crammer and Singer, 2003a). Their online update is more involved and computationally expensive, and moreover, their analysis only covers the realizable case.

Formally, let r_t denote the lowest ranked relevant label and let s_t denote the highest ranked irrelevant label on round t. That is,

$$r_t \;=\; \operatorname*{argmin}_{r\in Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,r) \qquad \text{and} \qquad s_t \;=\; \operatorname*{argmax}_{s\notin Y_t}\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,s). \tag{36}$$

The single constraint that we choose to satisfy is $\mathbf{w}\cdot\Phi(\mathbf{x}_t,r_t)-\mathbf{w}\cdot\Phi(\mathbf{x}_t,s_t)\geq 1$ and thus $\mathbf{w}_{t+1}$ is set to be the solution of the following simplified constrained optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}}\frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \mathbf{w}\cdot(\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t))\;\geq\;1. \tag{37}$$

The apparent benefit of this simplification lies in the fact that Eq. (37) has a closed form solution. To draw the connection between the multilabel setting and binary classification, we can think of the vector $\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t)$ as a virtual instance of a binary classification problem with a label of $+1$. With this reduction in mind, Eq. (37) becomes equivalent to Eq. (2). Therefore, the closed form solution of Eq. (37) is

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t+\tau_t(\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t)). \tag{38}$$

with,

$$\tau_t \;=\; \frac{\ell_t}{\|\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t)\|^2}.$$

Although we are essentially neglecting all but two labels on each step of the multiclass update, we can still obtain multiclass cumulative loss bounds. The key observation in our analysis it that,

$$\ell_{\text{MC}}\big(\mathbf{w}_t;(\mathbf{x}_t,Y_t)\big) \;=\; \ell\big(\mathbf{w}_t;(\Phi(\mathbf{x}_t,r_t)-\Phi(\mathbf{x}_t,s_t),+1)\big).$$

To remind the reader, ℓ on the right-hand side of the above equation is the binary classification loss defined in Eq. (1). Using this equivalence of definitions, we can convert Thm. 2 into a bound for

the multiclass PA algorithm. To do so we need to cast the assumption that for all t it holds that $\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\| \leq R$. This bound can immediately be converted into a bound on the norm of the feature set since $\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\| \leq \|\Phi(\mathbf{x}_t, r_t)\| + \|\Phi(\mathbf{x}_t, s_t)\|$. Thus, if the norm of the mapping $\Phi(\mathbf{x}_t, r)$ is bounded for all t and r then so is $\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|$. In particular, if we assume that $\|\Phi(\mathbf{x}_t, r)\| \leq R/2$ for all t and r we obtain the following corollary.

Corollary 8 *Let $(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_T, Y_T)$ be a sequence of examples with $\mathbf{x}_t \in \mathbb{R}^n$ and $Y_T \subseteq \{1, \ldots, k\}$. Let Φ be a mapping $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ such that $\|\Phi(\mathbf{x}_t, r)\| \leq R/2$ for all t and r. Assume that there exists a vector $\mathbf{u}$ such that $\ell(\mathbf{u}; (\mathbf{x}_t, Y_t)) = 0$ for all t. Then, the cumulative squared loss attained by the multiclass multilabel PA algorithm is bounded from above by,*

$$\sum_{t=1}^{T} \ell_t^2 \;\leq\; R^2 \|\mathbf{u}\|^2.$$

Similarly, we can obtain multiclass versions of PA-I and PA-II by using the update rule in Eq. (38) but setting τ_t to be either,

$$\tau_t \;=\; \min\left\{ C \,,\; \frac{\ell_t}{\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|^2} \right\} \qquad \text{or} \qquad \tau_t \;=\; \frac{\ell_t}{\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|^2 + \frac{1}{2C}},$$

respectively. The analysis of PA-I and PA-II in Thms. 4-5 also carries over from the binary case to the multilabel case in the same way.

Multi-prototype Classification In the above discussion we assumed that the feature vector $\Phi(\mathbf{x}, y)$ is label-dependent and used a single weight vector $\mathbf{w}$ to form the ranking function. However, in many applications of multiclass classification this setup is somewhat unnatural. Many times, there is a single natural representation for every instance rather than multiple feature representations for each individual class. For example, in optical character recognition problems (OCR) an instance can be a gray-scale image of the character and the goal is to output the content of this image. In this example, it is difficult to find a good set of label-dependent features.

The common construction in such settings is to assume that each instance is a vector in $\mathbb{R}^n$ and to associate a different weight vector (often referred to as prototype) with each of the k labels (Vapnik, 1998; Weston and Watkins, 1999; Crammer and Singer, 2001). That is, the multiclass predictor is now parameterized by $\mathbf{w}_t^1, \ldots, \mathbf{w}_t^k$, where $\mathbf{w}_t^r \in \mathbb{R}^n$. The output of the predictor is defined to be,

$$\Big((\mathbf{w}_t^1 \cdot \mathbf{x}_t), \ldots, (\mathbf{w}_t^k \cdot \mathbf{x}_t) \Big).$$

To distinguish this setting from the previous one we refer to this setting as the multi-prototype multiclass setting and to the previous one as the single-prototype multiclass setting. We now describe a reduction from the multi-prototype setting to the single-prototype one which enables us to use all of the multiclass algorithms discussed above in the multi-prototype setting as well. To obtain the desired reduction, we must define the feature vector representation $\Phi(\mathbf{x}, y)$ induced by the instance label pair $(\mathbf{x}, y)$. We define $\Phi(\mathbf{x}, y)$ to be a $k \cdot n$ dimensional vector which is composed of k blocks of size n. All blocks but the y'th block of $\Phi(\mathbf{x}, y)$ are set to be the zero vector while the y'th block is set to be $\mathbf{x}$. Applying a single prototype multiclass algorithm to this problem produces a weight vector $\mathbf{w}_t \in \mathbb{R}^{kn}$ on every online round. Analogous to the construction of $\Phi(\mathbf{x}, y)$, the vector $\mathbf{w}_t$ is composed

INPUT: cost function $\rho(y,y')$
INITIALIZE: $\mathbf{w}_1 = (0,\ldots,0)$
For $t = 1,2,\ldots$

- receive instance: $\mathbf{x}_t \in \mathbb{R}^n$
- predict: $\hat{y}_t = \arg\max_{y\in\mathcal{Y}} (\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, y))$
- receive correct label: $y_t \in \mathcal{Y}$
- define: $\tilde{y}_t = \arg\max_{r\in\mathcal{Y}} \left(\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, y_t) + \sqrt{\rho(y_t, r)}\right)$
- define:
$$q_t = \begin{cases} \hat{y}_t & \text{(PB)} \\ \tilde{y}_t & \text{(ML)} \end{cases}$$
- suffer loss: $\ell_t = \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, q_t) - \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, y_t) + \sqrt{\rho(y_t, q_t)}$
- set: $\tau_t = \frac{\ell_t}{\|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, q_t)\|^2}$
- update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \left(\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, q_t)\right)$

Figure 2: The *prediction-based* (PB) and *max-loss* (ML) passive-aggressive updates for cost-sensitive multiclass problems.

of k blocks of size n and denote block r by $\mathbf{w}_t^r$. By construction, we get that $\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) = \mathbf{w}_t^r \cdot \mathbf{x}_t$. Equipped with this construction we can use verbatim any single-prototype algorithm as a proxy for the multi-prototype variant. Namely, on round t we find the pair of indices r_t, s_t which corresponds to the largest violation of the margin constraints,

$$\begin{aligned} r_t &= \operatorname*{argmin}_{r\in Y_t} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, r) = \operatorname*{argmin}_{r\in Y_t} \mathbf{w}_t^r \cdot \mathbf{x}_t \ , \\ s_t &= \operatorname*{argmax}_{s\notin Y_t} \mathbf{w}_t \cdot \Phi(\mathbf{x}_t, s) = \operatorname*{argmax}_{s\notin Y_t} \mathbf{w}_t^s \cdot \mathbf{x}_t . \end{aligned} \tag{39}$$

Unraveling the single-prototype notion of margin and casting it as a multi-prototype one we get that the loss in the multi-prototype case amounts to,

$$\ell\left(\mathbf{w}_t^1, \ldots, \mathbf{w}_t^k; (\mathbf{x}_t, Y_t)\right) = \begin{cases} 0 & \mathbf{w}_t^{r_t} \cdot \mathbf{x}_t - \mathbf{w}_t^{s_t} \cdot \mathbf{x}_t \geq 1 \\ 1 - \mathbf{w}_t^{r_t} \cdot \mathbf{x}_t + \mathbf{w}_t^{s_t} \cdot \mathbf{x}_t & \text{otherwise} \end{cases} . \tag{40}$$

Furthermore, applying the same reduction to the update scheme we get that the resulting multi-prototype update is,

$$\mathbf{w}_{t+1}^{r_t} = \mathbf{w}_{t+1}^{r_t} + \tau_t \mathbf{x}_t \quad \text{and} \quad \mathbf{w}_{t+1}^{s_t} = \mathbf{w}_{t+1}^{s_t} - \tau_t \mathbf{x}_t . \tag{41}$$

For the PA algorithm, the value of τ_t is the ratio of the loss, as given by Eq. (40), and the squared norm of $\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)$. By construction, this vector has $k-2$ blocks whose elements are zeros and two blocks that are equal to $\mathbf{x}_t$ and $-\mathbf{x}_t$. Since the two non-zero blocks are non-overlapping we get that,

$$\|\Phi(\mathbf{x}_t, r_t) - \Phi(\mathbf{x}_t, s_t)\|^2 = \|\mathbf{x}_t\|^2 + \|-\mathbf{x}_t\|^2 = 2\|\mathbf{x}_t\|^2 .$$

Finally, due to our reduction we also get multi-prototype versions of Thm. 4 and Thm. 5.

8. Cost-Sensitive Multiclass Classification

Cost-sensitive multiclass classification is a variant of the multiclass single-label classification setting discussed in the previous section. Namely, each instance $\mathbf{x}_t$ is associated with a single correct label $y_t \in \mathcal{Y}$ and the prediction extended by the online algorithm is simply,

$$\hat{y}_t \;=\; \operatorname*{argmax}_{y \in \mathcal{Y}} \left(\mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y)\right). \tag{42}$$

A prediction mistake occurs if $y_t \neq \hat{y}_t$, however in the cost-sensitive setting different mistakes incur different levels of cost. Specifically, for every pair of labels (y, y') there is a cost $\rho(y, y')$ associated with predicting y' when the correct label is y. The cost function ρ is defined by the user and takes non-negative values. We assume that $\rho(y, y) = 0$ for all $y \in \mathcal{Y}$ and that $\rho(y, y') \geq 0$ whenever $y \neq y'$. The goal of the algorithm is to minimize the *cumulative cost* suffered on a sequence of examples, namely to minimize $\sum \rho(y_t, \hat{y}_t)$.

The multiclass PA algorithms discussed above can be adapted to this task by incorporating the cost function into the online update. Recall that we began the derivation of the multiclass PA update by defining a set of margin constraints in Eq. (35), and on every round we focused our attention on satisfying only one of these constraints. We repeat this idea here while incorporating the cost function into the margin constraints. Specifically, on every online round we would like for the following constraints to hold,

$$\forall r \in \{\mathcal{Y} \setminus y_t\} \quad \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y_t) - \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, r) \;\geq\; \sqrt{\rho(y_t, r)}. \tag{43}$$

The reason for using the square root function in the equation above will be justified shortly. As mentioned above, the online update focuses on a single constraint out of the $|\mathcal{Y}| - 1$ constraints in Eq. (43). We will describe and analyze two different ways to choose this single constraint, which lead to two different online updates for cost-sensitive classification. The two update techniques are called the *prediction-based* update and the *max-loss* update. Pseudo-code for these two updates is presented in Fig. 2. They share an almost identical analysis and may seem very similar at first, however each update possesses unique qualities. We discuss the significance of each update at the end of this section.

The prediction-based update focuses on the single constraint in Eq. (43) which corresponds to the predicted label $\hat{y}_t$. Concretely, this update sets $\mathbf{w}_{t+1}$ to be the solution to the following optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|^2 \quad \text{s.t.} \quad \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y_t) - \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, \hat{y}_t) \;\geq\; \sqrt{\rho(y_t, \hat{y}_t)}, \tag{44}$$

where $\hat{y}_t$ is defined in Eq. (42). This update closely resembles the multiclass update given in Eq. (37). Define the cost sensitive loss for the prediction-based update to be,

$$\ell_{\text{PB}}\big(\mathbf{w}; (\mathbf{x}, y)\big) \;=\; \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, \hat{y}) - \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}, y) + \sqrt{\rho(y, \hat{y})}. \tag{45}$$

Note that this loss equals zero if and only if a correct prediction was made, namely if $\hat{y}_t = y_t$. On the other hand, if a prediction mistake occurred it means that $\mathbf{w}_t$ ranked $\hat{y}_t$ higher than y_t, thus,

$$\sqrt{\rho(y_t, \hat{y}_t)} \;\leq\; \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, \hat{y}_t) - \mathbf{w}_t \cdot \mathbf{\Phi}(\mathbf{x}_t, y_t) + \sqrt{\rho(y_t, \hat{y}_t)} \;=\; \ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t)). \tag{46}$$

As in previous sections, we will prove an upper bound on the cumulative squared loss attained by our algorithm, $\sum_t \ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))^2$. The cumulative squared loss in turn bounds $\sum_t \rho(y_t,\hat{y}_t)$ which is the quantity we are trying to minimize. This explains the rationale behind our choice of the margin constraints in Eq. (43). The update in Eq. (44) has the closed form solution,

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t + \tau_t\left(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\right), \tag{47}$$

where,

$$\tau_t \;=\; \frac{\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))}{\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2}. \tag{48}$$

As before, we obtain cost sensitive versions of PA-I and PA-II by setting,

$$\begin{aligned}
\tau_t &= \min\left\{ C\,,\; \frac{\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))}{\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2} \right\} \qquad \text{(PA-I)} \\
\tau_t &= \frac{\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))}{\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2 + \frac{1}{2C}} \qquad \text{(PA-II)},
\end{aligned} \tag{49}$$

where in both cases $C > 0$ is a user-defined parameter.

The second cost sensitive update, the max-loss update, also focuses on satisfying a single constraint from Eq. (43). Let $\tilde{y}_t$ be the label in $\mathcal{Y}$ defined by,

$$\tilde{y}_t \;=\; \operatorname*{argmax}_{r\in\mathcal{Y}} \left(\mathbf{w}_t\cdot\Phi(\mathbf{x}_t,r) - \mathbf{w}_t\cdot\Phi(\mathbf{x}_t,y_t) + \sqrt{\rho(y_t,r)}\right). \tag{50}$$

$\tilde{y}_t$ is the loss-maximizing label. That is, we would suffer the greatest loss on round t if we were to predict $\tilde{y}_t$. The max-loss update focuses on the single constraint in Eq. (43) which corresponds to $\tilde{y}_t$. Note that the online algorithm continues to predict the label $\hat{y}_t$ as before and that $\tilde{y}_t$ only influences the online update. Concretely, the max-loss update sets $\mathbf{w}_{t+1}$ to be the solution to the following optimization problem,

$$\mathbf{w}_{t+1} \;=\; \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \mathbf{w}_t\cdot\Phi(\mathbf{x}_t,y_t) - \mathbf{w}_t\cdot\Phi(\mathbf{x}_t,\tilde{y}_t) \;\geq\; \sqrt{\rho(y_t,\tilde{y}_t)}, \tag{51}$$

The update in Eq. (51) has the same closed form solution given in Eq. (47) and Eq. (48) with $\hat{y}_t$ replaced by $\tilde{y}_t$. Define the loss for the max-loss update to be,

$$\ell_{\text{ML}}\big(\mathbf{w};(\mathbf{x},y)\big) \;=\; \mathbf{w}\cdot\Phi(\mathbf{x},\tilde{y}) - \mathbf{w}\cdot\Phi(\mathbf{x},y) + \sqrt{\rho(y,\tilde{y})}, \tag{52}$$

where $\tilde{y}$ is defined in Eq. (50). Note that since $\tilde{y}$ attains the maximal loss of all other labels, it follows that,

$$\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) \;\leq\; \ell_{\text{ML}}(\mathbf{w}_t;(\mathbf{x}_t,y_t)).$$

From the above inequality and Eq. (46) we conclude that ℓ_{ML} is also an upper bound on $\sqrt{\rho(y_t,\hat{y}_t)}$. A note-worthy difference between ℓ_{PB} and ℓ_{ML} is that $\ell_{\text{ML}}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) = 0$ if and only if Eq. (43) holds for all $r \in \{\mathcal{Y}\setminus y_t\}$, whereas this is not the case for ℓ_{PB}.

The prediction-based and max-loss updates were previously discussed in Dekel et al. (2004a), in the context of hierarchical classification. In that paper, a predefined hierarchy over the label set was used to induce the cost function ρ. The basic online algorithm presented there used the prediction-based update, whereas the max-loss update was mentioned in the context of a batch learning setting.

Dekel et al. (2004a) evaluated both techniques empirically and found them to be highly effective on speech recognition and text classification tasks.

Turning to the analysis of our cost sensitive algorithms, we follow the same strategy used in the analysis of the regression and uniclass algorithms. Namely, we begin by proving a cost sensitive version of Lemma 1 for both the prediction-based and the max-loss updates.

Lemma 9 *Let $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_T,y_T)$ be an arbitrary sequence of examples, where $\mathbf{x}_t \in \mathbb{R}$ and $y_t \in \mathcal{Y}$ for all t. Let $\mathbf{u}$ be an arbitrary vector in $\mathbb{R}^n$. If τ_t is defined as in Eq. (48) or Eq. (49) then,*

$$\sum_{t=1}^{T} \tau_t \left(2\ell_{PB}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) - \tau_t \|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2 - 2\ell_{ML}(\mathbf{u};(\mathbf{x}_t,y_t))\right) \;\leq\; \|\mathbf{u}\|^2.$$

If τ_t is defined as in Eq. (48) or Eq. (49) with $\hat{y}_t$ replaced by $\tilde{y}_t$ then,

$$\sum_{t=1}^{T} \tau_t \left(2\ell_{ML}(\mathbf{w}_t;(\mathbf{x}_t,y_t)) - \tau_t \|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\tilde{y}_t)\|^2 - 2\ell_{ML}(\mathbf{u};(\mathbf{x}_t,y_t))\right) \;\leq\; \|\mathbf{u}\|^2.$$

Proof We prove the first statement of the lemma, which involves the prediction-based update rule. The proof of the second statement is identical, except that $\hat{y}_t$ is replaced by $\tilde{y}_t$ and $\ell_{\text{PB}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))$ is replaced by $\ell_{\text{ML}}(\mathbf{w}_t;(\mathbf{x}_t,y_t))$.

As in the proof of Lemma 1, we use the definition $\Delta_t = \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|^2$ and the fact that,

$$\sum_{t=1}^{T} \Delta_t \;\leq\; \|\mathbf{u}\|^2. \tag{53}$$

We focus our attention on bounding Δ_t from below. Using the recursive definition of $\mathbf{w}_{t+1}$, we rewrite Δ_t as,

$$\begin{aligned}\Delta_t &= \|\mathbf{w}_t - \mathbf{u}\|^2 - \|\mathbf{w}_t - \mathbf{u} + \tau_t(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t))\|^2 \\ &= -2\tau_t(\mathbf{w}_t - \mathbf{u})\cdot(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)) - \tau_t^2\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2. \end{aligned} \tag{54}$$

By definition, $\ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))$ equals,

$$\max_{r\in\mathcal{Y}} \left(\mathbf{u}\cdot(\Phi(\mathbf{x}_t,r) - \Phi(\mathbf{x}_t,y_t)) + \sqrt{\rho(y_t,r)}\right).$$

Since $\ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))$ is the maximum over $\mathcal{Y}$, it is clearly greater than $\mathbf{u}\cdot(\Phi(\mathbf{x}_t,\hat{y}_t) - \Phi(\mathbf{x}_t,y_t)) + \sqrt{\rho(y_t,\hat{y}_t)}$. This can be written as,

$$\mathbf{u}\cdot(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)) \;\geq\; \sqrt{\rho(y_t,\hat{y}_t)} - \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t)).$$

Plugging the above back into Eq. (54) we get,

$$\begin{aligned}\Delta_t \;\geq\; &-2\tau_t\mathbf{w}_t\cdot(\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)) + 2\tau_t\left(\sqrt{\rho(y_t,\hat{y}_t)} - \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))\right) \\ &- \tau_t^2\|\Phi(\mathbf{x}_t,y_t) - \Phi(\mathbf{x}_t,\hat{y}_t)\|^2. \end{aligned} \tag{55}$$

Rearranging terms in the definition of ℓ_{PB}, we get that $\mathbf{w}_t \cdot (\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)) = \sqrt{\rho(y_t, \hat{y}_t)} - \ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t))$. This enables us to rewrite Eq. (55) as,

$$\begin{aligned}
\Delta_t &\geq -2\tau_t \left(\sqrt{\rho(y_t, \hat{y}_t)} - \ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t)) \right) + \\
&\qquad 2\tau_t \left(\sqrt{\rho(y_t, \hat{y}_t)} - \ell_{\text{ML}}(\mathbf{u}; (\mathbf{x}_t, y_t)) \right) - \tau_t^2 \|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)\|^2 \\
&= \tau_t \left(2\ell_{\text{PB}}(\mathbf{w}_t; (\mathbf{x}_t, y_t)) - \tau_t \|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)\|^2 - 2\ell_{\text{ML}}(\mathbf{u}; (\mathbf{x}_t, y_t)) \right).
\end{aligned}$$

Summing Δ_t over all t and comparing this lower bound with the upper bound provided in Eq. (53) gives the desired bound. ■

This lemma can now be used to obtain cost sensitive versions of Thms. 2,3 and 5 for both prediction-based and max-loss updates. The proof of these theorems remains essentially the same as before, however one cosmetic change is required: $\|\mathbf{x}_t\|^2$ is replaced by either $\|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \hat{y}_t)\|^2$ or $\|\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, \tilde{y}_t)\|^2$ in each of the theorems and throughout their proofs. This provides cumulative cost bounds for the PA and PA-II cost-sensitive algorithms.

Analyzing the cost-sensitive version of PA-I requires a slightly more delicate adaptation of Thm. 4. For brevity, we prove the following theorem for the max-loss variant of the algorithm and note that the proof for the prediction-based variant is essentially identical.

We make two simplifying assumptions: first assume that $\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\|$ is upper bounded by 1. Second, assume that C, the aggressiveness parameter given to the PA-I algorithm, is an upper bound on the square root of the cost function ρ.

Theorem 10 *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples where $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \mathcal{Y}$ and $\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\| \leq 1$ for all t. Let ρ be a cost function from $\mathcal{Y} \times \mathcal{Y}$ to $\mathbb{R}_+$ and let C, the aggressiveness parameter provided to the PA-I algorithm, be such that $\sqrt{\rho(y_t, \hat{y}_t)} \leq C$ for all t. Then for any vector $\mathbf{u} \in \mathbb{R}^n$, the cumulative cost obtained by the max-loss cost sensitive version of PA-I on the sequence is bounded from above by,*

$$\sum_{t=1}^{T} \rho(y_t, \hat{y}_t) \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_{ML}(\mathbf{u}; (\mathbf{x}_t, y_t)).$$

Proof We abbreviate $\rho_t = \rho(y_t, \hat{y}_t)$ and $\ell_t = \ell_{\text{ML}}(\mathbf{w}_t; (\mathbf{x}_t, y_t))$ throughout this proof. $\ell_t \geq \sqrt{\rho_t}$ on every round t, as discussed in this section. τ_t is defined as,

$$\tau_t \;=\; \min \left\{ \frac{\ell_t}{\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\|^2} \,,\, C \right\},$$

and due to our assumption on $\|\phi(\mathbf{x}_t, y_t) - \phi(\mathbf{x}_t, \tilde{y}_t)\|^2$ we get that $\tau_t \geq \min\{\ell_t, C\}$. Combining these two facts gives,

$$\min\{\rho_t, C\sqrt{\rho_t}\} \;\leq\; \tau_t \ell_t.$$

Using our assumption on C, we know that $C\sqrt{\rho_t}$ is at least ρ_t and therefore $\rho_t \leq \tau_t \ell_t$. Summing over all t we get the bound,

$$\sum_{t=1}^{T} \rho_t \;\leq\; \sum_{t=1}^{T} \tau_t \ell_t. \tag{56}$$

Again using the definition of τ_t, we know that $\tau_t \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t)) \leq C\ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t))$ and that $\tau_t \|\phi(\mathbf{x}_t,y_t) - \phi(\mathbf{x}_t,\tilde{y}_t)\|^2 \leq \ell_t$. Plugging these two inequalities into the second statement of Lemma 9 gives,

$$\sum_{t=1}^{T} \tau_t \ell_t \;\leq\; \|\mathbf{u}\|^2 + 2C \sum_{t=1}^{T} \ell_{\text{ML}}(\mathbf{u};(\mathbf{x}_t,y_t)). \tag{57}$$

Combining Eq. (57) with Eq. (56) proves the theorem. ■

This concludes our analysis of the cost-sensitive PA algorithms. We wrap up this section with a discussion on some significant differences between the prediction-based and the max-loss variants of our cost-sensitive algorithms. Both variants utilize the same prediction function to output the predicted label $\hat{y}_t$ however each variant follows a different update strategy and is evaluated with respect to a different loss function. The loss function used to evaluate the prediction-based variant is a function of y_t and $\hat{y}_t$, whereas the loss function used to evaluate the max-loss update essentially ignores $\hat{y}_t$. In this respect, the prediction-based loss is more natural.

On the other hand, the analysis of the prediction-based variant lacks the aesthetics of the max-loss analysis. The analysis of the max-loss algorithm uses ℓ_{ML} to evaluate both the performance of the algorithm and the performance of $\mathbf{u}$, while the analysis of the prediction-based algorithm uses ℓ_{PB} to evaluate the algorithm and ℓ_{ML} to evaluate $\mathbf{u}$. The prediction-based relative bound is to some extent like comparing apples and oranges, since the algorithm and $\mathbf{u}$ are not evaluated using the same loss function. In summary, both algorithms suffer from some theoretical disadvantage and neither of them is theoretically superior to the other.

Finally, we turn our attention to an important algorithmic difference between the two update strategies. The prediction-based update has a great advantage over the max-loss update in that the cost function ρ does not play a role in determining the single constraint which the update focuses on. In some cases, this can significantly speed-up the running time required by the online update. For example, in the following section we exploit this property when devising algorithms for the complex problem of sequence prediction. When reading the following section, note that the max-loss update could not have been used for sequence prediction in place of the prediction-based update. This is perhaps the most significant difference between the two cost sensitive updates.

9. Learning with Structured Output

A trendy and useful application of large margin methods is learning with structured output. In this setting, the set of possible labels are endowed with a predefined structure. Typically, the set of labels is very large and the structure plays a key role in constructing efficient learning and inference procedures. Notable examples for structured label sets are graphs (in particular trees) and sequences (Collins, 2000; Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004). We now overview how the cost-sensitive learning algorithms described in the previous section can be adapted to structured output settings. For concreteness, we focus on an adaptation for sequence prediction. Our derivation however can be easily mapped to other settings of learning with structured output. In sequence prediction problems we are provided with a predefined alphabet $\mathcal{Y} = \{1,\ldots,k\}$. Each input instance is associated with a label which is a sequence over $\mathcal{Y}$. For simplicity we assume that the output sequence is of a fixed length m. Thus, on round t, the learning algorithm receives an instance $\mathbf{x}_t$ and then predicts an output sequence $\hat{\mathbf{y}}_t \in \mathcal{Y}^m$. Upon predicting, the algorithm receives

the correct sequence $\mathbf{y}_t$ that is associated with $\mathbf{x}_t$. As in the cost-sensitive case, the learning algorithm is also provided with a cost function $\rho : \mathcal{Y}^m \times \mathcal{Y}^m \to \mathbb{R}_+$. The value of $\rho(\mathbf{y}, \mathbf{y}')$ represents the cost associated with predicting $\mathbf{y}'$ instead of $\mathbf{y}$. As before we assume that $\rho(\mathbf{y}, \mathbf{y}')$ equals zero if $\mathbf{y} = \mathbf{y}'$. Apart from this requirement, ρ may be any computable function. Most sequence prediction algorithms further assume that ρ is decomposable. Specifically, a common construction (Taskar et al., 2003; Tsochantaridis et al., 2004) is achieved by defining, $\rho(\mathbf{y}, \mathbf{y}') = \sum_{i=1}^{m} \tilde{\rho}(y_i, y'_i)$ where $\tilde{\rho}$ is any non-negative (local) cost over $\mathcal{Y} \times \mathcal{Y}$. In contrast, we revert to a general cost function over pairs of sequences.

As in the multiclass settings discussed above, we assume that there exists a set of features $\phi_1, \ldots, \phi_d$ each of which takes as its input an instance $\mathbf{x}$ and a sequence $\mathbf{y}$ and outputs a real number. We again denote by $\Phi(\mathbf{x}, \mathbf{y})$ the vector of features evaluated on $\mathbf{x}$ and $\mathbf{y}$. Equipped with Φ and ρ, we are left with the task of finding,

$$\hat{\mathbf{y}}_t \;=\; \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}^m} \left(\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, \mathbf{y}) \right), \tag{58}$$

on every online round. With $\hat{\mathbf{y}}_t$ on hand, the PA update for string prediction is identical to the prediction-based update described in the previous section. However, obtaining $\hat{\mathbf{y}}_t$ in the general case may require as many as k^m evaluations of $\mathbf{w}_t \cdot \Phi(\mathbf{x}_t, \mathbf{y})$. This problem becomes intractable as m becomes large. We must therefore impose some restrictions on the feature representation Φ which will enable us to find $\hat{\mathbf{y}}_t$ efficiently. A possible restriction on the feature representation is to assume that each feature ϕ_j takes the form,

$$\phi_j(\mathbf{x}_t, \mathbf{y}) \;=\; \psi_j^0(y_1, \mathbf{x}_t) + \sum_{i=2}^{m} \psi_j(y_i, y_{i-1}, \mathbf{x}_t), \tag{59}$$

where ψ_j^0 and ψ_j are any computable functions. This construction is analogous to imposing a first order Markovian structure on the output sequence. This form paves the way for an efficient inference, i.e. solving Eq. (58), using a dynamic programming procedure. Similar yet richer structures such as dynamic Bayes nets can be imposed so long as the solution to Eq. (58) can be computed efficiently. We note in passing that similar representation of Φ using efficiently computable feature sets were proposed in (Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004).

The analysis of the cost-sensitive PA updates carries over verbatim to the sequence prediction setting. Our algorithm for learning with structured outputs was successfully applied to the task of music to score alignment in (Shalev-Shwartz et al., 2004a).

10. Experiments

In this section we present experimental results that demonstrate different aspects of our PA algorithms and their accompanying analysis. In Sec. 10.1 we start with two experiments with synthetic data which examine the robustness of our algorithms to noise. In Sec. 10.2 we investigate the effect of the aggressiveness parameter C on the performance of the PA-I and PA-II algorithms. Finally, in Sec. 10.3, we compare our multiclass versions of the PA algorithms to other online algorithms for multiclass problems (Crammer and Singer, 2003a) on natural data sets.

The synthetic data set used in our experiments was generated as follows. First a label was chosen uniformly at random from $\{-1, +1\}$. For positive labeled examples, instances were chosen by randomly sampling a two-dimensional Gaussian with mean $(1, 1)$ and a diagonal covariance matrix

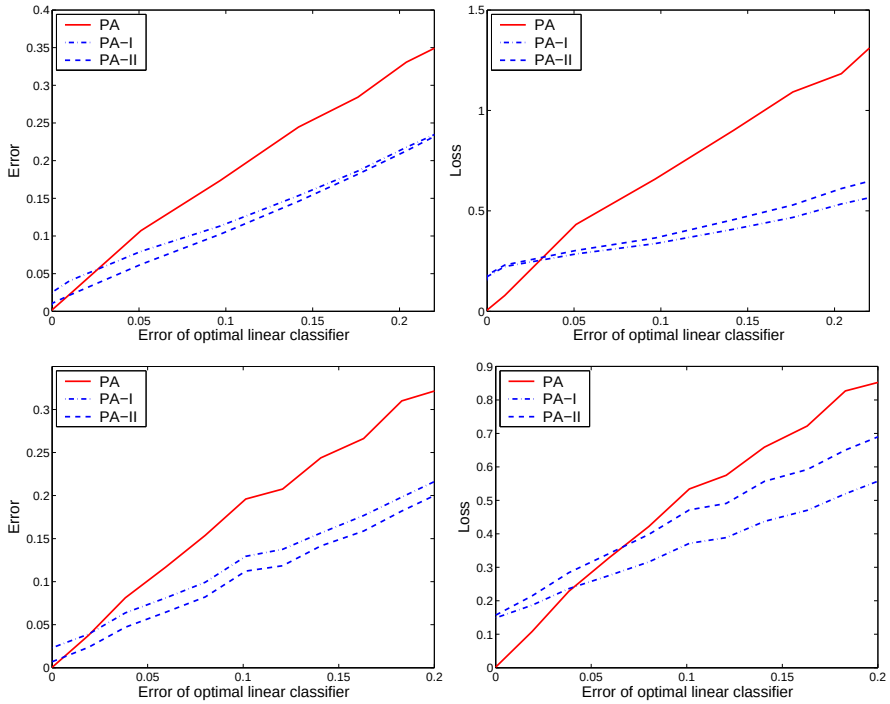


Figure 3: The average error (left) and the average loss (right) of PA, PA-I and PA-II as a function of the error of the optimal fixed linear classifier, in the presence of instance noise (top) and label noise (bottom).

with $(0.2, 2)$ on its diagonal. Similarly, for negative labeled examples, instances were sampled from a Gaussian with a mean of $(-1, -1)$ and the same covariance matrix as for positive labeled examples. To validate our results, we repeated each experiment 10 times where in each repetition we generated 4,000 random examples. The results reported are averaged over the 10 repetitions.

10.1 Robustness to Noise

Our first experiments examine the robustness of our algorithms to both instance noise and label noise. To examine instance noise, we contaminated each instance with a random vector sampled from a zero-mean Gaussian with a covariance matrix σI, where σ varied from 0 to 2. We set the parameter C of PA-I and PA-II to be 0.001. We then ran PA, PA-I and PA-II on the resulting sequence of examples. To evaluate our results, we used a brute-force numerical method to find the optimal fixed linear classifier, that is, the linear classifier that makes the fewest classification mistakes on the entire sequence of examples. We define the *average error* of an online learning algorithm on a given input sequence to be the number of prediction mistakes the algorithm makes on that sequence normalized by the length of the sequence. Similarly, we define the average loss of an online learning algorithm on a given sequence.

In the plots at the top of Fig. 3 we depict the average error and average loss of the three PA variants as a function of the average error of the optimal linear classifier. The plots underscore

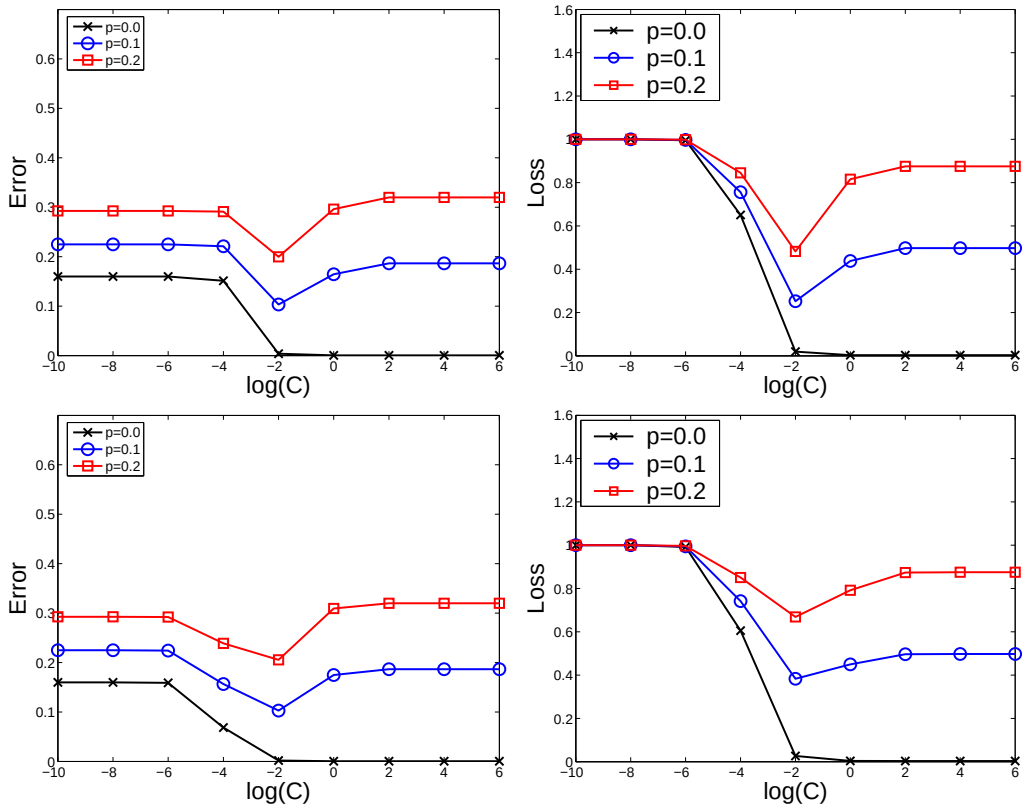


Figure 4: The average error (left) and the average loss (right) of PA-I (top) and PA-II (bottom) as a function of $\log(C)$ with different levels of label noise probability p.

several interesting phenomena. First note that for low levels of noise, all three PA variants make a similar number of errors. Our bounds from Sec. 4 suggest that as the noise level increases, PA-I and PA-II should outperform the basic PA algorithm. It is clear from the graphs that our expectations are met and that PA-I and PA-II outperform the basic PA algorithm when the noise level is high. Finally, in this experiment PA-I and PA-II performed equally well for all levels of noise.

In our second experiment we left the instances intact and instead flipped each label with a probability p, where p was set to different values in $[0, 0.3]$. As in the previous experiment, we set $C = 0.001$ for both PA-I and PA-II . The results are depicted at the bottom of Fig. 3. It is apparent from the graphs that the behavior observed in the previous experiment is repeated here as well.

10.2 The Effect of C

In our second set of experiments, we examine the effect of the aggressiveness parameter C on the performance of PA-I and PA-II . Again we flipped the label of each instance in our synthetic data set with probability p, this time with p set to 0, 0.1 and 0.2. We then ran PA-I and PA-II on the resulting sequence of examples with different values of the parameter C. The average error and average loss of the algorithms as a function of the parameter C are depicted in Fig. 4

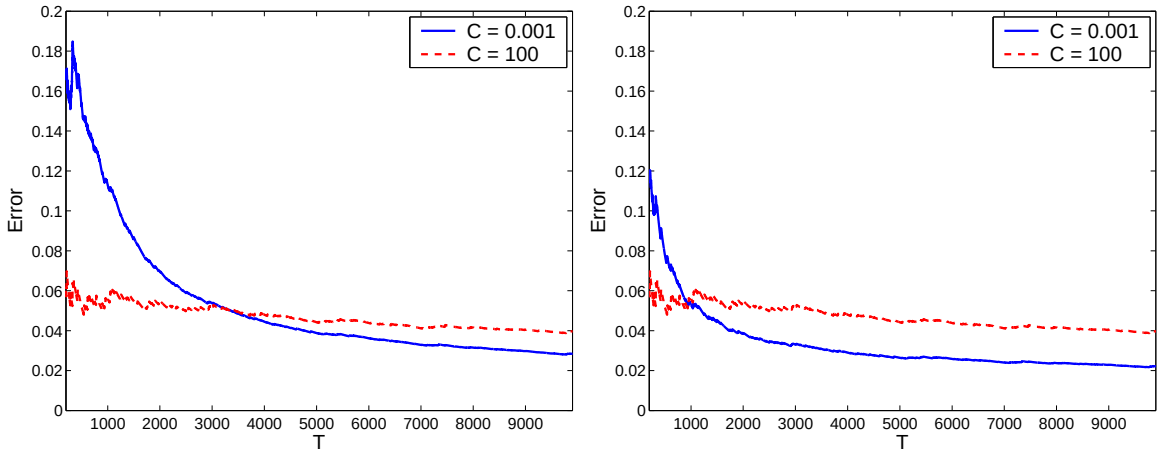


Figure 5: The average error of PA-I (left) and PA-II (right) as a function of the number of online rounds, T, for different values of C.

As can be seen from the graphs, the value of the parameter C significantly effects the results of the algorithms. The graphs can be explained using our loss bounds in Thm. 4 and Thm. 5. For concreteness, let us focus on the loss bound of the PA-II algorithm, given in Thm. 5. The bound on the cumulative loss of the algorithm is comprised of two terms, the first depends on the squared norm of the competitor, $(\|\mathbf{u}\|^2)$, while the second depends on the cumulative (squared) loss of the competitor $(\sum_t (\ell_t^\star)^2)$. The parameter C divides the first term and multiplies the second term. Therefore, when C is small the bound is dominated by the first term $(\|\mathbf{u}\|^2)$ and when C is large the bound is dominated by the second term $(\sum_t (\ell_t^\star)^2)$. Since the label noise applied to the data effects only the second term, we expect that for very small values of C the loss of PA-I and PA-II will be high, regardless of the noise level. On the other hand, as we increase the value of C, the difference between different noise levels becomes apparent. As a general rule-of-thumb, C should be small when the data is noisy.

So far, the length of the sequence of examples presented to the online algorithms was fixed. In the following, we discuss the effect of C on the performance of the algorithms as a function of sequence length (T). We generated a synthetic data set consisting of 10^4 examples with label noise probability $p = 0.02$. We ran the PA-I and PA-II algorithms on the data set, once with $C = 100$ and once with $C = 0.001$. At the end of each online round we calculated the average error attained so far. The results are given in Fig. 5. For both PA-I and PA-II , setting C to be a small number leads to a slow progress rate, since each online update changes the online hypothesis by a small amount. On the other hand, when C is large, the error rate decreases much faster, but at the price of inferior performance later on.

10.3 Multiclass Experiments

Our last experiment demonstrates the efficiency of the PA algorithms on multiclass problems. This experiment was performed with standard multiclass data sets: the USPS and MNIST data sets of handwritten digits. We compared the multiclass versions of PA, PA-I and PA-II to the online

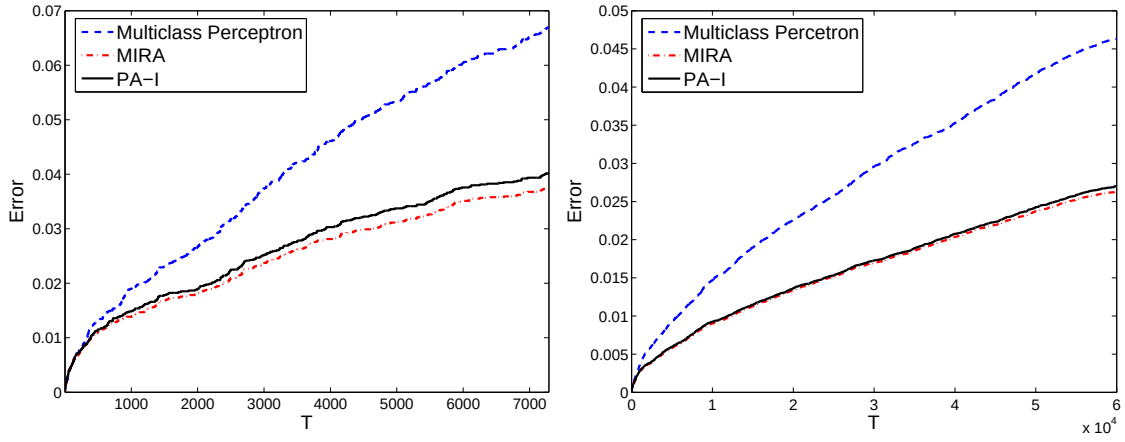


Figure 6: The number of prediction mistakes made by different multiclass online algorithms as a function of the online round index, on the USPS (left) and MNIST (right) data sets.

multiclass algorithms described in (Crammer and Singer, 2003a). Specifically, Crammer and Singer (2003a) present three multiclass versions of the Perceptron algorithm and a new margin based online multiclass algorithm named MIRA. As a preprocessing step, we shifted and scaled the instances of each data set so that its mean equals zero and its average squared Euclidean norm is 1. We used Mercer kernels in all of the algorithms, namely, we replaced the standard dot product with a polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (a + \mathbf{x}_i \cdot \mathbf{x}_j)^d$, where $a = 0$ and $d = 3$ for the USPS data set and $a = 0.5$ and $d = 5$ for the MNIST data set. These kernel parameters were set rather arbitrarily, based on previous experience with these data sets using different algorithms. We set the parameter C of PA-I and PA-II to 100 (we note that similar results hold for any $C > 100$). The parameter β of MIRA was set to 0.01, following (Crammer and Singer, 2003a).

The plots in Fig. 6 depict the number of online prediction mistakes made on the two data sets by three different algorithms: PA-I , the uniform-update version of multiclass Perceptron and MIRA. The performance of PA and PA-II is not presented in this figure, since it is virtually indistinguishable from that of PA-I . For the same reason, only the uniform-update version of the multiclass Perceptron is presented in the figure. It is apparent that both PA-I and MIRA outperform the Perceptron. In addition, the performance of PA-I is comparable to that of MIRA with a slight advantage to the latter. However, while each online update of MIRA requires solving a complex optimization problem, each update of PA has a simple closed-form expression and is thus much faster and easier to implement.

11. Discussion

We described an online algorithmic framework for solving numerous prediction problems ranging from classification to sequence prediction. We derived several loss bounds for our algorithms (Thms. 2-5). The proofs of all of the bounds are based on a single lemma (Lemma 1). There are several possible extensions of the work presented in this paper. We already conducted further research on applications of the PA algorithmic framework for learning margin-based suffix

trees (Dekel et al., 2004b), pseudo-metrics (Shalev-Shwartz et al., 2004b), hierarchical classification (Dekel et al., 2004a), and segmentation of sequences (Shalev-Shwartz et al., 2004a). While the focus of this paper is on online settings, online algorithms can also serve as building blocks in the construction of well performing batch algorithms. Online to batch conversions of the proposed algorithms are yet another important future research direction. The update taken by our algorithms is aggressive in the sense that even a small loss forces an update of the hypothesis. When using kernels, this property often results in the use of many examples for representing the learned predictor. Thus, the memory requirements imposed when using kernels can be quite demanding. We are currently pursuing extensions of the PA framework that operate in the realm of bounded memory constraints.

Acknowledgments

This research was funded by the Israeli Science Foundation under grant number 522/04. Most of this work was carried out at the Hebrew University of Jerusalem.

Appendix A. Derivation of the PA-I and PA-II Updates

As in Sec. 3, whenever $\ell_t = 0$ no update occurs and τ_t equals zero. If $\ell_t > 0$ we derive these updates by defining the Lagrangian of the respective optimization problem and satisfying the KKT conditions. The Lagrangian of the PA-I optimization problem is,

$$\begin{aligned} \mathcal{L}(\mathbf{w},\xi,\tau,\lambda) &= \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + C\xi + \tau(1-\xi-y_t(\mathbf{w}\cdot\mathbf{x}_t)) - \lambda\xi \\ &= \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 + \xi(C-\tau-\lambda) + \tau(1-y_t(\mathbf{w}\cdot\mathbf{x}_t)), \end{aligned} \tag{60}$$

where $\tau \geq 0$ and $\lambda \geq 0$ are Lagrange multipliers. We now find the minimum of the Lagrangian with respect to the (unconstrained) primal variables $\mathbf{w}$ and ξ. As in the previously discussed PA update, differentiating this Lagrangian with respect to the elements of $\mathbf{w}$ and setting these partial derivatives to zero gives Eq. (5) and we can write $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$. Next, note that the minimum of the term $\xi(C-\tau-\lambda)$ with respect to ξ is zero whenever $C-\tau-\lambda = 0$. If however $C-\tau-\lambda \neq 0$ then $\xi(C-\tau-\lambda)$ can be made to approach $-\infty$. Since we need to maximize the dual we can rule out the latter case and pose the following constraint on the dual variables,

$$C-\tau-\lambda = 0. \tag{61}$$

The KKT conditions confine λ to be non-negative so we conclude that $\tau \leq C$. We now discuss two possible cases: if $\ell_t/\|\mathbf{x}_t\|^2 \leq C$ then we can plugging Eq. (61) back into Eq. (60) and we return to the Lagrangian of the original PA algorithm (see Eq. (4)). From this point and on, we can repeat the same derivation as in the original PA update and get $\tau_t = \ell_t/\|\mathbf{x}_t\|^2$. The other case is when $\ell_t/\|\mathbf{x}_t\|^2 > C$. This condition can be rewritten as

$$C\|\mathbf{x}_t\|^2 < 1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t). \tag{62}$$

We also know that the constraint in Eq. (6) must hold at the optimum, so $1-y_t(\mathbf{w}\cdot\mathbf{x}_t) \leq \xi$. Using the explicit form of $\mathbf{w}$ given in Eq. (5), we can rewrite this constraint as $1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)-\tau\|\mathbf{x}_t\|^2 \leq \xi$.

Combining this inequality with the inequality in Eq. (62) gives,

$$C\|\mathbf{x}_t\|^2 - \tau\|\mathbf{x}_t\|^2 \;<\; \xi.$$

We now use our earlier conclusion that $\tau \leq C$ to obtain $0 < \xi$. Turning to the KKT complementarity condition, we know that $\xi\lambda = 0$ at the optimum. Having concluded that ξ is strictly positive, we get that λ must equal zero. Plugging $\lambda = 0$ into Eq. (61) gives $\tau = C$. Summing up, we used the KKT conditions to show that in the case where $\ell_t/\|\mathbf{x}_t\|^2 > C$, it is optimal to select $\tau = C$. Folding all of the possible cases into a single equation, we define τ_t to be,

$$\tau_t = \min\left\{ C \;,\; \ell_t/\|\mathbf{x}_t\|^2 \right\}. \tag{63}$$

The update of PA-I is like the update of PA clipped at C.

Turning to the update of PA-II , we again recall that $\ell_t = 0$ leads to $\tau_t = 0$, and deal with those rounds where $\ell_t > 0$. The Lagrangian of the optimization problem in Eq. (7) equals,

$$\mathcal{L}(\mathbf{w},\xi,\tau) \;=\; \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \;+\; C\xi^2 \;+\; \tau\big(1-\xi-y_t(\mathbf{w}\cdot\mathbf{x}_t)\big), \tag{64}$$

where $\tau \geq 0$ is a Lagrange multiplier. Again, differentiating this Lagrangian with respect to the elements of $\mathbf{w}$ and setting these partial derivatives to zero gives Eq. (5) and we can write $\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t$. Differentiating the Lagrangian with respect to ξ and setting that partial derivative to zero results in,

$$0 \;=\; \frac{\partial\mathcal{L}(\mathbf{w},\xi,\tau)}{\partial\xi} \;=\; 2C\xi - \tau \qquad\Longrightarrow\qquad \xi = \frac{\tau}{2C}.$$

Expressing ξ as above and replacing $\mathbf{w}$ in Eq. (60) with $\mathbf{w}_t + \tau y_t \mathbf{x}_t$, we rewrite the Lagrangian as,

$$\mathcal{L}(\tau) \;=\; -\frac{\tau^2}{2}\left(\|\mathbf{x}_t\|^2 \;+\; \frac{1}{2C}\right) \;+\; \tau\big(1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)\big).$$

Setting the derivative of the above to zero gives,

$$0 \;=\; \frac{\partial\mathcal{L}(\tau)}{\partial\tau} \;=\; -\tau\left(\|\mathbf{x}_t\|^2 + \frac{1}{2C}\right) + \big(1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)\big) \qquad\Longrightarrow\qquad \tau = \frac{1-y_t(\mathbf{w}_t\cdot\mathbf{x}_t)}{\|\mathbf{x}_t\|^2 \;+\; \frac{1}{2C}}.$$

As in PA and PA-I , we can give a definition of τ_t which holds in all cases,

$$\tau_t \;=\; \frac{\ell_t}{\|\mathbf{x}\|^2 + \frac{1}{2C}}. \tag{65}$$

References

S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3): 382–392, 1954.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

M. Collins. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. A new family of online algorithms for category ranking. *Jornal of Machine Learning Research*, 3:1025–1058, 2003a.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Jornal of Machine Learning Research*, 3:951–991, 2003b.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004a.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. In *Advances in Neural Information Processing Systems 17*, 2004b.

A. Elisseeff and J. Weston. A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

C. Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3), 2002.

D.P Helmbold, J. Kivinen, and M. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 1999.

M. Herbster. Learning additive models online with fast evaluating kernels. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, pages 444–460, 2001.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2002.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

N. Klasner and H.U. Simon. From noise-free to noise-tolerant and from on-line to batch learning. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 250–264, 1995.

Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1–3): 361–387, 2002.

N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms*. PhD thesis, U. C. Santa Cruz, March 1989.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume XII, pages 615–622, 1962.

J. Rocchio. Relevance feedback information retrieval. In Gerard Salton, editor, *The Smart retrieval system—experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5), 1988.

R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 80–91, 1998. To appear, *Machine Learning*.

R.E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 32(2/3), 2000.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

S. Shalev-Shwartz, J. Keshet, and Y. Singer. Learning to align polyphonic music. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004a. http://www.cs.huji.ac.il/∼shais/.

S. Shalev-Shwartz, Y. Singer, and A. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004b.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.

Journal of Machine Learning Research 7 (2006) 587–602 Submitted 12/04; Revised 12/05; Published 4/06

Toward Attribute Efficient Learning of Decision Lists and Parities

Adam R. Klivans[*] KLIVANS@CS.UTEXAS.EDU
Department of Computer Science
University of Texas at Austin
Austin, TX 78712, USA

Rocco A. Servedio[†] ROCCO@CS.COLUMBIA.EDU
Department of Computer Science
Columbia University
New York, NY 10027, USA

Editor: Dana Ron

Abstract

We consider two well-studied problems regarding attribute efficient learning: learning decision lists and learning parity functions. First, we give an algorithm for learning decision lists of length k over n variables using $2^{\tilde{O}(k^{1/3})}\log n$ examples and time $n^{\tilde{O}(k^{1/3})}$. This is the first algorithm for learning decision lists that has both subexponential sample complexity and subexponential running time in the relevant parameters. Our approach establishes a relationship between attribute efficient learning and polynomial threshold functions and is based on a new construction of low degree, low weight polynomial threshold functions for decision lists. For a wide range of parameters our construction matches a lower bound due to Beigel for decision lists and gives an essentially optimal tradeoff between polynomial threshold function degree and weight.

Second, we give an algorithm for learning an unknown parity function on k out of n variables using $O(n^{1-1/k})$ examples in $\mathrm{poly}(n)$ time. For $k = o(\log n)$ this yields a polynomial time algorithm with sample complexity $o(n)$; this is the first polynomial time algorithm for learning parity on a superconstant number of variables with sublinear sample complexity. We also give a simple algorithm for learning an unknown length-k parity using $O(k\log n)$ examples in $n^{k/2}$ time, which improves on the naive n^k time bound of exhaustive search.

Keywords: PAC learning, attribute efficiency, learning parity, decision lists, Winnow

1. Introduction

An important goal in machine learning theory is to design *attribute efficient* algorithms for learning various classes of Boolean functions. A class C of Boolean functions over n variables $x_1,\ldots,x_n$ is said to be *attribute efficiently learnable* if there is a $\mathrm{poly}(n)$ time algorithm which can learn any function $f \in C$ using a number of examples which is polynomial in the "size" (description length) of the function f to be learned, rather than in n, the number of features in the domain over which learning takes place. (Note that the running time of the learning algorithm must in general be at least n since each example is an n-bit vector.) Thus an attribute efficient learning algorithm for e.g. the class of Boolean conjunctions must be able to learn any Boolean conjunction of k literals over $x_1,\ldots,x_n$ using $\mathrm{poly}(k,\log n)$ examples, since $k\log n$ bits are required to specify such a conjunction.

∗. Work done at Harvard University and supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.
†. Supported in part by NSF CAREER award CCF-0347282.

A longstanding open problem in machine learning, posed first by Blum (1990) and subsequently by various authors (Blum, 1996; Blum et al., 1995; Blum and Langley, 1997; Valiant, 1999), is whether or not there exist attribute efficient algorithms for learning *decision lists*, which are essentially nested "if-then-else" statements (we give a precise definition in Section 2). One motivation for considering the problem comes from the *infinite attribute model* introduced in Blum (1990). Blum et al. (1995) showed that for many concept classes (including decision lists) attribute efficient learnability in the standard n-attribute model is equivalent to learnability in the infinite attribute model. Since simple classes such as disjunctions and conjunctions are attribute efficiently learnable (and hence learnable in the infinite attribute model), this motivated Blum (1990) to ask whether the richer class of decision lists is thus learnable as well. Several researchers (Blum, 1996; Blum and Langley, 1997; Dhagat and Hellerstein, 1994; Nevo and El-Yaniv, 2002; Servedio, 2000) have since considered this problem; we summarize this previous work in Section 1.2. More recently, Valiant (1999) relates the problem of learning decision lists attribute efficiently to questions about human learning abilities.

Another outstanding challenge in machine learning is to determine whether there exist attribute efficient algorithms for learning *parity functions*. The parity function on a set of 0/1-valued variables $x_{i_1},\ldots,x_{i_k}$ takes value $+1$ or -1 depending on whether $x_{i_1}+\cdots+x_{i_k}$ is even or odd. As with decision lists, a simple PAC learning algorithm is known for the class of parity functions but no attribute efficient algorithm is known.

1.1 Our Results

We give the first learning algorithm for decision lists that is subexponential in both sample complexity (in the relevant parameters k and $\log n$) and running time (in the relevant parameter k). Our results demonstrate for the first time that it is possible to simultaneously avoid the "worst case" in both sample complexity and running time, and thus suggest that it may perhaps be possible to learn decision lists attribute efficiently. Our main learning result for decision lists is:

Theorem 1 *There is an algorithm which learns length-k decision lists over $\{0,1\}^n$ with mistake bound $2^{\tilde{O}(k^{1/3})}\log n$ and time $n^{\tilde{O}(k^{1/3})}$.*

This bound improves on the sample complexity of Littlestone's well-known Winnow algorithm (Littlestone, 1988) for all k and improves on its running time as well for $k=\Omega(\log^{3/2} n)$; see Section 1.2.

We prove Theorem 1 in two parts; first we generalize the Winnow algorithm for learning linear threshold functions to learn *polynomial threshold functions* (PTFs). In recent work on learning DNF formulas (Klivans and Servedio, 2004), intersections of halfspaces (Klivans et al., 2004), and Boolean formulas of superconstant depth (O'Donnell and Servedio, 2003), PTFs of degree d have been learned in time $n^{O(d)}$ by using polynomial time linear programming algorithms such as the Ellipsoid algorithm (see Klivans and Servedio, 2004). In contrast, since we want to achieve low sample complexity as well as an $n^{O(d)}$ running time, we use a generalization of the Winnow algorithm to learn PTFs. This generalization has sample complexity and running time bounds which depend on the degree and the total magnitude of the integer coefficients (which we call the weight) of the PTF:

Theorem 2 *Let $\mathcal{C}$ be a class of Boolean functions over $\{0,1\}^n$ with the property that each $f \in \mathcal{C}$ has a PTF of degree at most d and weight at most W. Then there is an online learning algorithm for $\mathcal{C}$ which runs in n^d time per example and has mistake bound $O(W^2 \cdot d \cdot \log n)$.*

This reduces the decision list learning problem to a problem of representing decision lists with PTFs of low weight and low degree. To this end we prove:

Theorem 3 *Let L be a decision list of length k. Then L is computed by a polynomial threshold function of degree $\tilde{O}(k^{1/3})$ and weight $2^{\tilde{O}(k^{1/3})}$.*

Theorem 1 follows directly from Theorems 2 and 3. We emphasize that Theorem 3 does *not* follow from previous results (Klivans and Servedio, 2004) on representing DNF formulas as PTFs; the PTF construction from Klivans and Servedio (2004) in fact has exponentially larger weight ($2^{2^{\tilde{O}(k^{1/3})}}$ rather than $2^{\tilde{O}(k^{1/3})}$) than the construction in this paper.

Our PTF construction is essentially optimal in the tradeoff between degree and weight that it achieves. In 1994 Beigel (1994) gave a lower bound showing that any degree d PTF for a certain decision list must have weight $2^{\Omega(n/d^2)}$. [1] For $d = n^{1/3}$, Beigel's lower bound implies that our construction in Theorem 3 is essentially the best possible.

For parity functions, we give an $O(n^4)$ time algorithm which can PAC learn an unknown parity on k variables out of n using $\tilde{O}(n^{1-1/k})$ examples. To our knowledge this is the first algorithm for learning parity on a superconstant number of variables with sublinear sample complexity. Our algorithm works by finding a "low weight" solution to a system of m linear equations (corresponding to a set of m examples). We prove that with high probability we can find a solution of weight $O(n^{1-1/k})$ irrespective of m. Thus by taking m to be only slightly larger than $n^{1-1/k}$, standard arguments show that our solution is a good hypothesis.

We also describe a simple algorithm, due to Dan Spielman, for learning an unknown parity on k variables using $O(k \log n)$ examples and $\tilde{O}(n^{k/2})$ time. This gives a square root running time improvement over a naive $O(n^k)$ exhaustive search.

1.2 Previous Results

In previous work several algorithms with different performance bounds (running time and sample complexity) have been given for learning length-k decision lists.

- Rivest (1987) gave the first algorithm for learning decision lists in Valiant's PAC model of learning from random examples. Littlestone (Blum, 1996) later gave an analogue of Rivest's algorithm in the online learning model. The algorithm can learn any decision list of length k in $O(kn^2)$ time using $O(kn)$ examples.

- A brute-force approach is to maintain the set of all length-k decision lists which are consistent with the examples seen so far, and to predict at each stage using majority vote over the surviving hypotheses. This "halving algorithm," proposed in various forms in Angluin (1988); Barzdin and Freivald (1972); Mitchell (1982), can learn decision lists of length k using only $O(k \log n)$ examples, but the running time is $n^{\Theta(k)}$.

1. Krause (2002) claims a lower bound of degree d and weight $2^{\Omega(n/d)}$ for a particular decision list; this claim, however, is in error.

- Several researchers (Blum, 1996; Valiant, 1999) have observed that Winnow can learn length-k decision lists from $2^{O(k)}\log n$ examples in time $2^{O(k)}n\log n$. This follows from the fact that any decision list of length k can be expressed as a linear threshold function with integer coefficients of magnitude $2^{\Theta(k)}$.

- Finally, several researchers have considered the special case of learning a length-k decision list in which the output bits of the list have at most D alternations. Valiant (1999) and Nevo and El-Yaniv (2002) have given refined analyses of Winnow's performance for this case (see Dhagat and Hellerstein, 1994). However, for the general case where D can be as large as k, these results do not improve on the standard Winnow analysis described above.

Note that all of these earlier algorithms have an exponential dependence on at least one of the relevant parameters (k and $\log n$ for sample complexity, k for running time).

Little previous work has been published on learning parity functions attribute efficiently in the PAC model. The standard PAC learning algorithm for parity (based on solving a system of linear equations) is due to Helmbold et al. (1992); however this algorithm is not attribute efficient since it uses $\Omega(n)$ examples regardless of k. Several authors have considered learning parity attribute efficiently in a model where the learner is allowed to make membership queries. Attribute efficient learning is easier in this framework since membership queries can help identify relevant variables. Blum et al. (1995) give a randomized polynomial time membership-query algorithm for learning parity on k variables using only $O(k\log n)$ examples, and these results were later refined Uehara et al. (2000).

1.3 Organization

In Section 2 we give necessary background. In Section 3 we show how to reduce the decision list learning problem to a problem of finding suitable PTF representations of decision lists (Theorem 2). In Section 4 we give our PTF construction for decision lists (Theorem 3). In Section 5 we discuss the connection between Theorem 3 and Beigel's ODDMAXBIT lower bound. In Section 6 we give our results on learning parity functions, and we conclude in Section 7.

2. Preliminaries

Attribute efficient learning has been chiefly studied in the *online mistake-bound* model of concept learning which was introduced in Littlestone (1988, 1989a). In this model learning proceeds in a series of trials, where in each trial the learner is given an unlabeled boolean example $x \in \{0,1\}^n$ and must predict the value $f(x)$ of the unknown target function f. After each prediction the learner is given the true value of $f(x)$ and can update its hypothesis before the next trial begins. The *mistake bound* of a learning algorithm on a target concept c is the worst-case number of mistakes that the algorithm makes over all (possibly infinite) sequences of examples, and the mistake bound of a learning algorithm on a concept class (class of Boolean functions) C is the worst-case mistake bound across all functions $f \in C$. The running time of a learning algorithm A for a concept class C is defined as the product of the mistake bound of A on C times the maximum running time required by A to evaluate its hypothesis and update its hypothesis in any trial.

Our main interests are the classes of *decision lists* and *parity functions*. A decision list L of length k over the Boolean variables $x_1,\ldots,x_n$ is represented by a list of k pairs and a bit (ℓ_1,b_1), $(\ell_2,b_2),\ldots,(\ell_k,b_k),b_{k+1}$ where each ℓ_i is a literal and each b_i is either -1 or 1. Given any $x \in$

$\{0,1\}^n$, the value of $L(x)$ is b_i if i is the smallest index such that ℓ_i is made true by x; if no ℓ_i is true then $L(x) = b_{k+1}$. A parity function of length k is defined by a set of variables $S \subset \{x_1, \ldots, x_n\}$ such that $|S| = k$. The parity function $\chi_S(x)$ takes value 1 (-1) on inputs which set an even (odd) number of variables in S to 1.

Given a concept class C over $\{0,1\}^n$ and a Boolean function $f \in C$, let size(f) denote the description length of f under some reasonable encoding scheme. We say that a learning algorithm A for C in the mistake-bound model is *attribute efficient* if the mistake bound of A on any concept $f \in C$ is polynomial in size(f). In particular, the description length of a length k decision list (parity) is $O(k \log n)$, and thus we would ideally like to have poly(n)-time algorithms which learn decision lists (parities) of length k with a mistake bound of poly$(k, \log n)$.

We note here that attribute efficiency has also been studied in other learning models, including Valiant's Probably Approximately Correct (PAC) model of learning from random examples. Standard conversion techniques are known (Angluin, 1988; Haussler, 1988; Littlestone, 1989b) which can be used to transform any mistake bound algorithm into a PAC learning algorithm. These transformations essentially preserve the running time of the mistake bound algorithm, and the sample size required by the PAC algorithm is essentially the mistake bound. Thus, positive results for mistake bound learning, such as those we give for decision lists in this paper, directly yield corresponding positive results for the PAC model.

Finally, our results for decision lists are achieved by a careful analysis of *polynomial threshold functions*. Let f be a Boolean function $f : \{0,1\}^n \to \{-1,1\}$ and let p be a polynomial in n variables with integer coefficients. Let d denote the degree of p and let W denote the sum of the absolute values of p's integer coefficients. If the sign of $p(x)$ equals $f(x)$ for every $x \in \{0,1\}^n$, then we say that p is a *polynomial threshold function (PTF) of degree d and weight W* for f.

3. Expanded-Winnow: Learning Polynomial Threshold Functions

Littlestone (1988) introduced the online Winnow algorithm and showed that it can attribute efficiently learn Boolean conjunctions, disjunctions, and low weight linear threshold functions. Throughout its execution Winnow maintains a linear threshold function as its hypothesis; at the heart of the algorithm is an update rule which makes a multiplicative update to each coefficient of the hypothesis each time a mistake is made. Since its introduction Winnow has been intensively studied from both applied and theoretical standpoints (see Blum, 1997; Golding and Roth, 1999; Kivinen et al., 1997; Servedio, 2002).

The following theorem due to Littlestone (1988) gives a mistake bound for Winnow for linear threshold functions:

Theorem 4 *Let $f(x)$ be the linear threshold function sign$(\sum_{i=1}^n w_i x_i - \theta)$ over inputs $x \in \{0,1\}^n$ where θ and $w_1, \ldots, w_n$ are integers. Let $W = \sum_{i=1}^n |w_i|$. Then Winnow learns $f(x)$ with mistake bound $O(W^2 \log n)$ and uses $O(n)$ time steps per example.*

We will use a generalization of the Winnow algorithm, which we call Expanded-Winnow, to learn *polynomial* threshold functions of degree at most d. Our generalization introduces $\sum_{i=1}^d \binom{n}{i}$ new variables (one for each monomial of degree up to d) and runs Winnow to learn a linear threshold function over these new variables. More precisely, in each trial we convert the n-bit received example $x = (x_1, \ldots, x_n)$ into a $\sum_{i=1}^d \binom{n}{i}$ bit expanded example (where the bits in the expanded example correspond to monomials over $x_1, \ldots, x_n$), and we give the expanded example to Winnow. Thus

the hypothesis which Winnow maintains – a linear threshold function over the space of expanded features – is a polynomial threshold function of degree d over the original n variables $x_1,\ldots,x_n$. Theorem 2, which follows directly from Theorem 4, summarizes the performance of Expanded-Winnow:

Theorem 2 *Let $\mathcal{C}$ be a class of Boolean functions over $\{0,1\}^n$ with the property that each $f \in \mathcal{C}$ has a polynomial threshold function of degree at most d and weight at most W. Then Expanded-Winnow algorithm runs in $n^{O(d)}$ time per example and has mistake bound $O(W^2 \cdot d \cdot \log n)$ for $\mathcal{C}$.*

Theorem 2 shows that the degree of a polynomial threshold function strongly affects Expanded-Winnow's running time, and the weight of a polynomial threshold function strongly affects its sample complexity.

4. Constructing PTFs for Decision Lists

In previous constructions of polynomial threshold functions for computational learning theory applications (Klivans and Servedio, 2004; Klivans et al., 2004; O'Donnell and Servedio, 2003) the sole goal has been to minimize the degree of the polynomials regardless of the size of the coefficients. As one example, the construction of Klivans and Servedio (2004) of $\tilde{O}(n^{1/3})$ degree PTFs for DNF formulae yields polynomials whose coefficients can be doubly exponential in the degree. In contrast, we must now construct PTFs that have low degree and low weight.

We give two constructions of PTFs for decision lists, each of which has relatively low degree and relatively low weight. We then combine these to achieve an optimal construction with improved bounds on both degree and weight.

4.1 Outer Construction

Let L be a decision list of length k over variables $x_1,\ldots,x_k$. We first give a simple construction of a degree h, weight $2^{O(k/h+h)}$ PTF for L which is based on breaking the list L into sublists. We call this construction the "outer construction" since we will ultimately combine this construction with a different construction for the "inner" sublists.

We begin by showing that L can be expressed as a threshold of *modified decision lists*, which we now define. The set $\mathcal{B}_h$ of modified decision lists is defined as follows: each function in $\mathcal{B}_h$ is a decision list $(\ell_1,b_1),(\ell_2,b_2),\ldots,(\ell_h,b_h),0$ where each ℓ_i is some literal over $x_1,\ldots,x_n$ and each $b_i \in \{-1,1\}$. Thus the only difference between a modified decision list $f \in \mathcal{B}_h$ and a normal decision list of length h is that the final output value is 0 rather than $b_{h+1} \in \{-1,+1\}$.

Now assume we have a list $L = (\ell_1,b_1),\ldots,(\ell_k,b_k),b_{k+1}$. We break L sequentially into k/h blocks each of length h (assume k/h is an integer, otherwise we can use $\lceil k/h \rceil$ everywhere). Let $f_i \in \mathcal{B}_h$ be the modified decision list which corresponds to the ith block of L, i.e. f_i is the list $(\ell_{(i-1)h+1},b_{(i-1)h+1}),\ldots,(\ell_{(i+1)h},b_{(i+1)h}),0$. Intuitively f_i computes the ith block of L and equals 0 only if we "fall of the edge" of the ith block. We then have the following straightforward claim:

Claim 5 *The decision list L is eqivalent to*

$$sign\left(\sum_{i=1}^{k/h} 2^{k/h-i+1} f_i(x) \;+\; b_{k+1}\right). \tag{1}$$

Proof Given an input x let $r = (i-1)h + c$ be the first index such that ℓ_r is satisfied. It is easy to see that $f_j(x) = 0$ for $j < i$ and hence the value in (1) is $2^{k/h-i+1}b_r + \sum_{j=i+1}^{k/h} 2^{k/h-j+1} f_j(x) + b_{k+1}$, the sign of which is easily seen to be b_r. Finally, if no literal is satisfied then the argument to (1) is b_{k+1}. ■

Note: It is easily seen that we can replace the 2 in formula (1) by a 3; this will prove useful later.

As an aside, note that Claim 5 can already be used to obtain a tradeoff between running time and sample complexity for learning decision lists. The class $\mathcal{B}_h$ contains at most $(4n)^h$ functions. Thus as in Section 3 it is possible to run the Winnow algorithm using the functions in $\mathcal{B}_h$ as the base features for Winnow. (So for each example x which it receives, the algorithm would first compute the value of $f(x)$ for each $f \in \mathcal{B}_h$, and would then use this vector of $(f(x))_{f \in \mathcal{B}_h}$ values as the example point for Winnow.) A direct analogue of Theorem 2 now implies that Expanded-Winnow (run over this expanded feature space of functions from $\mathcal{B}_h$) can be used to learn L_k in time $n^{O(h)}2^{O(k/h)}$ with mistake bound $2^{O(k/h)}h\log n$.

However, it will be more useful for us to obtain a PTF for L. We can do this from Claim 5 as follows:

Theorem 6 *Let L be a decision list of length k. For any $h < k$ we have that L is computed by a polynomial threshold function of degree h and weight $2^{O(k/h+h)}$.*

Proof Consider the first modified decision list $f_1 = (\ell_1, b_1), (\ell_2, b_2), \ldots, (\ell_h, b_h), 0$ in the expression (1). For ℓ a literal let $\tilde{\ell}$ denote x_i if ℓ is an unnegated variable x_i and let $\tilde{\ell}$ denote $1 - x_i$ if if ℓ is a negated variable $\bar{x}_i$. We have that for all $x \in \{0,1\}^h$, $f_1(x)$ is computed exactly by the polynomial

$$f_1(x) = \tilde{\ell}_1 b_1 + (1-\tilde{\ell}_1)\tilde{\ell}_2 b_2 + (1-\tilde{\ell}_1)(1-\tilde{\ell}_2)\tilde{\ell}_3 b_3 + \cdots + (1-\tilde{\ell}_1)\cdots(1-\tilde{\ell}_{h-1})\tilde{\ell}_h b_h.$$

This polynomial has degree h and has weight at most 2^{h+1}. Summing these polynomial representations for $f_1, \ldots, f_{k/h}$ as in (1) we see that the resulting PTF given by (1) has degree h and weight at most $2^{k/h+1} \cdot 2^{h+1} = 2^{O(k/h+h)}$. ■

Specializing to the case $h = \sqrt{k}$ we obtain:

Corollary 7 *Let L be a decision list of length k. Then L is computed by a polynomial threshold function of degree $k^{1/2}$ and weight $2^{O(k^{1/2})}$.*

We close this section by observing that an intermediate result of Klivans and Servedio (2004) can be used to give an alternate proof of Corollary 7 with slightly weaker parameters; however our later proofs require the construction given in this section.

4.2 Inner Approximator

In this section we construct low degree, low weight polynomials which approximate (in the L_∞ norm) the modified decision lists from the previous subsection. Moreover, the polynomials we construct are exactly correct on inputs which "fall off the end":

Theorem 8 *Let $f \in \mathcal{B}_h$ be a modified decision list of length h. Then there is a degree $O(\sqrt{h}\log h)$ polynomial p such that*

- *for every input $x \in \{0,1\}^h$ we have $|p(x) - f(x)| \leq 1/h$.*
- *$f(x) = 0$ implies $p(x) = 0$.*

Proof

We construct a PTF satisfying the above requirements for a decision list f of the form (x_1, b_1), $\ldots, (x_h, b_h), 0$. The proof for a general modified decision list is similar. As in the proof of Theorem 6 we have that

$$f(x) = b_1 x_1 + b_2(1-x_1)x_2 + \cdots + b_h(1-x_1)\cdots(1-x_{h-1})x_h.$$

We will construct a lower (roughly $\sqrt{h}$) degree polynomial which closely approximates f. Essentially this construction has been done several times before (see Klivans and Servedio, 2004; Klivans et al., 2004).

Let T_i denote $(1-x_1)\ldots(1-x_{i-1})x_i$, so we can rewrite f as

$$f(x) = b_1 T_1 + b_2 T_2 + \cdots + b_h T_h.$$

We approximate each T_i separately as follows: set $A_i(x) = h - i + x_i + \sum_{j=1}^{i-1}(1-x_j)$. Note that for $x \in \{0,1\}^h$, we have $T_i(x) = 1$ iff $A_i(x) = h$ and $T_i(x) = 0$ iff $0 \leq A_i(x) \leq h-1$. Let $d = \lceil \sqrt{h} \rceil$. Now define the polynomial

$$Q_i(x) = q(A_i(x)/h) \quad \text{where} \quad q(y) = C_d(y(1+1/h)).$$

As in Klivans and Servedio (2004), here $C_d(x)$ is the dth Chebyshev polynomial of the first kind (a univariate polynomial of degree d). We will need the following facts about Chebyshev polynomials (Cheney, 1966):

- $|C_d(x)| \leq 1$ for $|x| \leq 1$ with $C_d(1) = 1$;
- $C_d'(x) \geq d^2$ for $x > 1$ with $C_d'(1) = d^2$;
- The coefficients of C_d are integers each of whose magnitude is at most 2^d.

The first two facts imply that $q(1) \geq 2$ but $|q(y)| \leq 1$ for $y \in [0, 1 - \frac{1}{h}]$. We thus have that $Q_i(x) = q(1) \geq 2$ if $T_i(x) = 1$ and $|Q_i(x)| \leq 1$ if $T_i(x) = 0$. Now define $P_i(x) = \left(\frac{Q_i(x)}{q(1)}\right)^{2\log h}$. This polynomial is easily seen to be a good approximator for T_i: if $x \in \{0,1\}^h$ is such that $T_i(x) = 1$ then $P_i(x) = 1$, and if $x \in \{0,1\}^h$ is such that $T_i(x) = 0$ then $|P_i(x)| < \left(\frac{1}{2}\right)^{2\log h} < \frac{1}{h^2}$.

Now define $R(x) = \sum_{i=1}^{\ell} b_i P_i(x)$ and $p(x) = R(x) - R(0^h)$. It is clear that $p(0^h) = 0$. We will show that for every input $0^h \neq x \in \{0,1\}^h$ we have $|p(x) - f(x)| \leq 1/h$. Fix such an x; let i be the first index such that $x_i = 1$. As shown above we have $P_i(x) = 1$. Moreover, by inspection of $T_j(x)$ we have that $T_j(x) = 0$ for all $j \neq i$, and hence $|P_j(x)| < \frac{1}{h^2}$. Consequently the value of $R(x)$ must lie in $[b_i - \frac{h-1}{h^2}, b_i + \frac{h-1}{h^2}]$. Since $|R(0^h)|$ is at most ℓ/h^2 and $f(x) = b_i$, we have that $p(x)$ is an L_∞ approximator for $f(x)$ as desired.

Finally, it is straightforward to verify that $p(x)$ has the claimed degree. ■

Strictly speaking we cannot discuss the weight of the polynomial p since its coefficients are rational numbers but not integers. However, by multiplying p by a suitable integer (clearing denominators) we obtain an integer polynomial with essentially the same properties. Using the third fact about Chebyshev polynomials from our proof above, we have that $q(1)$ is a rational number N_1/N_2 where N_1 and N_2 are both integers of magnitude $h^{O(\sqrt{h})}$. Each $Q_i(x)$ for $i = 1,\ldots,h$ can be written as an integer polynomial (of weight $h^{O(\sqrt{h})}$) divided by $h^{\sqrt{h}}$. Thus each $P_i(x)$ can be written as $\tilde{P}_i(x)/(h^{\sqrt{h}}N_1)^{2\log h}$ where $\tilde{P}_i(x)$ is an integer polynomial of weight $h^{O(\sqrt{h}\log h)}$. It follows that $p(x)$ equals $\tilde{p}(x)/C$, where C is an integer which is at most $2^{O(h^{1/2}\log^2 h)}$ and $\tilde{p}$ is a polynomial with integer coefficients and weight $2^{O(h^{1/2}\log^2 h)}$. We thus have

Corollary 9 *Let $f \in \mathcal{B}_h$ be a modified decision list of length h. Then there is an integer polynomial $p(x)$ of degree $2\sqrt{h}\log h$ and weight $2^{O(h^{1/2}\log^2 h)}$ and an integer $C = 2^{O(h^{1/2}\log^2 h)}$ such that*

- *for every input $x \in \{0,1\}^h$ we have $|p(x) - Cf(x)| \leq C/h$.*
- *$f(x) = 0$ implies $p(x) = 0$.*

The fact that $p(x)$ is exactly 0 when $f(x)$ is 0 will be important in the next subsection when we combine the inner approximator with the outer construction.

4.3 Composing the Constructions

In this section we combine the two constructions from the previous subsections to obtain our main polynomial threshold construction:

Theorem 10 *Let L be a decision list of length k. Then for any $h < k$, L is computed by a polynomial threshold function of degree $O(h^{1/2}\log h)$ and weight $2^{O(k/h+h^{1/2}\log^2 h)}$.*

Proof Again assume L is the decision list $(x_1,b_1),\ldots,(x_k,b_k)$, b_{k+1} (the case when L contains negated literals is entirely similar). We begin with the outer construction: from the note following Claim 5 we have that

$$L(x) = \text{sign}\left(C\left[\sum_{i=1}^{k/h} 3^{k/h-i+1} f_i(x) \;+\; b_{k+1}\right]\right)$$

where C is the value from Corollary 9 and each f_i is a modified decision list of length h computing the restriction of L to its ith block as defined in Subsection 4.1. Now we use the inner approximator to replace each Cf_i above by p_i, the approximating polynomial from Corollary 9, i.e. consider $\text{sign}(H(x))$ where

$$H(x) = \sum_{i=1}^{k/h} (3^{k/h-i+1} p_i(x)) \;+\; Cb_{k+1}.$$

We will show that $\text{sign}(H(x))$ is a PTF which computes L correctly and has the desired degree and weight.

Fix any $x \in \{0,1\}^k$. If $x = 0^k$ then by Corollary 9 each $p_i(x)$ is 0 so $H(x) = Cb_{k+1}$ has the right sign. Now suppose that $r = (i-1)h + c$ is the first index such that $x_r = 1$. By Corollary 9, we have that

- $3^{k/h-j+1}p_j(x) = 0$ for $j < i$;
- $3^{k/h-i+1}p_i(x)$ differs from $3^{k/h-i+1}Cb_r$ by at most $C3^{k/h-i+1} \cdot \frac{1}{h}$;
- The magnitude of each value $3^{k/h-j+1}p_j(x)$ is at most $C3^{k/h-j+1}(1+\frac{1}{h})$ for $j > i$.

Combining these bounds, the value of $H(x)$ differs from $3^{k/h-i+1}Cb_r$ by at most

$$C\left(\frac{3^{k/h-i+1}}{h} + \left(1+\frac{1}{h}\right)\left[3^{k/h-i} + 3^{k/h-i-1} + \cdots + 3\right] + 1\right)$$

which is easily seen to be less than $C3^{k/h-i+1}$ in magnitude (for $h > 1$). Thus the sign of $H(x)$ equals b_r, and consequently $\text{sign}(H(x))$ is a valid polynomial threshold representation for $L(x)$. Finally, our degree and weight bounds from Corollary 9 imply that the degree of $H(x)$ is $O(h^{1/2}\log h)$ and the weight of $H(x)$ is $2^{O(k/h)+O(h^{1/2}\log^2 h)}$, and the theorem is proved. ■

Taking $h = k^{2/3}/\log^{4/3} k$ in the above theorem we obtain our main result on representing decision lists as polynomial threshold functions:

Theorem 3 *Let L be a decision list of length k. Then L is computed by a polynomial threshold function of degree $k^{1/3}\log^{1/3} k$ and weight $2^{O(k^{1/3}\log^{4/3} k)}$.*

Theorem 3 immediately implies that Expanded-Winnow can learn decision lists of length k using $2^{\tilde{O}(k^{1/3})}\log n$ examples and time $n^{\tilde{O}(k^{1/3})}$.

4.4 Application to Learning Decision Trees

Ehrenfeucht and Haussler (1989) gave an a time $n^{O(\log s)}$ algorithm for learning decision trees with s leaves over n variables. Their algorithm uses $n^{O(\log s)}$ examples, and they asked if the sample complexity could be reduced to $\text{poly}(n,s)$. We can apply our techniques here to give an algorithm using $2^{\tilde{O}(s^{1/3})}\log n$ examples, if we are willing to spend $n^{\tilde{O}(s^{1/3})}$ time.

First we need to generalize Theorem 10 for higher order decision lists. An r-decision list is like a standard decision list but each pair is now of the form (C_i, b_i) where C_i is a conjunction of at most r literals and as before $b_i = \pm 1$. The output of such an r-decision list on input x is b_i where i is the smallest index such that $C_i(x) = 1$.

We have the following:

Corollary 11 *Let L be an r-decision list of length k. Then for any $h < k$, L is computed by a polynomial threshold function of degree $O(rh^{1/2}\log h)$ and weight $2^{r+O(k/h+h^{1/2}\log^2 h)}$.*

Proof Let L be the r-decision list $(C_1, b_1), \ldots, (C_k, b_k), b_{k+1}$. By Theorem 10 there is a polynomial threshold function of degree $O(h^{1/2}\log h)$ and weight $2^{O(k/h+h^{1/2}\log^2 h)}$ over the variables $C_1, \ldots, C_k$. Now replace each variable C_i by the interpolating polynomial which computes it exactly as a function from $\{0,1\}^n$ to $\{0,1\}$. Each such interpolating polynomial has degree r and integer coefficients of total magnitude at most 2^r, and the corollary follows. ■

Corollary 12 *There is an algorithm for learning r-decision lists over $\{0,1\}^n$ which, when learning an r-decision list of length k, has mistake bound $2^{\tilde{O}(r+k^{1/3})}\log n$ and runs in time $n^{\tilde{O}(rk^{1/3})}$.*

Now we can apply Corollary 12 to obtain a tradeoff between running time and sample complexity for learning decision trees:

Theorem 13 *Let D be a decision tree of size s over n variables. Then D can be learned with mistake bound $2^{\tilde{O}(s^{1/3})}\log n$ in time $n^{\tilde{O}(s^{1/3})}$.*

Proof Blum (1992) has shown that any decision tree of size s is computed by a $(\log s)$-decision list of length s. Applying Corollary 12 we thus see that Expanded-Winnow can be used to learn decision trees of size s over $\{0,1\}^n$ with the claimed bounds on time and sample complexity. ■

5. Lower Bounds for Decision Lists

Here we observe that our construction from Theorem 10 is essentially optimal in terms of the tradeoff it achieves between polynomial threshold function degree and weight.

Beigel (1994) constructs an oracle separating PP from P^{NP}. At the heart of his construction is a proof that any low degree PTF for a particular decision list called the ODDMAXBIT$_n$ function must have large weights:

Definition 14 *The* ODDMAXBIT$_n$ *function on input $x = x_1,\ldots,x_n \in \{0,1\}^n$ equals $(-1)^i$ where i is the index of the first nonzero bit in x.*

It is clear that the ODDMAXBIT$_n$ function is equivalent to a decision list $(x_1,-1)$, $(x_2,1)$, $(x_3,-1),\ldots,(x_n,(-1)^n),(-1)^{n+1}$ of length n. The main technical theorem that Beigel proves is as follows:

Theorem 15 *Let p be a degree d PTF with integer coefficients which computes* ODDMAXBIT$_n$. *Then $w = 2^{\Omega(n/d^2)}$ where w is the weight of p.*

(As stated in Beigel (1994) the bound is actually $w \geq \frac{1}{s}2^{\Omega(n/d^2)}$ where s is the number of nonzero coefficients in p. Since $s \leq w$ this implies the result as stated above.)

A lower bound of $2^{\Omega(n)}$ on the weight of any linear threshold function ($d = 1$) for ODDMAXBIT$_n$ has long been known (Myhill and Kautz, 1961); Beigel's proof generalizes this lower bound to all $d = O(n^{1/2})$. A matching upper bound of $2^{O(n)}$ on weight for $d = 1$ has also long been known (Myhill and Kautz, 1961). Our Theorem 10 gives an upper bound which matches Beigel's lower bound (up to logarithmic factors) for all $d = O(n^{1/3})$:

Observation 16 *For any $d = O(n^{1/3})$ there is a polynomial threshold function of degree d and weight $2^{\tilde{O}(n/d^2)}$ which computes* ODDMAXBIT$_n$.

Proof Set $d = h^{1/2}\log h$ in Theorem 10. The weight bound given by Theorem 10 is $2^{O(\frac{n\log^2 d}{d^2}+d\log d)}$ which is $2^{\tilde{O}(n/d^2)}$ for $d = O(n^{1/3})$. ■

Note that since the ODDMAXBIT$_n$ function has a polynomial size DNF, Beigel's lower bound gives a polynomial size DNF f such that any degree $\tilde{O}(n^{1/3})$ polynomial threshold function for f must have weight $2^{\tilde{\Omega}(n^{1/3})}$. This suggests that the Expanded-Winnow algorithm cannot learn polynomial size DNF in $2^{\tilde{O}(n^{1/3})}$ time from $2^{n^{1/3-\varepsilon}}$ examples for any $\varepsilon > 0$, and thus suggests that improving the sample complexity of the DNF learning algorithm from Klivans and Servedio (2004) while maintaining its $2^{\tilde{O}(n^{1/3})}$ running time may be difficult.

6. Learning Parity Functions

Recall that the standard algorithm for learning parity functions works by viewing a set of m labelled examples as a set of m linear equations over $GF(2)$. Gaussian elimination is used to solve the system and thus find a consistent parity. Even though there exists a solution of weight at most k (since the target parity is of length k), Gaussian elimination applied to a system of m equations in n variables over $GF(2)$ may yield a solution of weight as large as $\min(m,n)$. Thus this standard algorithm and analysis give an $O(n)$ sample complexity bound for learning a parity of length at most k.

6.1 A Polynomial Time Algorithm

We now describe a simple poly(n)-time algorithm for PAC learning an unknown length-k parity using $\tilde{O}(n^{1-1/k})$ examples (for a formal definition of the PAC model we refer the reader to the book by Kearns and Vazirani, 1994). As far as we know this is the first improvement on the standard algorithm and analysis described above.

Theorem 17 *The class of all parity functions on at most k variables is PAC learnable in $O(n^4)$ time using $O(n^{1-1/k}\log n)$ examples. The hypothesis output by the learning algorithm is a parity function on $O(n^{1-1/k})$ variables.*

Proof If $k = \Omega(\log n)$ then the standard algorithm suffices to prove the claimed bound. We thus assume that $k = o(\log n)$.

Let $\ell = n^{1-1/k}$. Let H be the set of all parity functions of length at most ℓ. Note that $|H| \leq n^{n^{1-1/k}}$ so $\log|H| \leq n^{1-1/k}\log n$. Consider the following algorithm:

1. Choose $m = \frac{1}{\varepsilon}(\log|H| + \log(1/\delta))$ examples. Express each example as a linear equation in n variables over $GF(2)$ as described above.

2. Randomly choose a set of $n-\ell$ variables and assign them the value 0.

3. Use Gaussian elimination to attempt to solve the resulting system of equations on the remaining ℓ variables. If the system has a solution, output the corresponding parity (of length at most $\ell = n^{1-1/k}$) as the hypothesis. If the system has no solution, output "FAIL."

If the simplified system of equations has a solution, then by a standard Occam's Razor argument (see Kearns and Vazirani, 1994, for details), this solution is a good hypothesis. We will show that the simplified system has a solution with probability $\Omega(1/n)$. The theorem follows by repeating steps 2 and 3 of the above algorithm until a solution is found. An expected $O(n)$ repetitions will suffice, and since Gaussian elimination runs in time $O(n^3)$, the running time of our algorithm is $O(n^4)$.

Let V be the set of k relevant variables on which the unknown parity function depends. It is easy to see that as long as no variable in V is assigned a 0, the resulting simplified system of equations will have a solution. The probability that in Step 2 the $n-\ell$ variables chosen do not include any variables in V is exactly $\binom{n-k}{n-\ell}/\binom{n}{\ell}$ which equals $\binom{n-k}{\ell-k}/\binom{n}{\ell}$. Expanding binomial coefficients we have

$$\begin{aligned}\frac{\binom{n-k}{\ell-k}}{\binom{n}{\ell}} &= \prod_{i=1}^{k}\frac{\ell-k+i}{n-k+i} > \left(\frac{\ell-k}{n-k}\right)^k = \left(\frac{\ell}{n}\right)^k\left(\frac{1-\frac{k}{\ell}}{1-\frac{k}{n}}\right)^k \\ &> \frac{1}{n}\left(1-\frac{k}{\ell}\right)^k > \frac{1}{n}\left(1-\frac{k^2}{\ell}\right) > \frac{1}{2n}\end{aligned}$$

and the proof of the theorem is complete. ■

6.2 An $\tilde{O}(n^{k/2})$ Time Attribute Efficient Algorithm

Spielman (2003) has observed that it is possible to improve on the n^k time bound of a naive search algorithm for learning parity using $k\log n$ examples:

Theorem 18 (Spielman) *The class of all parity functions on at most k variables is PAC learnable in $\tilde{O}(n^{k/2})$ time using $O(k\log n)$ examples. The hypothesis output by the learning algorithm is a parity function on at most k variables.*

Proof By Occam's Razor we need only show that given a set of $m = O(k\log n)$ labelled examples, a consistent length-k parity can be found in $\tilde{O}(n^{k/2})$ time.

Given a labelled example $(x_1,\ldots,x_n;y)$ we will view y as an $(n+1)$st attribute x_{n+1}. Thus our task is to find a set of $(k+1)$ attributes $x_{i_1},\ldots,x_{i_{k+1}}$, one of which must be x_{n+1}, which sum to 0 in every example in the sample.

Let $(x^1;y_1),\ldots(x^m;y_m)$ be the labelled examples in our sample. Given a subset S of variables, let v_S denote the length-m binary vector $(\chi_S(x^1),\ldots,\chi_S(x^m))$ obtained by computing the parity function χ_S on each example in our sample.

We construct two lists, each containing $\binom{n}{k/2}$ vectors of length m. The first list contains all the vectors v_S where S ranges over all $k/2$-element subsets of $\{x_1,\ldots,x_n\}$. The second list contains all the vectors $v_{S\cup\{x_{n+1}\}}$ where S again ranges over all $k/2$-element subsets of $\{x_1,\ldots,x_n\}$.

After sorting these two lists of vectors, which takes $\tilde{O}(n^{k/2})$ time, we scan through them in parallel in time linear in the length of the lists and find a pair of vectors v_{S_1} from the first list and $v_{S_2\cup\{x_{n+1}\}}$ from the second list which are the same. (Note that any decomposition of the target parity into two subsets S_1 and S_2 of $k/2$ variables each will give such a pair). The set $S_1\cup S_2$ is then a consistent parity of length k. ■

7. Future Work

An obvious goal for future work is to improve our algorithmic results for learning decision lists. As a first step, one might attempt to extend the tradeoffs we achieve: is it possible to learn decision lists of length k in $n^{k^{1/2}}$ time from $\mathrm{poly}(k,\log n)$ examples?

Another goal is to extend our results for decision lists to broader concept classes. In particular, it would be interesting to obtain analogues of our algorithmic results for learning general linear threshold functions (independent of their weight). We note here that Goldmann et al. (1992) have given a linear threshold function over $\{-1,1\}^n$ for which any polynomial threshold function must have weight $2^{\Omega(n^{1/2})}$ regardless of its degree. Moreover Krause and Pudlak (1998) have shown that any Boolean function which has a polynomial threshold function over $\{0,1\}^n$ of weight w has a polynomial threshold function over $\{-1,1\}^n$ of weight n^2w^4. These results imply that *representational* results akin to Theorem 3 for general linear threshold functions must be quantitatively weaker than Theorem 3; in particular, there is a linear threshold function over $\{0,1\}^n$ with k nonzero coefficients for which any polynomial threshold function, regardless of degree, must have weight $2^{\Omega(k^{1/2})}$.

For parity functions many questions remain as well: can we learn parity functions on $k = \Theta(\log n)$ variables in polynomial time using a sublinear number of examples? Can we learn length-k parities in polynomial time using fewer than $n^{1-1/k}$ examples? Can we learn length-k parities from $O(k \log n)$ examples in time $\tilde{O}(n^{k/3})$? Progress on any of these fronts would be quite interesting.

8. Acknowledgements

We thank Les Valiant for his observation that Claim 5 can be reinterpreted in terms of polynomial threshold functions, and we thank Jean Kwon for suggesting the Chebychev polynomial. We thank Dan Spielman for allowing us to include his proof of Theorem 18.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

J. Barzdin and R. Freivald. On the prediction of general recursive functions. *Soviet Mathematics Doklady*, 13:1224–1228, 1972.

R. Beigel. Perceptrons, PP, and the Polynomial Hierarchy. *Computational Complexity*, 4:339–349, 1994.

A. Blum. Learning Boolean functions in an infinite attribute space. In *Proceedings of the 22nd Annual Symposium on Theory of Computing*, pages 64–72, 1990.

A. Blum. Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters*, 42(4):183–185, 1992.

A. Blum. On-line algorithms in machine learning. available at `http://www.cs.cmu.edu/~avrim/Papers/pubs.html`, 1996.

A. Blum. Empirical support for Winnow and weighted-majority algorithms: results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. *Journal of Computer and System Sciences*, 50:32–40, 1995.

E. Cheney. *Introduction to approximation theory*. McGraw-Hill, New York, New York, 1966.

A. Dhagat and L. Hellerstein. PAC learning with irrelevant attributes. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 64–74, 1994.

A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

A.R. Golding and D. Roth. A Winnow-based approach to spelling correction. *Machine Learning*, 34:107–130, 1999.

M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

D. Haussler. Space efficient learning algorithms. Technical Report UCSC-CRL-88-2, University of California at Santa Cruz, 1988.

D. Helmbold, R. Sloan, and M. Warmuth. Learning integer lattices. *SIAM Journal on Computing*, 21(2):240–266, 1992.

M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.

J. Kivinen, M. Warmuth, and P. Auer. The Perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97(1-2):325–343, 1997.

A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer & System Sciences*, 68(2):303–318, 2004.

A. Klivans, R. O'Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer & System Sciences*, 68(4):808–840, 2004.

M. Krause. On the computational power of Boolean decision lists. In *19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 372–383, 2002.

M. Krause and P. Pudlak. Computing Boolean functions by polynomials and threshold circuits. *Computational Complexity*, 7(4):346–370, 1998.

N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms*. PhD thesis, University of California at Santa Cruz, 1989a.

N. Littlestone. From online to batch learning. In *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 269–284, 1989b.

T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

J. Myhill and W. Kautz. On the size of weights required for linear-input switching functions. *IRE Trans. on Electronic Computers*, EC10(2):288–290, 1961.

Z. Nevo and R. El-Yaniv. On online learning of decision lists. *Journal of Machine Learning Research*, 3:271–301, 2002.

R. O'Donnell and R. Servedio. New degree bounds for polynomial threshold functions. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 325–334, 2003.

R. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

R. Servedio. Computational sample complexity and attribute-efficient learning. *Journal of Computer and System Sciences*, 60(1):161–178, 2000.

R. Servedio. Perceptron, Winnow and PAC learning. *SIAM Journal on Computing*, 31(5):1358–1369, 2002.

D. Spielman. Personal communication, 2003.

R. Uehara, K. Tsuchida, and I. Wegener. Identification of partial disjunction, parity, and threshold functions. *Theoretical Computer Science*, 230:131–147, 2000.

L. Valiant. Projection learning. *Machine Learning*, 37(2):115–130, 1999.

Journal of Machine Learning Research 7 (2006) 603–624 Submitted 10/05; Published 4/06

A Direct Method for Building Sparse Kernel Learning Algorithms

Mingrui Wu MINGRUI.WU@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Gökhan Bakır GOEKHAN.BAKIR@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editor: Nello Cristianini

Abstract

Many kernel learning algorithms, including support vector machines, result in a kernel machine, such as a kernel classifier, whose key component is a weight vector in a feature space implicitly introduced by a positive definite kernel function. This weight vector is usually obtained by solving a convex optimization problem. Based on this fact we present a direct method to build sparse kernel learning algorithms by adding one more constraint to the original convex optimization problem, such that the sparseness of the resulting kernel machine is explicitly controlled while at the same time performance is kept as high as possible. A gradient based approach is provided to solve this modified optimization problem. Applying this method to the support vectom machine results in a concrete algorithm for building sparse large margin classifiers. These classifiers essentially find a discriminating subspace that can be spanned by a small number of vectors, and in this subspace, the different classes of data are linearly well separated. Experimental results over several classification benchmarks demonstrate the effectiveness of our approach.

Keywords: sparse learning, sparse large margin classifiers, kernel learning algorithms, support vector machine, kernel Fisher discriminant

1. Introduction

Many kernel learning algorithms (KLA) have been proposed for solving different kinds of problems. For example the support vector machine (SVM) (Vapnik, 1995) have been widely used for classification and regression, the minimax probability machine (MPM) (Lanckriet et al., 2002) is another competitive algorithm for classification, the one-class SVM (Schölkopf and Smola, 2002) is a useful tool for novelty detection, while the kernel Fisher discriminant (KFD) (Mika et al., 2003) and the kernel PCA (KPCA) (Schölkopf and Smola, 2002) are powerful algorithms for feature extraction.

Many kernel learning algorithms result in a kernel machine (KM) (such as a kernel classifier), whose output can be calculated as

$$\tau(\mathbf{x}) = \sum_{i=1}^{N_{XV}} \hat{\alpha}_i K(\hat{\mathbf{x}}_i, \mathbf{x}) + b, \tag{1}$$

where $\mathbf{x} \in \mathcal{X} \subset \mathcal{R}^d$ is the input data, $\mathcal{X}$ is the input space, $\hat{\mathbf{x}}_i \in \mathcal{X}$, $1 \leq i \leq N_{XV}$, are called expansion vectors (XVs) in this paper,[1] N_{XV} is the number of XVs, $\hat{\alpha}_i \in \mathcal{R}$ is the expansion coefficient associated with $\hat{\mathbf{x}}_i$, $b \in \mathcal{R}$ is the bias and $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ is a kernel function.

Usually K is a positive definite kernel (Schölkopf and Smola, 2002), which implicitly introduces a feature space $\mathcal{F}$. Let $\phi(\cdot)$ denote the map from $\mathcal{X}$ to $\mathcal{F}$, then

$$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle, \qquad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}.$$

Hence (1) can also be written as a linear function

$$\tau(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b, \tag{2}$$

where

$$\mathbf{w} = \sum_{i=1}^{N_{XV}} \hat{\alpha}_i \phi(\hat{\mathbf{x}}_i) \tag{3}$$

is the weight vector of the KM and it equals the linear expansion of XVs in the feature space $\mathcal{F}$.

For all the KLAs mentioned above, the vector $\mathbf{w}$ is obtained by solving a *convex* optimization problem. For example, the SVM can be formulated as a quadratic programming problem (Vapnik, 1995), in (Mika et al., 2003), a convex formulation is proposed for KFD and a least squares SVM formulation is established for KPCA in (Suykens et al., 2002).

From the above description, we can see that although many KLAs are proposed for solving different kinds of problems and have various formulations, there are three widely known common points among them. First, each of them results in a KM whose key component is a weight vector $\mathbf{w}$, which can be expressed as the linear expansion of XVs in the feature space $\mathcal{F}$. Second, the vector $\mathbf{w}$ is obtained by solving a convex optimization problem. Third, the output of the resulting KM is calculated as (1) (or (2)).

When solving practical problems, we want the time for computing the output to be as short as possible. For example in real-time image recognition, in addition to good classification accuracy, high classification speed is also desirable. The time of calculating (1) (or (2)) is proportional to N_{XV}. Thus several sparse learning algorithms have been proposed to build KMs with small N_{XV}.

The reduced set (RS) method (Burges, 1996; Schölkopf and Smola, 2002) was proposed to simplify (1) by determining N_z vectors $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$ and corresponding expansion coefficients $\beta_1, \ldots, \beta_{N_z}$ such that

$$\| \mathbf{w} - \sum_{j=1}^{N_z} \beta_j \phi(\mathbf{z}_j) \|^2 \tag{4}$$

is minimized. RS methods approximate and replace $\mathbf{w}$ in (2) by $\sum_{j=1}^{N_z} \beta_j \phi(\mathbf{z}_j)$, where $N_z < N_{XV}$. The objective of RS method does not directly relate to the performance of the

1. The $\hat{\mathbf{x}}_i$, $1 \leq i \leq N_{XV}$ have different names in different kernel learning algorithms. For example, they are called support vectors in the SVM, and relevance vectors in the relevance vector machine (Tipping, 2001). In this paper we uniformly call them expansion vectors for the sake of simplicity.

KM it aims to simplify, and in order to apply RS methods, we need to build another KM in advance.

In (Lee and Mangasarian, 2001), the reduced support vector machine (RSVM) algorithm is proposed, which randomly selects N_z vectors from the training set as XVs, and then computes the expansion coefficients. This algorithm can be applied to build sparse kernel classifiers. But as the XVs are chosen randomly, and may not be good representatives of the training data, good classification performance can not be guaranteed when N_z is small (Lin and Lin, 2003).

The relevance vector machine (RVM) (Tipping, 2001) is another algorithm which leads to sparse KMs. The basic idea of the RVM is to assume a prior of the expansion coefficients which favors sparse solutions.

In this paper, based on the common points of KLAs mentioned before, we present a direct method to build sparse kernel learning algorithms (SKLA). In particular, given a KLA, we modify it by adding one more constraint to its corresponding convex optimization problem. The added constraint explicitly controls the sparseness of the resulting KM and a gradient based approach is proposed to solve the modified optimization problem. We will also show that applying this method to the SVM will result in a specific algorithm for building sparse large margin classifiers (SLMC).[2]

The remainder of this paper is organized as follows. In section 2, we describe a direct method for building SKLAs. After this, we will focus on a particular application of this method to the SVM algorithm, leading to a detailed algorithm for building SLMC. The SLMC algorithm is presented in section 3, where we will also point out that it actually finds a discriminating subspace of the feature space $\mathcal{F}$. Some comparisons with related approaches are given in section 4. Experimental results are provided in section 5 and we conclude the paper in the last section.[3]

2. A Direct Method of Building SKLAs

In this section, we propose a direct method for building SKLAs.

2.1 Basic Idea

As mentioned before, many KLAs can be formulated as an optimization problem, which can be written in a general form as follows:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad f(\mathbf{w},b,\boldsymbol{\xi}), \tag{5}$$

$$\text{subject to} \quad g_i(\mathbf{w},b,\boldsymbol{\xi}) \le 0, \quad 1 \le i \le N_g, \tag{6}$$

$$h_j(\mathbf{w},b,\boldsymbol{\xi}) = 0, \quad 1 \le j \le N_h, \tag{7}$$

where $f(\mathbf{w},b,\boldsymbol{\xi})$ is the objective function to be minimized, $\mathbf{w} \in \mathcal{F}$ and $b \in \mathcal{R}$ are respectively the weight vector and the bias of the KM to be built, $\boldsymbol{\xi} = [\xi_1,\ldots,\xi_{N_\xi}]^\top \in \mathcal{R}^{N_\xi}$ is a vector of some auxiliary variables (such as the slack variables in the soft margin training problem),

2. Here we don't use the phrase "sparse SVM" because the XVs of the resulting classifier are not necessarily support vectors, i.e. they may not lie near the classification boundary.
3. This paper is an extension of our previous work (Wu et al., 2005).

N_ξ is the number of auxiliary variables, N_g is the number of inequality constraints specified by $g_i(\mathbf{w}, b, \boldsymbol{\xi})$, while N_h is the number of equality constraints specified by $h_j(\mathbf{w}, b, \boldsymbol{\xi})$.

Our objective is as follows: given a KLA and a positive integer N_z, we want to modify the given KLA such that the number of XVs of the resulting KM equals N_z while at the same time the performance of the KM should be kept as well as possible. To achieve this, we propose to solve the following problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\beta},\mathbf{Z}} \quad & f(\mathbf{w}, b, \boldsymbol{\xi}), && (8)\\ \text{subject to} \quad & g_i(\mathbf{w}, b, \boldsymbol{\xi}) \leq 0, \quad 1 \leq i \leq N_g, && (9)\\ & h_j(\mathbf{w}, b, \boldsymbol{\xi}) = 0, \quad 1 \leq j \leq N_h, && (10)\\ & \mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i, && (11) \end{aligned}$$

where (8)–(10) are exactly the same as (5)–(7), while $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}] \in \mathcal{R}^{d \times N_z}$ is the matrix of XVs and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{N_z}]^\top \in \mathcal{R}^{N_z}$ is the vector of expansion coefficients.

It can be seen that the above problem is the problem (5)–(7) with one added constraint (11) saying that the weight vector of the resulting KM equals the expansion of the $\phi(\mathbf{z}_i)$, $1 \leq i \leq N_z$. Note that the $\mathbf{z}_i$ are also variables, so they need to be computed when solving the optimization problem.

Due to the constraint (11), solving the problem (8)–(11) will naturally lead to a sparse KM. Further more, since the objective function of the problem (8)–(11) is exactly the same as the original problem (5)–(7), so in principle the performance of the resulting KM can be kept as well as possible.

Because of the non-convex constraint (11), it is difficult to obtain the global optimum of the above problem, thus we propose a gradient based approach. However, our gradient based minimization will be performed *only* on the expansion vectors $\mathbf{Z}$ but not on all the variables. To this end, we define the following the *marginal function* $W(\mathbf{Z})$ which is obtained by keeping the expansion vectors in problem (8)–(11) fixed, i.e. :

$$\begin{aligned} W(\mathbf{Z}) := \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\beta}} \quad & f(\mathbf{w}, b, \boldsymbol{\xi}), && (12)\\ \text{subject to} \quad & g_i(\mathbf{w}, b, \boldsymbol{\xi}) \leq 0, \quad 1 \leq i \leq N_g, && (13)\\ & h_j(\mathbf{w}, b, \boldsymbol{\xi}) = 0, \quad 1 \leq j \leq N_h, && (14)\\ \text{and} \quad & \mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i. && (15) \end{aligned}$$

The above problem is the same as problem (8)–(11) except that $\mathbf{Z}$ is not variable but fixed.

Clearly any global (or local) minimum of $W(\mathbf{Z})$ is also a global (or local) minimum of problem (8)–(11), which means that the minima of the original problem (8)–(11) can be found by computing the minima of function $W(\mathbf{Z})$. Here we propose to minimize $W(\mathbf{Z})$ by the gradient based algorithm. To this end, at any given $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, we need to calculate both the function value $W(\mathbf{Z})$ and the gradient $\nabla W(\mathbf{Z})$. These two problems are discussed in the next subsection.

2.2 Computing $W(\mathbf{Z})$ and its Gradient $\nabla W(\mathbf{Z})$

To compute the function value of $W(\mathbf{Z})$ at any given $\mathbf{Z}$, we need to solve the optimization problem (12)–(15). Obviously this is a problem dependent task. However as mentioned before, the original optimization problem (5)–(7) of many current KLAs are convex, and (15) is just a linear constraint once $\mathbf{Z}$ is fixed. Therefore the problem (12)–(15) is still convex, which means its global optimum, and thus the function value of $W(\mathbf{Z})$, can be readily computed.

Next we turn to consider how to compute $\nabla W(\mathbf{Z})$, which requires more carefulness since in general $W(\mathbf{Z})$ is not necessarily differentiable. According to the constraint (11), the weight vector $\mathbf{w}$ can be completely determined by $\boldsymbol{\beta}$ and $\mathbf{Z}$, so the functions f, g_i and h_j can be regarded as functions of b, $\boldsymbol{\xi}$, $\boldsymbol{\beta}$ and $\mathbf{Z}$. Without causing confusions, we write them as $f(\mathbf{x}, \mathbf{Z})$, $g_i(\mathbf{x}, \mathbf{Z})$ and $h_j(\mathbf{x}, \mathbf{Z})$ in the following, where $\mathbf{x} := [b, \boldsymbol{\xi}^\top, \boldsymbol{\beta}^\top]^\top$.

Substituting (15) into (12)–(14), we have

$$\min_{\mathbf{x}} \quad f(\mathbf{x}, \mathbf{Z}), \tag{16}$$

$$\text{subject to} \quad g_i(\mathbf{x}, \mathbf{Z}) \le 0, \quad 1 \le i \le N_g, \tag{17}$$

$$h_j(\mathbf{x}, \mathbf{Z}) = 0, \quad 1 \le j \le N_h, \tag{18}$$

and $W(\mathbf{Z})$ is the optimal value of the above optimization problem.

To compute $\nabla W(\mathbf{Z})$, we can apply the following lemma which gives both the conditions when $W(\mathbf{Z})$ is differentiable and an explicit form of the derivative:

Lemma 1 *(Gauvin and Dubeau, 1982): Assume that all the functions f, g_i and h_j in problem (16)–(18) are continuously differentiable and suppose that problem (16)–(18) has a unique optimal solution $\bar{\mathbf{x}}$ at $\mathbf{Z} = \bar{\mathbf{Z}}$. Furthermore let $\bar{\alpha}_i$ and $\bar{\beta}_j$ be the unique corresponding Lagrange multipliers associated with g_i and h_j respectively, $1 \le i \le N_g$, $1 \le j \le N_h$. Assume further that the feasible set of problem (16)–(18) is uniformly compact at $\bar{\mathbf{Z}}$ and that, the optimal solution $\bar{\mathbf{x}}$ is Mangasarian-Fromovitz regular, then the gradient of $W(\mathbf{Z})$ exists at $\bar{\mathbf{Z}}$ and equals*

$$\nabla W(\mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}} = \nabla_{\mathbf{Z}} f(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}} + \sum_{i=1}^{N_g} \bar{\alpha}_i \, \nabla_{\mathbf{Z}} g_i(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}} + \sum_{j=1}^{N_h} \bar{\beta}_j \, \nabla_{\mathbf{Z}} h_j(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}}, \tag{19}$$

where $\nabla_{\mathbf{Z}} f(\bar{\mathbf{x}}, \mathbf{Z})|_{\mathbf{Z}=\bar{\mathbf{Z}}}$ denotes the gradient of $f(\mathbf{x}, \mathbf{Z})$ with respect to $\mathbf{Z}$ at $\mathbf{Z} = \bar{\mathbf{Z}}$ while fixing $\mathbf{x}$ at $\bar{\mathbf{x}}$.

As can be seen that in addition to the uniqueness of the optimal solution and its corresponding Lagrange multipliers, lemma 1 also requires the feasible set to be uniformly compact and the optimal solution to be Mangasarian-Fromovitz regular. The formal definitions of the last two conditions are given in appendix B. Since the properties of the feasible set and the optimal solutions depend on the specific KLA, we will discuss all these conditions for some particular KLAs in subsection 3.3 and appendix A. We will see that the conditions of lemma 1 are very mild for many current KLAs.

3. Building an SLMC

Having described our direct method for building SKLAs, we will focus on a concrete application of this method in the rest of this paper. In particular, we will apply this method to SVMs in order to obtain an algorithm for building SLMCs. We will also analyze its properties and compare it with other related approaches. In this section, we will present the SLMC algorithm by closely following the discussions of section 2.

3.1 Objective

Now we begin to consider the binary classification problem, where we are given a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{X}$ is the input data, and $y_i \in \{-1, 1\}$ is the class label.

Our objective is as follows: given a training data set and an positive integer N_z, we want to build a classifier such that the number of XVs of the classifier equals N_z and the margin of the classifier is as large as possible. This way we build a large margin classifier whose sparseness is explicitly controlled.

Based on the direct method described in the last section, we need to solve the problem (8)–(11) to achieve this goal. For the moment, (8)–(10) become the SVM training problem and the constraint (11) controlls the sparseness of the resulting classifier. So we should solve the following problem

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi},b,\boldsymbol{\beta},\mathbf{Z}} \quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i, && (20)\\
\text{subject to} \quad & y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, && (21)\\
& \xi_i \geq 0, \quad \forall i, && (22)\\
& \mathbf{w} = \sum_{i=1}^{N_z}\phi(\mathbf{z}_i)\beta_i, && (23)
\end{aligned}$$

where C is a positive constant, $\mathbf{w} \in \mathcal{F}$ is the weight vector of the decision hyperplane in feature space, $b \in \mathcal{R}$ is the bias of the classifier, $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_N]^\top \in \mathcal{R}^N$ is the vector of slack variables, $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}] \in \mathcal{R}^{d\times N_z}$ is the matrix of XVs and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{N_z}]^\top \in \mathcal{R}^{N_z}$ is the vector of expansion coefficients.

Following our proposed method, to solve the above problem, we turn to minimize the marginal function $W(\mathbf{Z})$ defined in problem (12)–(15). For the current problem, the value of $W(\mathbf{Z})$ is the minimum of the following optimization problem,

$$\begin{aligned}
\min_{\mathbf{w},\boldsymbol{\xi},b,\boldsymbol{\beta}} \quad & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{N}\xi_i, && (24)\\
\text{subject to} \quad & y_i(\mathbf{w}^\top\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, && (25)\\
& \xi_i \geq 0, \quad \forall i, && (26)\\
& \mathbf{w} = \sum_{i=1}^{N_z}\phi(\mathbf{z}_i)\beta_i. && (27)
\end{aligned}$$

The above problem is the same as problem (20)–(23) except that $\mathbf{Z}$ is not variable but fixed.

Following the discussion in section 2.1, the (local) minimum of the original problem (20)–(23) can be found by computing the (local) minimum of function $W(\mathbf{Z})$ and we will use the gradient based algorithm to do this. In the following two subsections we will discuss how to compute the function value $W(\mathbf{Z})$ and the gradient $\nabla W(\mathbf{Z})$ respectively.

3.2 Computing $W(\mathbf{Z})$ and $\boldsymbol{\beta}$

To compute the function value of $W(\mathbf{Z})$ at any given $\mathbf{Z}$, we need to solve the convex optimization problem (24)–(27), which is actually a problem of building an SVM with given XVs $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$. This problem has already been considered in the RSVM algorithm (Lee and Mangasarian, 2001; Lin and Lin, 2003). But in the RSVM algorithm, only an approximation of the problem (24)–(27) is solved. Here we will propose a different method which exactly solves this problem. (See section 4.2 for a discussion and section 5.5 for a comparison of the experimental results of these two methods.)

Substituting (27) into (24) and (25), we have

$$\begin{aligned} \min_{\boldsymbol{\xi},b,\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta} + C\sum_{i=1}^{N} \xi_i, && (28) \\ \text{subject to} \quad & y_i(\boldsymbol{\beta}^\top \psi_z(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i, && (29) \\ & \xi_i \geq 0, \quad \forall i, && (30) \end{aligned}$$

where

$$\psi_z(\mathbf{x}_i) = [K(\mathbf{z}_1, \mathbf{x}_i), \ldots, K(\mathbf{z}_{N_z}, \mathbf{x}_i)]^\top \tag{31}$$

is the empirical kernel map (Schölkopf and Smola, 2002) and $\mathbf{K}^z$ is the kernel matrix of $\mathbf{z}_i$, i.e. $\mathbf{K}^z_{ij} = K(\mathbf{z}_i, \mathbf{z}_j)$.

Note that when $N_z = N$ and $\mathbf{z}_i = \mathbf{x}_i$, $1 \leq i \leq N$, this is the standard SVM training problem. In contrast, the problem (28)–(30) is to train a linear SVM in a subspace spanned by $\phi(\mathbf{z}_i)$, $1 \leq i \leq N_z$, where $\mathbf{z}_i$ are are not necessarily training examples.

Now we investigate its dual problem. To derive it, we introduce the Lagrangian,

$$\begin{aligned} & L(\boldsymbol{\xi}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) && (32) \\ = \; & \frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta} + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \gamma_i \xi_i \\ & - \sum_{i=1}^{N} \alpha_i [y_i(\boldsymbol{\beta}^\top \psi_z(\mathbf{x}_i) + b) - 1 + \xi_i], \end{aligned}$$

with Lagrange multipliers $\gamma_i \geq 0$ and $\alpha_i \geq 0$.

The derivatives of $L(\boldsymbol{\xi}, b, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\gamma})$ with respect to the primal variables must vanish,

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \mathbf{K}^z \boldsymbol{\beta} - \sum_{i=1}^{N} \alpha_i y_i \psi_z(\mathbf{x}_i) = 0, \tag{33}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0, \quad \forall i, \tag{34}$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0, \quad \forall i. \tag{35}$$

Equation (33) leads to

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1} \sum_{i=1}^{N} \alpha_i y_i \psi_z(\mathbf{x}_i). \tag{36}$$

Substituting (33)–(35) into (32) and using (36), we arrive at the dual form of the optimization problem:

$$\max_{\boldsymbol{\alpha} \in \mathcal{R}^N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j), \tag{37}$$

$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0, \tag{38}$$

$$\text{and} \quad 0 \leq \alpha_i \leq C, \quad \forall i, \tag{39}$$

where

$$\hat{K}_z(\mathbf{x}_i, \mathbf{x}_j) = \psi_z(\mathbf{x}_i)^\top (\mathbf{K}^z)^{-1} \psi_z(\mathbf{x}_j). \tag{40}$$

The function $\hat{K}_z(\cdot, \cdot)$ defined by (40) is a positive definite kernel function (Schölkopf and Smola, 2002). To see this, consider the following map,[4]

$$\phi_z(\mathbf{x}_i) = \mathbf{T} \psi_z(\mathbf{x}_i), \tag{41}$$

where $\psi_z(\cdot)$ is defined by (31) and

$$\mathbf{T} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \mathbf{V}^\top, \tag{42}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues of matrix $\mathbf{K}^z$ and $\mathbf{V}$ is a matrix whose columns are eigenvectors of $\mathbf{K}^z$. So

$$\mathbf{T}^\top \mathbf{T} = \mathbf{V} \boldsymbol{\Lambda}^{-1} \mathbf{V} = (\mathbf{K}^z)^{-1}. \tag{43}$$

Combining equation (41)and (43) we have

$$\langle \phi_z(\mathbf{x}_i), \phi_z(\mathbf{x}_j) \rangle = \psi_z(\mathbf{x}_i)^\top (\mathbf{K}^z)^{-1} \psi_z(\mathbf{x}_j) = \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j).$$

It can be seen that problem (37)–(39) has the same form as the dual of an SVM training problem. Therefore *given* $\mathbf{Z}$*, computing the expansion coefficients of SVM with kernel*

4. The map defined in (41) is called the "whitened" empirical kernel map or "kernel PCA map"(Schölkopf and Smola, 2002).

function K is equivalent to training an SVM with a modified kernel function $\hat{K}_z$ defined by (40).

Since problem (37)–(39) is the dual of problem (24)–(27), the optima of these two problems are equal to each other. So given $\mathbf{Z}$, assuming $\alpha_i^z, 1 \le i \le N$ are the solution of (37)–(39), then we can compute $W(\mathbf{Z})$ as

$$W(\mathbf{Z}) = \sum_{i=1}^{N} \alpha_i^z - \frac{1}{2} \sum_{i}^{N} \sum_{j=1}^{N} \alpha_i^z \alpha_j^z y_i y_j \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j). \tag{44}$$

According to (36), the expansion coefficients $\boldsymbol{\beta}$ can be calculated as

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1} \sum_{i=1}^{N} \alpha_i^z y_i \psi_z(\mathbf{x}_i) = (\mathbf{K}^z)^{-1} (\mathbf{K}^{zx}) \mathbf{Y} \boldsymbol{\alpha}^z, \tag{45}$$

where $\psi_z(\cdot)$ is defined by (31), $\mathbf{K}^{zx}$ is the matrix defined by $\mathbf{K}_{ij}^{zx} = K(\mathbf{z}_i, \mathbf{x}_j)$, $\mathbf{Y}$ is a diagonal matrix of class labels, i.e. $\mathbf{Y}_{ii} = y_i$, and $\boldsymbol{\alpha}^z = [\alpha_1^z, \ldots, \alpha_N^z]^\top$.

3.3 Computing $\nabla W(\mathbf{Z})$ of SLMC

To compute $\nabla W(\mathbf{Z})$, we can apply lemma 1 to the soft margin SVM training problem (37)–(39) and yield the following result.

Corollary 2 *In the soft margin SVM training problem (37)–(39), assume that the kernel function $\hat{K}_z(\cdot, \cdot)$ is strictly positive definite and the resulting support vectors come from both positive and negative classes,*[5] *then the derivatives of $W(\mathbf{Z})$ with respect to $\mathbf{z}_{uv}$, which denotes the v-th component of vector $\mathbf{z}_u$, $1 \le u \le N_z, 1 \le v \le d$, exists and can be computed as follows:*

$$\frac{\partial W}{\partial \mathbf{z}_{uv}} = -\frac{1}{2} \sum_{i,j=1}^{N} \alpha_i^z \alpha_j^z y_i y_j \frac{\partial \hat{K}_z(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{z}_{uv}}, \tag{46}$$

where α_i^z, $1 \le i \le N$ denote the solution of problem (37)–(39). In other words, $\nabla W(\mathbf{Z})$ can be computed as if α^z did not depend on $\mathbf{Z}$.[6]

The proof of corollary 2 is given in appendix C.

As can be seen in corollary 2, to apply lemma 1 to calculate $\nabla W(\mathbf{Z})$, we only need to make two assumptions on problem (37)–(39): The kernel function $\hat{K}_z(\cdot, \cdot)$ is strictly positive definite and the resulting support vectors come from both classes. Certainly these are *not* strict assumptions for most practical applications. Similarly one can verify that lemma 1 can also be applied to many other KLAs with mild assumptions, such as the one-class SVM.

5. Let α_i^z, $1 \le i \le N$ denote the optimal solution of problem (37)–(39), support vectors are those input data $\mathbf{x}_i$ whose corresponding α_i^z are larger than 0 (Vapnik, 1995).
6. In corollary 2, α_i^z is bounded above by C as shown in (39). A similar conclusion is proposed in (Chapelle et al., 2002) for the hard margin SVM training problem, where there is no upper bound on α_i^z. This implies that the feasible set is not compact, hence lemma 1 can not be applied any more. Actually in (Chapelle et al., 2002), only the uniqueness of the optimal solution is emphasized, which, to our knowledge, is not enough to guarantee the differentiability of the marginal function $W(\mathbf{Z})$.

And we will show another example in appendix A on applying our direct method to sparsify the KFD algorithm (Mika et al., 2003).

According to (40),

$$\begin{aligned}\frac{\partial \hat{K}_z(\mathbf{x}_i,\mathbf{x}_j)}{\partial \mathbf{z}_{uv}} &= (\frac{\partial \psi_z(\mathbf{x}_i)}{\partial \mathbf{z}_{uv}})^\top (\mathbf{K}^z)^{-1}\psi_z(\mathbf{x}_j) + \psi_z(\mathbf{x}_i)^\top (\mathbf{K}^z)^{-1}\frac{\partial \psi_z(\mathbf{x}_j)}{\partial \mathbf{z}_{uv}} \\ &+ \psi_z(\mathbf{x}_i)^\top \frac{\partial (\mathbf{K}^z)^{-1}}{\partial \mathbf{z}_{uv}}\psi_z(\mathbf{x}_j),\end{aligned}$$

where $\frac{\partial (\mathbf{K}^z)^{-1}}{\partial \mathbf{z}_{uv}}$ can be calculated as

$$\frac{\partial (\mathbf{K}^z)^{-1}}{\partial \mathbf{z}_{uv}} = -(\mathbf{K}^z)^{-1}\frac{\partial \mathbf{K}^z}{\partial \mathbf{z}_{uv}}(\mathbf{K}^z)^{-1}.$$

So at any given $\mathbf{Z}$, $W(\mathbf{Z})$ and $\nabla W(\mathbf{Z})$ can be computed as (44) and (46) respectively. In our implementation, we use the LBFGS algorithm (Liu and Nocedal, 1989) to minimize $W(\mathbf{Z})$, which is an efficient gradient based optimization algorithm.

3.4 The Kernel Function $\hat{K}_z$ and Its Corresponding Feature Space $\mathcal{F}_z$

The kernel function $\hat{K}_z$ plays an important role in our approach. In this section, some analysis of $\hat{K}_z$ is provided, which will give us insights into how to build an SLMC.

It is well known that training an SVM with a nonlinear kernel function K in the input space $\mathcal{X}$ is equivalent to building a linear SVM in a feature space $\mathcal{F}$. The map $\phi(\cdot)$ from $\mathcal{X}$ to $\mathcal{F}$ is implicitly introduced by K. In section 2.2, we derived that for a given set of XVs $\mathbf{z}_1,\ldots,\mathbf{z}_{N_z}$, training an SVM with kernel function K is equivalent to building an SVM with another kernel function $\hat{K}_z$, which is in turn equivalent to constructing a linear SVM in another feature space. Let $\mathcal{F}_z$ denote this feature space, then the map from the $\mathcal{X}$ to $\mathcal{F}_z$ is $\phi_z(\cdot)$, which is explicitly defined by (41).

According to (41), $\phi_z(\mathbf{x}) = \mathbf{T}\psi_z(\mathbf{x})$. To investigate the role of the matrix $\mathbf{T}$, consider $\mathbf{U}^z$ defined by

$$\mathbf{U}^z = [\phi(\mathbf{z}_1),\ldots,\phi(\mathbf{z}_{N_z})]\mathbf{T}^\top.$$

Then

$$(\mathbf{U}^z)^\top\mathbf{U}^z = \mathbf{T}\mathbf{K}^z\mathbf{T}^\top = \mathbf{I},$$

where $\mathbf{I}$ is the unit matrix, which means that $\mathbf{T}^\top$ orthonormalizes $\phi(\mathbf{z}_i)$ in the feature space $\mathcal{F}$. Thus the columns of $\mathbf{U}^z$ can be regarded as an orthonormal basis of a subspace of $\mathcal{F}$. For any $\mathbf{x}\in\mathcal{X}$, if we calculate the projection of $\phi(\mathbf{x})$ into this subspace, we have

$$\begin{aligned}(\mathbf{U}^z)^\top\phi(\mathbf{x}) &= \mathbf{T}[\phi(\mathbf{z}_1),\ldots,\phi(\mathbf{z}_{N_z})]^\top\phi(\mathbf{x}) \\ &= \mathbf{T}[K(\mathbf{z}_1,\mathbf{x}),\ldots,K(\mathbf{z}_{N_z},\mathbf{x})]^\top \\ &= \mathbf{T}\psi_z(\mathbf{x}) = \phi_z(\mathbf{x}).\end{aligned}$$

This shows that the subspace spanned by the columns of $\mathbf{U}^z$ is identical to $\mathcal{F}_z$. As $\mathbf{U}^z$ are obtained by orthonormalizing $\phi(\mathbf{z}_i)$, $\mathcal{F}_z$ is a subspace of $\mathcal{F}$ and it is spanned by $\phi(\mathbf{z}_i)$, $1\le i\le N_z$.

Now that for a given set of XVs $\mathbf{z}_i$, building an SVM with a kernel function K is equivalent to building a linear SVM in $\mathcal{F}_z$, in order to get good classification performance, we have to find a discriminating subspace $\mathcal{F}_z$ where two classes of data are linearly well separated. Based on this point of view, we can see that our proposed approach essentially finds a subspace $\mathcal{F}_z$ where the margin of the training data is maximized.

4. Comparison with Related Approaches

In this section we compare the SLMC algorithm with related approaches.

4.1 Modified RS Method

In the second step of the RS method, after the XVs $\mathbf{z}_1, \ldots, \mathbf{z}_{N_z}$ are obtained, the expansion coefficients $\boldsymbol{\beta}$ are computed by minimizing (4), which leads to (Schölkopf and Smola, 2002)

$$\boldsymbol{\beta} = (\mathbf{K}^z)^{-1}(\mathbf{K}^{zx})\mathbf{Y}\boldsymbol{\alpha}, \tag{47}$$

where $\mathbf{K}^{zx}$ and $\mathbf{Y}$ are defined as in (45), and $\boldsymbol{\alpha}$ is the solution of building an SVM with kernel function K on the training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$.

We propose to modify the second step of RS method as (45). Clearly (47) and (45) are of the same form. The only difference is that in (47), $\boldsymbol{\alpha}$ is the solution of training an SVM with kernel function K, while in (45), $\boldsymbol{\alpha}^z$ is the solution of training an SVM with the kernel function $\hat{K}_z$, which takes the XVs $\mathbf{z}_i$ into consideration. As $\boldsymbol{\beta}$ calculated by (45) maximizes the margin of the resulting classifier, we can expect a better classification performance of this modified RS method. We will see this in the experimental results.

4.2 Comparison with RSVM and a Modified RSVM Algorithm

One might argue that our approach appears to be similar to the RSVM, because the RSVM algorithm also restricts the weight vector of the decision hyperplane to be a linear expansion of N_z XVs.

However there are two important differences between the RSVM and our approach. The first one (and probably the fundamental one) is that in the RSVM approach, N_z XVs are randomly selected from the training data in advance, but are not computed by finding a discriminating subspace $\mathcal{F}_z$. The second difference lies in the method for computing the expansion coefficients $\boldsymbol{\beta}$. Our method exactly solves the problem (28)–(30) without any simplifications. But in the RSVM approach, certain simplifications are performed, among which the most significant one is changing the first term in the objective function (28) from $\frac{1}{2}\boldsymbol{\beta}^\top \mathbf{K}^z \boldsymbol{\beta}$ to $\frac{1}{2}\boldsymbol{\beta}^\top \boldsymbol{\beta}$. This step immediately reduces the problem (28)–(30) to a standard linear SVM training problem (Lin and Lin, 2003), where $\boldsymbol{\beta}$ becomes the weight vector of the decision hyperplane and the training set becomes $\{\psi_z(\mathbf{x}_i), y_i\}_{i=1}^N$.

On the other hand, our method of computing $\boldsymbol{\beta}$ is to build a linear SVM in the subspace $\mathcal{F}_z$, which is to train a linear SVM for the training data set $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$.

Now let us compare the two training sets mentioned above, i.e. $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$ and $\{\psi_z(\mathbf{x}_i), y_i\}_{i=1}^N$. As derived in section 3.4, $\phi_z(\mathbf{x}_i)$ are calculated by projecting $\phi(\mathbf{x}_i)$ onto a set of vectors, which is obtained by orthonormalizing $\phi(\mathbf{z}_j)$ $(1 \le j \le N_z)$, while $\psi_z(\mathbf{x}_i)$ is

calculated by computing the dot production between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{z}_j)$ $(1 \leq j \leq N_z)$ directly, without the step of orthonormalization.

Analogous to the modified RS method, we propose a modified RSVM algorithm: Firstly, N_z training data are randomly selected as XVs, then the expansion coefficients $\boldsymbol{\beta}$ are computed by (45).

4.3 Comparison with the RVM

The RVM (Tipping, 2001) algorithm and many other sparse learning algorithms, such as sparse greedy algorithms (Nair et al., 2002), or SVMs with l_1-norm regularization (Bennett, 1999), result in a classifier whose XVs are a subset of the training data. In contrast, the XVs of SLMC do not necessarily belong to the training set. This means that SLMC can in principle locate better discriminating XVs. Consequently, with the same number of XVs, SLMC can have better classification performance than the RVM and other sparse learning algorithms which select the XVS only from the training data. This can be seen from the experimental results provided in section 5.5.

4.4 SLMC vs Neural Networks

Since the XVs of the SLMC do not necessarily belong to the training set and training an SLMC is a gradient based process,[7] the SLMC can be thought of as a neural network with weight regularization (Bishop, 1995). However, there are clear differences between the SLMC algorithm and a feed forward neural network. First, analogous to an SVM, the SLMC considers the geometric concept of margin, and aims to maximizes it. To this end, the regularizer takes into account the kernel matrix $\mathbf{K}^z$. Second, SLMC minimizes the "hinge-loss", which is different from the loss functions adopted by neural networks.[8] Therefore, both the regularizer and the loss function of SLMC are different from those of traditional perceptrons.

Furthermore, the SLMC algorithm is just an application of our 'direct sparse' method. It is straightforward to apply this method to build sparse one-class SVM algorithm (Schölkopf and Smola, 2002), to which there is no obvious neural network counterpart.

On the other hand, analogous to neural networks, we also have an additional regularization via the number N_z determining the number of XVs, which is an advantage in some practical applications where runtime constraints exist and the maximum prediction time is known a priori. Note that the prediction time (the number of kernel evaluations) of a soft margin SVM scales linearly with the number of training patterns (Steinwart, 2003).

5. Experimental Results

Now we conduct some experiments to investigate the performance of the SLMC algorithm and compare it with other related approaches.

7. This is also similar to the recent work of (Snelson and Ghahramani, 2006) on building sparse Gaussian processes, which was done at almost the same time with our previous work (Wu et al., 2005).
8. Note that the shape of the hinge-loss is similar to that of the loss function adopted in logistic regression, where the logarithm of the logistic sigmoid function (Bishop, 1995) is involved. Here the logistic sigmoid function refers to $y(x) = \frac{1}{1+\mathrm{e}^{-x}}$. So the shape of the hing-loss is different from that of the loss function used by the perceptron.

5.1 Approaches to be Compared

The following approaches are compared in the experiments: Standard SVM, RS method, modified RS method (MRS, cf. section 4.1), RSVM, modified RSVM (MRSVM, cf. section 4.2), relevance vector machine (RVM), and the proposed SLMC approach.

Note that in our experiments, RS and MRS use exactly the same XVs, but they compute the expansion coefficients by (47) and (45) respectively. Similarly RSVM and MRSVM also use the same set of XVs, the difference lies in the method for computing the expansion coefficients.

5.2 Data Sets

Seven classification benchmarks are considered: USPS, Banana, Breast Cancer, Titanic, Waveform, German and Image. The last six data sets are provided by Gunnar Rätsch and can be downloaded from http://ida.first.fraunhofer.de/projects/bench. For the USPS data set, 7291 examples are used for training and the remaining 2007 are for testing. For each of the last six data sets, there are 100 training/test splits and we follow the same scheme as (Tipping, 2001): our results show averages over the first 10 of those.

5.3 Parameter Selection

A Gaussian kernel is used in the experiments:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \parallel \mathbf{x} - \mathbf{x}' \parallel^2). \tag{48}$$

The parameters for different approaches are as follows:

Standard SVM: For the USPS data set, we use the same parameters as in (Schölkopf and Smola, 2002): $C = 10$ and $\gamma = 1/128$. For the other data sets, we use the parameters provided by Gunnar Rätsch, which are shown on the same website where these data sets are downloaded.[9]

RSVM and MRSVM: We perform 5-fold cross validation on the training set to select parameters for the RSVM. MRSVM uses the same parameters as the RSVM.

RS method: The RS method uses the same kernel parameter as the standard SVM, since it aims to simplify the standard SVM solution.

SLMC and MRS: In our experiments, they use exactly the same parameters as the standard SVM on all the data sets.

RVM: The results for the RVM are taken directly from (Tipping, 2001), where 5-fold cross validation was performed for parameter selection.

5.4 Experimental Settings

For each data set, first a standard SVM is trained with the LIBSVM software.[10] (For the USPS, ten SVMs are built, each trained to separate one digit from all others). Then the other approaches are applied. The ratio N_z/N_{SV} varies from 5% to 10%.

For the RSVM, we use the implementation contained in the LIBSVM Tools.[11]

9. See http://ida.first.fraunhofer.de/projects/bench
10. From http://www.csie.ntu.edu.tw/~cjlin/libsvm
11. From http://www.csie.ntu.edu.tw/~cjlin/libsvmtools

For the RS method, there is still no standard or widely accepted implementation, so we try three different ones: a program written by ourselves, the code contained in the machine learning toolbox SPIDER,[12] and the code contained in the statistical pattern recognition toolbox STPRTOOL.[13] For each data set, we apply these three implementations and select the best one corresponding to the minimal value of the objective function (4).

5.5 Numerical Results

Experimental results are shown in Table 1, where the initial XVs of the SLMC are randomly selected from the training data. In Table 1, N_{SV} stands for the number of support vectors (SVs) of the standard SVM, N_z represents the number of XVs of other sparse learning algorithms.

Data Set		USPS	Banana	Breast Cancer	Titanic	Waveform	German	Image
SVM	N_{SV}	2683	86.7	112.8	70.6	158.9	408.2	172.1
	Error(%)	4.3	11.8	28.6	22.1	9.9	22.5	2.8
N_z/N_{SV} = 5%	RS	**4.9**	39.4	28.8	37.4	**9.9**	22.9	37.6
	MRS	**4.9**	27.6	28.8	**23.9**	10.0	22.5	19.4
	RSVM	11.6	29.9	29.5	24.5	15.1	23.6	23.6
	MRSVM	11.5	28.1	29.4	24.8	14.7	23.9	20.7
	SLMC	**4.9**	**16.5**	**27.9**	26.4	**9.9**	**22.3**	**5.2**
N_z/N_{SV} = 10%	RS	**4.7**	21.9	**27.9**	26.6	10.0	22.9	18.3
	MRS	4.8	17.5	29.0	22.6	**9.9**	**22.6**	6.9
	RSVM	8.2	17.5	31.0	22.9	11.6	24.5	14.2
	MRSVM	8.0	16.9	30.3	23.9	11.8	23.7	12.7
	SLMC	**4.7**	**11.0**	**27.9**	**22.4**	**9.9**	22.9	**3.6**
RVM	N_z/N_{SV}(%)	11.8	13.2	5.6	92.5	9.2	3.1	20.1
	Error(%)	5.1	**10.8**	29.9	23.0	10.9	**22.2**	3.9

Table 1: Results on seven classification benchmarks. The test error rates of each algorithm are presented. The N_{SV} for the last six data sets are the averages over 10 training/test splits. The best result in each group is shown in boldface. The number of XVs of the RVM is not chosen a priori, but comes out as a result of training. So for the RVM, the ratio N_z/N_{SV} is given in order to compare it with other algorithms. For each data set, the result of the RVM is shown in boldface if it is the best compared to the other sparse learning algorithms.

From Table 1, it can be seen the classification accuracy of SLMC is comparable with the full SVM when $N_z/N_{SV} = 0.1$.

Table 1 also illustrates that SLMC outperforms the other sparse learning algorithms in most cases. Also the SLMC usually improves the classification results of the RS method. In some cases the improvement is large such as on Banana and Image data sets.

When comparing MRS with RS, and MRSVM with RSVM, the results in Table 1 demonstrate that in most cases MRS beats RS, and similarly, MRSVM usually outperforms RSVM

12. From http://www.kyb.mpg.de/bs/people/spider
13. From http://cmp.felk.cvut.cz/~xfrancv/stprtool

a little. This means that for a given set of XVs, computing the expansion coefficients according to (45) is a good choice.

5.6 Some Implementation Details

In table 1, we report the results obtained by random initialization. The K-means algorithm has also been tried to choose the initial XVs and resulting classification results are similar. To illustrate this quantitatively, in table 2, we present the results obtained by using the K-means algorithm for initialization.

Data Set		USPS	Banana	Breast Cancer	Titanic	Waveform	German	Image
N_z/N_{SV} = 5%	random	4.9	16.5	27.9	26.4	9.9	22.3	5.2
	k-means	4.9	16.2	26.8	24.4	9.9	22.7	6.0
N_z/N_{SV} = 10%	random	4.7	11.0	27.9	22.4	9.9	22.9	3.6
	k-means	4.6	10.9	27.3	23.2	9.9	22.7	3.8

Table 2: Results of the SLMC algorithm, obtained by random initialization and k-means initialization.

In our proposed approach, we need to compute the inverse of $\mathbf{K}^z$ (see for example (45)). Theoretically if the Gaussian kernel is used and $\mathbf{z}_i$, $1 \leq i \leq N_z$, are different from each other, then $\mathbf{K}^z$ should be full rank, whose inverse can be computed without any problems. However in experiments, we do observe the cases where $\mathbf{K}^z$ was ill conditioned. But we find out the reason for this is that there are some duplicated data points in the data sets, which are accidentally selected as the initial XVs. For example, in the first training set of the Image data set, the first and the 521st data point are exactly the same. So in experiments, we remove the duplicated points as a preprocessing step.

5.7 Some Results on XVs

It is known that the XVs of standard SVM are support vectors, which lie near the classification boundary. Here we give two examples to illustrate what the XVs of SLMC look like.

Example 1. Building an SVM involves solving problem (20)–(22), while building an SLMC is to solve the same problem plus one more constraint (23). If we want to build an SLMC with the same number of XVs as a standard SVM, namely $N_z = N_{SV}$, then the optimal solution of problem (20)–(22) is also a *global optimal* solution of problem (20)–(23), since it satisfies all the constraints. So in this special case, the support vectors of the standard SVM are also an optimal choice of XVs for SLMC.

Example 2. On the USPS data set, we built an SVM on training data with $\gamma = 1/128$, $C = 10$ to separate digit '3' from digit '8'. The resulting SVM has 116 SVs and a test error rate of 1.8%. Then we built an SLMC with the same γ and C, while $Nz = 12$ (i.e. about 10% of the number of SVs). The resulting SLMC also has a test error rate of 1.8%. As shown in Figure 1, the images of the 12 XVs produced by SLMC approach look like digits.

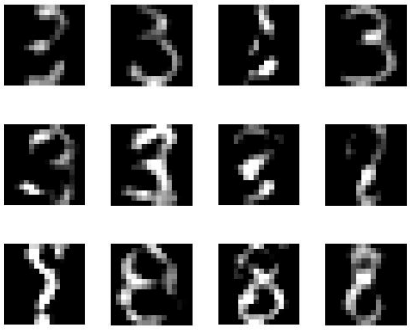


Figure 1: Images of XVs for separating '3' and '8'.

5.8 Training Time of SLMC

Building an SLMC is a gradient based process, where each iteration consists of computing the $\phi_z(\mathbf{x}_i)$, $1 \leq i \leq N$, training a linear SVM over $\{\phi_z(\mathbf{x}_i), y_i\}_{i=1}^N$,[14] and then computing the gradient $\nabla W(\mathbf{Z})$.

Let $T_{SVM}(N, d)$ denote the time complexity of training a linear SVM over a data set containing N d-dimensional vectors, then the time complexity of training an SLMC is

$$O(n \times (NN_z d + T_{SVM}(N, N_z) + N_z^3 + N_z^2 d)),$$

where n is the number of iterations of the SLMC training process. In experiments, we find that the SLMC algorithm requires 20–200 iterations to converge.

We cannot directly compare the training time of SLMC with RS methods and RVM (relevance vector machine), because we used C++ to implement our approach, while the publicly available code of RS methods and the RVM is written in Matlab. Using these implementations and a personal computer with a Pentium 4 CPU of 3GHz, one gets the following numbers: On the USPS data set, SLMC takes 6.9 hours to train, while the RS method takes 2.3 hours. On the Banana data set, SLMC training is about 1.5 seconds, and RVM training is about 5 seconds.

Training an SLMC is time consuming on large data sets. However in practice, once the training is finished, the trained KM will be put into use processing large amount of test data. For applications where processing speed is important, such as real time computer vision, sparse KMs can be of great value. This is the reason why several SKLAs have been developed although they are more expensive than the standard SVM algorithm.

Furthermore, some kernel learning algorithms are not sparse at all, such as kernel ridge regression, KFD, KPCA, which means that all the training data need to be saved as the XVs in the resulting KMs trained by these algorithms. Hence building sparse versions of these algorithms can not only accelerate the evaluation of the test data, but also dramatically

14. Equivalently we can build an SVM with the kernel function $\hat{K}_z$ over $\{\mathbf{x}_i, y_i\}_{i=1}^N$. But this is much slower because it is time consuming to compute $\hat{K}_z(\cdot, \cdot)$ defined by (40).

reduce the space needed for storing the trained KM. An example of this will be given in appendix A.

6. Conclusions

We present a direct method to build sparse kernel learning algorithms. There are mainly two advantages of this method: First, it can be applied to sparsify many current kernel learning algorithms. Second it simultaneously considers the sparseness and the performance of the resulting KM. Based on this method we propose an approach to build sparse large margin classifiers, which essentially finds a discriminating subspace $\mathcal{F}_z$ of the feature space $\mathcal{F}$. Experimental results indicate that this approach often exhibits a better classification accuracy, at comparable sparsity, than the sparse learning algorithms to which we compared.

A by-product of this paper is a method for calculating the expansion coefficients of SVMs for given XVs. Based on this method we proposed a modified version of the RS method and the RSVM. Experimental results show that these two modified algorithms can improve the classification accuracy of their counterparts. One could also try this method on other algorithms such as the RVM.

Possible future work may include applying the proposed method to other kernel learning algorithms and running the direct method greedily to find the XVs one after another in order to accelerate the training procedure.

Acknowledgments

We would like to acknowledge the anonymous reviewers for their comments that significantly improved the quality of the manuscript.

Appendix A. Sparse KFD

We have derived the SLMC algorithm as an example of our direct sparse learning approach. Here we will show another example to build sparse KFD (Mika et al., 2003).

In the KFD algorithm, we need to consider the following Rayleigh quotient maximization problem (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004).[15]

$$\max_{\mathbf{w}\in\mathcal{F}} \frac{\mathbf{w}^\top \mathbf{A}\mathbf{w}}{\mathbf{w}^\top \mathbf{B}\mathbf{w} + \mu \|\mathbf{w}\|^2}. \tag{49}$$

In (49), $\mathbf{A}$ and $\mathbf{B}$ are respectively the between-class and within-class variances of the training data in the feature space $\mathcal{F}$, while μ is a regularization parameter. It can be seen that both $\mathbf{A}$ and $\mathbf{B}$ are positive definite.

15. As mentioned before, convex formulations have been proposed for the KFD (Mika et al., 2003; Suykens et al., 2002). Here we only consider the traditional non-convex formulation whose solution can be easily obtained via eigen decomposition. This also illustrates that our method can also be applied to non-convex optimization problems in some special cases.

The following is an equivalent form of (49)

$$\min_{\mathbf{w}\in\mathcal{F}} \quad -\mathbf{w}^\top \mathbf{A}\mathbf{w}, \tag{50}$$

$$\text{subject to} \quad \mathbf{w}^\top \hat{\mathbf{B}}\mathbf{w} = 1, \tag{51}$$

where $\hat{\mathbf{B}} = \mathbf{B} + \mu\mathbf{I}$, implying that $\hat{\mathbf{B}}$ is strictly positive definite.

Following our proposed approach, in order to build sparse KFD (SKFD), we have to solve the following problem:

$$\min_{\mathbf{w}\in\mathcal{F},\boldsymbol{\beta}\in\mathcal{R}^{N_z},\mathbf{Z}\in\mathcal{R}^{d\times N_z}} \quad -\mathbf{w}^\top \mathbf{A}\mathbf{w}, \tag{52}$$

$$\text{subject to} \quad \mathbf{w}^\top \hat{\mathbf{B}}\mathbf{w} = 1, \tag{53}$$

$$\mathbf{w} = \sum_{i=1}^{N_z} \phi(\mathbf{z}_i)\beta_i. \tag{54}$$

Substituting (54) into (52) and (53), we have

$$\min_{\boldsymbol{\beta}\in\mathcal{R}^{N_z},\mathbf{Z}\in\mathcal{R}^{d\times N_z}} \quad -\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta}, \tag{55}$$

$$\text{subject to} \quad \boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top = 1, \tag{56}$$

where $\mathbf{A}^z = (\phi(\mathbf{Z}))^\top \mathbf{A}(\phi(\mathbf{Z}))$ and $\mathbf{B}^z = (\phi(\mathbf{Z}))^\top \hat{\mathbf{B}}(\phi(\mathbf{Z}))$, where with a little abuse of symbols, we use $\phi(\mathbf{Z})$ to denote the matrix $[\phi(\mathbf{z}_1), \ldots, \phi(\mathbf{z}_{N_z})]$. Note that in the problem (55)–(56), $\mathbf{A}^z \in \mathcal{R}^{N_z\times N_z}$ is positive definite, while $\mathbf{B}^z \in \mathcal{R}^{N_z\times N_z}$ is strictly positive definite.

As before, we define the marginal function $W(\mathbf{Z})$ as the minimal value of the following optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathcal{R}^{N_z}} \quad -\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta}, \tag{57}$$

$$\text{subject to} \quad \boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top = 1. \tag{58}$$

Note the above problem is the same as the problem (55)–(56) except that in the above problem, $\mathbf{Z}$ is fixed rather than variable.

Now we need to consider how to compute $W(\mathbf{Z})$ and $\nabla W(\mathbf{Z})$. To compute the function value $W(\mathbf{Z})$, we need to solve the problem (57)–(58). By the Lagrange multiplier method, this problem can be solved by solving the following unconstrained optimization problem:

$$\min_{\boldsymbol{\beta}\in\mathcal{R}^{N_z},\lambda\in\mathcal{R}} J(\boldsymbol{\beta},\lambda) = -\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta} + \lambda(\boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top - 1), \tag{59}$$

with Lagrange multiplier $\lambda \in \mathcal{R}$.

The derivative of $J(\boldsymbol{\beta},\lambda)$ with respect to $\boldsymbol{\beta}$ and λ must vanish, leading to

$$\mathbf{A}^z\boldsymbol{\beta} = \lambda\mathbf{B}^z\boldsymbol{\beta}, \tag{60}$$

$$\boldsymbol{\beta}^\top \mathbf{B}^z\boldsymbol{\beta}^\top = 1. \tag{61}$$

Equation (60) shows that $\boldsymbol{\beta}$ should be an eigenvector of the matrix $(\mathbf{B}^z)^{-1}\mathbf{A}^z$ and λ should be the corresponding eigenvalue. Left multiplying both sides of equation (60) by $\boldsymbol{\beta}^\top$ and using (61), we have

$$\boldsymbol{\beta}^\top \mathbf{A}^z\boldsymbol{\beta} = \lambda. \tag{62}$$

Since $-\boldsymbol{\beta}^\top \mathbf{A}^z \boldsymbol{\beta}$ is the objective function (57) we are minimizing, we can see that its minimal value should equal the negative of the largest eigenvalue of $(\mathbf{B}^z)^{-1}\mathbf{A}^z$. Therefore the function value $W(\mathbf{Z})$ is obtained.

Let $\bar{\boldsymbol{\beta}}$ and $\bar{\lambda}$ denote the optimal solution and the corresponding Lagrange multiplier of problem (57)–(58), as derived above, $\bar{\lambda}$ is the largest eigenvalue of $(\mathbf{B}^z)^{-1}\mathbf{A}^z$ and $\bar{\boldsymbol{\beta}}$ is the corresponding eigenvector multiplied by a constant such that equation (61) is satisfied.

As mentioned above, $\mathbf{B}^z$ is strictly positive definite. Here we assume that that there is an unique eigenvector corresponding to $\bar{\lambda}$. As equation (58) is the only constraint of problem (57)–(58), the optimal solution $\bar{\boldsymbol{\beta}}$ is Mangasarian-Fromovitz regular. And it is straightforward to verify that the set of feasible solutions $\mathcal{S}(\mathbf{Z})$ is uniformly compact if $\mathbf{B}^z$ is strictly positive definite. Therefore according to lemma 1, the derivative of $W(\mathbf{Z})$ with respect to $\mathbf{z}_{uv}$, which denotes the v-th component of vector $\mathbf{z}_u$, $1 \le u \le N_z, 1 \le v \le d$, exists and can be computed as follows:

$$\frac{\partial W(\mathbf{Z})}{\partial \mathbf{z}_{uv}} = -\bar{\boldsymbol{\beta}}^\top \frac{\partial \mathbf{A}^z}{\partial \mathbf{z}_{uv}} \bar{\boldsymbol{\beta}} + \bar{\lambda} \bar{\boldsymbol{\beta}}^\top \frac{\partial \mathbf{B}^z}{\partial \mathbf{z}_{uv}} \bar{\boldsymbol{\beta}}. \tag{63}$$

Now that both the function value $W(\mathbf{Z})$ and the gradient $\nabla W(\mathbf{Z})$ can be computed, the (local) optimum of the problem (52)–(54) can be computed by the gradient based algorithm.

After obtaining the XVs $\mathbf{z}_i$ by solving the problem (52)–(54), we can take the solution β_i of this problem as the expansion coefficients. However we can not get the bias b for the resulting KM in this way (c.f equation (1)). As mentioned before, having $\mathbf{z}_i$, we can apply our proposed method that the expansion coefficients and the bias can be calculated by solving the problem (37)–(39).

Experimental results on six classification benchmarks for the proposed SKFD algorithm are provided in table 3.

Data Set		Banana	Breast Cancer	Titanic	Waveform	German	Image
SVM	N_{SV}	86.7	112.8	70.6	158.9	408.2	172.1
	Error(%)	11.8	28.6	22.1	9.9	22.5	2.8
KFD	N_z	400	200	150	400	700	1300
	Error(%)	10.8	25.8	23.2	9.9	23.7	3.3
N_z/N_{SV} = 10%	RS	21.9	27.9	26.6	10.0	22.9	18.3
	MRS	17.5	29.0	**22.6**	**9.9**	**22.6**	6.9
	RSVM	17.5	31.0	22.9	11.6	24.5	14.2
	MRSVM	16.9	30.3	23.9	11.8	23.7	12.7
	SKFD	**10.8**	**25.5**	23.5	**9.9**	23.5	**4.0**

Table 3: Results on six classification benchmarks. The SKFD is initialized with the k-means algorithm. The best results among the sparse learning algorithms are in boldface. Similarly as before, the results reported here are the averages over the first 10 training/test splits, except for the KFD algorithm, whose results are taken directly from (Schölkopf and Smola, 2002), which are the averages over all the 100 training/test splits. For the KFD algorithm, the number of expansion vectors N_z is the same as the number of training data since KFD is not sparse at all.

The results presented in table 3 validate the effectiveness of the proposed SKFD algorithm. Furthermore, the original KFD algorithm is not sparse all all, i.e. all the training data need to be stored as the XVs. Therefore the proposed SKFD algorithm not only accelerates the test phase, but also significantly reduces the space needed for storing the resulting KM. For example, the Banana data set contains 400 training data, implying that on average only $86.7 \times 10\% = 8.7$ XVs need to stored in the resulting KM trained by the SKFD, saving about $1 - 8.7/400 = 97.8\%$ storage compared with the orignial KFD algorithm.

Appendix B. Formal Definitions of the Two Conditions in Lemma 1

For each $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, let $\mathcal{S}(\mathbf{Z})$ denote the feasible set of problem (16)–(18)

$$\mathcal{S}(\mathbf{Z}) = \{\mathbf{x} \mid g_i(\mathbf{x}, \mathbf{Z}) \leq 0,\ 1 \leq i \leq N_g\} \cap \{\mathbf{x} \mid h_j(\mathbf{x}, \mathbf{Z}) = 0,\ 1 \leq j \leq N_h\}.$$

Definition 3 *(Gauvin and Dubeau, 1982) The feasible set $\mathcal{S}(\mathbf{Z})$ of problem (16)–(18) is uniformly compact at $\bar{\mathbf{Z}}$ if there is a neighborhood $\mathcal{N}(\bar{\mathbf{Z}})$ of $\bar{\mathbf{Z}}$ such that the closure of $\bigcup_{\mathbf{Z} \in \mathcal{N}(\bar{\mathbf{Z}})} \mathcal{S}(\mathbf{Z})$ is compact.*

Definition 4 *(Mangasarian, 1969) For any $\mathbf{Z}$, a feasible point $\bar{\mathbf{x}} \in \mathcal{S}(\mathbf{Z})$ of problem (16)–(18) is said to be Mangasarian-Fromovitz regular if it satisfies the following Mangasarian-Fromovitz regularity condition:*

1. *There exists a vector $\mathbf{v}$ such that*

$$\mathbf{v}^\top \nabla_{\mathbf{x}} g_i(\mathbf{x}, \mathbf{Z})|_{\mathbf{x}=\bar{\mathbf{x}}} < 0, \quad i \in \{i \mid g_i(\bar{\mathbf{x}}, \mathbf{Z}) = 0\}, \tag{64}$$

$$\mathbf{v}^\top \nabla_{\mathbf{x}} h_j(\mathbf{x}, \mathbf{Z})|_{\mathbf{x}=\bar{\mathbf{x}}} = 0, \quad 1 \leq j \leq N_h, \tag{65}$$

2. *The gradients $\{\nabla_{\mathbf{x}} h_j(\mathbf{x}, \mathbf{Z})|_{\mathbf{x}=\bar{\mathbf{x}}},\ 1 \leq j \leq N_h\}$ are linearly independent,*

where $\nabla_{\mathbf{x}}$ denotes the gradient with respect to the variables $\mathbf{x}$.

Appendix C. Proof of Corollary 2

Proof The proof is just to verify all the conditions in lemma 1.

First, for the soft margin SVM training problem (37)–(39), it is known that if the kernel function $\hat{K}_z(\cdot, \cdot)$ is strictly positive definite then the problem has an unique solution and the corresponding Lagrange multipliers are also unique.

Second, it can be seen that at any $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$, the set of feasible solutions $\mathcal{S}(\mathbf{Z})$ is compact and does not depend on $\mathbf{Z}$, therefore $\mathcal{S}(\mathbf{Z})$ is uniformly compact at any $\mathbf{Z} \in \mathcal{R}^{d \times N_z}$.

Third, obviously the second part of the Mangasarian-Fromovitz regularity condition holds since there is only one equality constraint. Now we prove that the first part of the Mangasarian-Fromovitz regularity condition also holds by constructing a vector $\mathbf{v} = [v_1, \ldots, v_N]^\top \in \mathcal{R}^N$ that satisfies both (64) and (65). For ease of description, we partition the index set $\{1, \ldots, N\}$ into the following three subsets according to α_i^z $(1 \leq i \leq N)$, which are the solution of problem (37)–(39): $S_1 = \{i \mid \alpha_i^z = 0\}$, $S_2 = \{i \mid \alpha_i^z = C\}$ and $S_3 = \{i \mid 0 < \alpha_i^z < C\}$. Furthermore, for any vector $\mathbf{t} = [t_1, \ldots, t_N]^\top \in \mathcal{R}^N$, we use $\mathbf{t}_{s_k}$ $(1 \leq k \leq 3)$ to denote the sub-vector of $\mathbf{t}$ that is composed of t_i for $i \in S_k$, $1 \leq k \leq 3$.

First, to make (64) hold, we can simply assign an arbitrary positive value to v_i if $i \in S_1$, and an arbitrary negative value to v_j if $j \in S_2$. To make (65) true, we distinguish two cases:

Case 1, $|S_3| > 0$. In this case, the vector $\mathbf{v}_{s_3}$ can be easily computed as

$$\mathbf{v}_{s_3} = -\frac{\mathbf{y}_{s_3}}{\|\mathbf{y}_{s_3}\|^2} \sum_{i \in S_1 \cup S_2} v_i y_i,$$

where $\mathbf{y} = [y_1, \ldots y_N]^\top$. The above equation results in $\mathbf{v}^\top \mathbf{y} = 0$, which is the same as (65).

Case 2, $|S_3| = 0$. In this case, all the resulting support vectors correspond to α_i^z for $i \in S_2$, which come from both classes according to the assumptions in corollary 2. Therefore the left side of equation (65) equals

$$\begin{aligned} \mathbf{v}^\top \mathbf{y} &= \sum_{i \in S_1} v_i y_i + \sum_{i \in S_2,\ y_i > 0} v_i y_i + \sum_{i \in S_2,\ y_i < 0} v_i y_i \\ &= \sum_{i \in S_1} v_i y_i + \sum_{i \in S_2,\ y_i > 0} v_i - \sum_{i \in S_2,\ y_i < 0} v_i. \end{aligned} \tag{66}$$

Recall that we construct $\mathbf{v}$ such that $v_i < 0$ for $i \in S_2$. So if $\mathbf{v}^\top \mathbf{y} = 0$, equation (65) already holds. If $\mathbf{v}^\top \mathbf{y} > 0$, we can always decrease the values (or equivalently, increase the absolute values) of v_i in the second term of equation (66) to make $\mathbf{v}^\top \mathbf{y} = 0$, while at the same time to keep $v_i < 0$ for $i \in S_2$ so that (64) still holds. Similarly, if $\mathbf{v}^\top \mathbf{y} < 0$, we can decrease the values of v_i in the third term of equation (66).

Thus the first part of the Mangasarian-Fromovitz regularity condition also holds. Hence the optimal solution of problem (37)–(39) is Mangasarian-Fromovitz regular.

Therefore all the conditions in lemma 1 are satisfied and (46) follows from (19) since the constraints of problem (37)–(39) do not depend on $\mathbf{Z}$, which means both the second and the third terms in (19) are $\mathbf{0}$. ■

References

K. P. Bennett. Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, pages 307–326. The MIT Press, Cambridge MA, 1999.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proc. 13th International Conference on Machine Learning*, pages 71–77. Morgan Kaufmann, 1996.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

J. Gauvin and F. Dubeau. Differential properties of the marginal function in mathematical programming. *Mathematical Programming Study*, 19:101–119, 1982.

G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharyya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

Y. Lee and O. L. Mangasarian. RSVM: reduced support vector machines. In *CD Proceedings of the First SIAM International Conference on Data Mining*, Chicago, 2001.

K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14:1449–1459, 2003.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.

O. L. Mangasarian. *Nonlinear Programming*. McGraw-Hill, New York, 1969.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A. J. Smola, and K.-R. Mueller. Constructing descriptive and discriminative non-linear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):623–628, 2003.

P. B. Nair, A. Choudhury, and A. J. Keane. Some greedy learning algorithms for sparse regression and classification with Mercer kernels. *Journal of Machine Learning Research*, 3:781–801, 2002.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1259–1266. MIT Press, Cambridge, MA, 2006.

I. Steinwart. Sparseness of support vector machine. *Journal of Machine Learning Research*, 4:1071–1105, 2003.

J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

M. Wu, B. Schölkopf, and G. Bakir. Building sparse large margin classifiers. In L. D. Raedt and S. Wrobel, editors, *Proc. 22th International Conference on Machine Learning*, pages 1001–1008. ACM, 2005.

Journal of Machine Learning Research 7 (2006) 625–644 Submitted 1/05; Revised 9/05; Published 4/06

Stochastic Complexities of Gaussian Mixtures in Variational Bayesian Approximation

Kazuho Watanabe KAZUHO23@PI.TITECH.AC.JP
Department of Computational Intelligence and Systems Science
Tokyo Institute of Technology
MailBox R2-5, 4259 Nagatsuta, Midori-ku, Yokohama, 226-8503, Japan

Sumio Watanabe SWATANAB@PI.TITECH.AC.JP
P & I Lab.
Tokyo Institute of Technology
MailBox R2-5, 4259 Nagatsuta, Midori-ku, Yokohama, 226-8503, Japan

Editor: Tommi Jaakkola

Abstract

Bayesian learning has been widely used and proved to be effective in many data modeling problems. However, computations involved in it require huge costs and generally cannot be performed exactly. The variational Bayesian approach, proposed as an approximation of Bayesian learning, has provided computational tractability and good generalization performance in many applications.

The properties and capabilities of variational Bayesian learning itself have not been clarified yet. It is still unknown how good approximation the variational Bayesian approach can achieve. In this paper, we discuss variational Bayesian learning of Gaussian mixture models and derive upper and lower bounds of variational stochastic complexities. The variational stochastic complexity, which corresponds to the minimum variational free energy and a lower bound of the Bayesian evidence, not only becomes important in addressing the model selection problem, but also enables us to discuss the accuracy of the variational Bayesian approach as an approximation of true Bayesian learning.

Keywords: Gaussian mixture model, variational Bayesian learning, stochastic complexity

1. Introduction

A Gaussian mixture model is a learning machine which estimates the target probability density by the sum of normal distributions. This learning machine is widely used especially in statistical pattern recognition and data clustering. In spite of wide range of its applications, its properties have not yet been made clear enough. This is because the Gaussian mixture model is a non-regular statistical model. A statistical model is regular if and only if a set of conditions (referred to as "regularity conditions") that ensure the asymptotic normality of the maximum likelihood estimator is satisfied. The regularity conditions are not satisfied for mixture models because the parameters are not identifiable, in other words, the mapping from parameters to probability distributions is not one-to-one. Other than mixture models, statistical models with hidden variables such as hidden Markov models and Bayesian networks fall into the class of non-regular models.

Recently, a lot of attentions has been paid to the non-regular models. In Bayesian learning, mathematical foundation for analyzing non-regular models was established with an algebraic geometrical method (Watanabe, 2001). The Bayesian stochastic complexities or the marginal like-

lihoods of several non-regular models have been clarified in some recent studies (Yamazaki and Watanabe, 2003a,b). The Bayesian framework provides better generalization performance in non-regular models than the maximum likelihood (ML) method that tends to overfit the data.

In the Bayesian framework, rather than learning a single model, one computes the distribution over all possible parameter values and considers an ensemble with respect to the posterior distribution. However, computing the Bayesian posterior can seldom be performed exactly and requires some approximations. Well-known approximate methods include Markov chain Monte Carlo (MCMC) methods and the Laplace approximation. The former attempts to find the exact posterior distribution but typically requires huge computational resources. The latter approximates the posterior distribution by a Gaussian distribution, which can be insufficient for models containing hidden variables.

The variational Bayesian (VB) framework was proposed as another approximation for computations in the models with hidden variables (Attias, 1999; Ghahramani and Beal, 2000). This framework provides computationally tractable posterior distributions over the hidden variables and the parameters with an iterative algorithm. The variational Bayesian framework has been applied to various real-world data modeling problems and empirically proved to be both computational tractable and generalize well.

The properties of variational Bayesian learning remain unclear from a theoretical stand point. Although the variational Bayesian framework is an approximation, questions like how accurately it approximates the true distribution have yet to be answered.

In this paper, we focus on variational Bayesian learning of Gaussian mixture models. As the main contribution, we derive asymptotic upper and lower bounds on the variational stochastic complexity. It is shown that the variational stochastic complexity is smaller than in regular statistical models, so the advantage of Bayesian learning still remains in variational Bayesian learning. The variational stochastic complexity, which corresponds to the minimum variational free energy and a lower bound of the Bayesian evidence, is an important quantity for model selection. Giving the asymptotic bounds on it also contributes to the following two issues. One is the accuracy of variational Bayesian learning as an approximation method since the variational stochastic complexity shows the distance from the variational posterior distribution to the true Bayesian posterior distribution in terms of Kullback information. Another is the influence of the hyperparameters on the learning process. Since the variational Bayesian algorithm minimizes the variational free energy, the derived bounds indicate how the hyperparameters influence the learning process. Our results indicate how to determine the hyperparameter values before the learning process.

We consider the case in which the true distribution is contained in the learned model, in other words, the model has redundant components to attain the true distribution. Analyzing the variational stochastic complexity in this case is most valuable for comparing variational Bayesian learning with true Bayesian learning. This is because the advantage of Bayesian learning is typical in this case (Watanabe, 2001). Furthermore, this analysis is necessary and essential for addressing the model selection and hypothesis testing problems.

This paper is organized as follows. In Section 2, the Gaussian mixture model is briefly introduced. In Section 3, we describe Bayesian learning. In Section 4, the variational Bayesian framework is outlined and the variational stochastic complexity is defined. In Section 5, we state the main theorem of this paper. The main theorem is proved in Section 6. In Section 7, we experimentally examine the quality of the bounds given in the main theorem. Discussion and conclusions follow in Section 8 and Section 9.

2. Gaussian Mixture Models

Denote by $g(x|\mu,\Sigma)$ a density function of an M-dimensional normal distribution whose mean is $\mu \in R^M$ and variance-covariance matrix is $\Sigma \in R^{M\times M}$. A Gaussian mixture model $p(x|\theta)$ of an M-dimensional input $x \in R^M$ with a parameter vector θ is defined by

$$p(x|\theta) = \sum_{k=1}^{K} a_k g(x|\mu_k, \Sigma_k),$$

where integer K is the number of components and $\{a_k | a_k \geq 0, \sum_{k=1}^{K} a_k = 1\}$ is the set of mixing proportions. The parameter θ of the model is $\theta = \{a_k, \mu_k, \Sigma_k\}_{k=1}^{K}$.

In some applications, the parameter is restricted to the means of each component and it is assumed that there is no correlation between each input dimension. In this case, the model is written by

$$p(x|\theta) = \sum_{k=1}^{K} \frac{a_k}{\sqrt{2\pi\sigma_k^2}^M} \exp(-\frac{\|x-\mu_k\|^2}{2\sigma_k^2}), \tag{1}$$

where $\sigma_k > 0$ is a constant.

In this paper, we consider this type eq.(1) of Gaussian mixture models in the variational Bayesian framework and show upper and lower bounds of the variational stochastic complexity in Theorem 3.

The Gaussian mixture model can be rewritten as follows using a hidden variable $y = (y^1, \cdots, y^K) \in \{(1,0,\cdots,0),(0,1,\cdots,0),\cdots,(0,0,\cdots,1)\}$,

$$p(x,y|\theta) = \prod_{k=1}^{K} \Big[\frac{a_k}{\sqrt{2\pi\sigma_k^2}^M} \exp\{-\frac{\|x-\mu_k\|^2}{2\sigma_k^2}\} \Big]^{y^k}.$$

The hidden variable y is not observed and is representing the component from which the datum x is generated. If the datum x is from the kth component, then $y^k = 1$, if otherwise, $y^k = 0$. And

$$\sum_{y} p(x,y|\theta) = p(x|\theta)$$

holds where the sum over y ranges over all possible values of the hidden variable.

The Gaussian mixture model is a non-regular statistical model, since the parameters are non-identifiable. More specifically, if the true distribution can be realized by a model with the smaller number of components, the true parameter is not a point but an analytic set with singularities. If the parameters are non-identifiable, the usual asymptotic theory of regular statistical models cannot be applied. Some studies have revealed that Gaussian mixture models have quite different properties from those of regular statistical models. In particular, the Gaussian mixture model given by eq.(1) has been studied as a prototype of non-regular models in the case of the maximum likelihood estimation(Hartigan, 1985; Dacunha-Castelle and Gassiat, 1997).

3. Bayesian Learning

Suppose n training samples $X^n = \{x_1, \cdots, x_n\}$ are independently and identically taken from the true distribution $p_0(x)$. In Bayesian learning of a model $p(x|\theta)$ whose parameter is θ, first, the prior

distribution $\varphi(\theta)$ on the parameter θ is set. Then the posterior distribution $p(\theta|X^n)$ is computed from the given data set and the prior by

$$p(\theta|X^n) = \frac{1}{Z(X^n)}\varphi(\theta)\prod_{i=1}^{n} p(x_i|\theta),$$

where $Z(X^n)$ is the normalization constant that is also known as the marginal likelihood or the Bayesian evidence of the data set X^n (Mackay, 1992).

The Bayesian predictive distribution $p(x|X^n)$ is given by averaging the model over the posterior distribution as follows,

$$p(x|X^n) = \int p(x|\theta)p(\theta|X^n)d\theta,$$

and its generalization error can be measured by the Kullback information from the true distribution,[1]

$$K(p_0(x)||p(x|X^n)) = \int p_0(x)\log\frac{p_0(x)}{p(x|X^n)}dx.$$

The Bayesian stochastic complexity $F(X^n)$ is defined by

$$F(X^n) = -\log Z(X^n), \tag{2}$$

which is also called the free energy and is important in most data modeling problems. Practically, it is used as a criterion by which the learning model is selected and the hyperparameters in the prior are optimized (Akaike, 1980; Schwarz, 1978).

The Bayesian posterior can be rewritten as

$$p(\theta|X^n) = \frac{1}{Z_0(X^n)}\exp(-nH_n(\theta))\varphi(\theta),$$

where $H_n(\theta)$ is the empirical Kullback information,

$$H_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}\log\frac{p_0(x_i)}{p(x_i|\theta)}, \tag{3}$$

and $Z_0(X^n)$ is the normalization constant. Let

$$S(X^n) = -\sum_{i=1}^{n}\log p_0(x),$$

and define the normalized Bayesian stochastic complexity $F_0(X^n)$ by

$$\begin{aligned} F_0(X^n) &= -\log Z_0(X^n) \\ &= F(X^n) - S(X^n). \end{aligned} \tag{4}$$

1. Throughout this paper, we use the notation $K(q(x)||p(x))$ for the Kullback information from a distribution $q(x)$ to a distribution $p(x)$, that is,

$$K(q(x)||p(x)) = \int q(x)\log\frac{q(x)}{p(x)}dx.$$

It is noted that the empirical entropy $S(X^n)$ does not depend on the model $p(x|\theta)$. Therefore minimization of $F(X^n)$ is equivalent to that of $F_0(X^n)$.

Let $E_{X^n}[\cdot]$ denote the expectation over all sets of training samples. Then it follows from eq.(4) that

$$E_{X^n}[F(X^n) - F_0(X^n)] = nS,$$

where $S = -\int p_0(x)\log p_0(x)dx$ is the entropy. There is the following relationship between the average Bayesian stochastic complexity and the average generalization error(Levin et al., 1990),

$$\begin{aligned} E_{X^n}[K(p_0(x)||p(x|X^n))] &= E_{X^{n+1}}[F(X^{n+1})] - E_{X^n}[F(X^n)] - S \\ &= E_{X^{n+1}}[F_0(X^{n+1})] - E_{X^n}[F_0(X^n)]. \end{aligned} \tag{5}$$

Recently, in Bayesian learning, an advanced mathematical method for analyzing non-regular models was established(Watanabe, 2001), which enabled us to clarify the asymptotic behavior of the Bayesian stochastic complexity of non-regular models. More specifically, by using concepts in algebraic analysis, it was proved that the average normalized Bayesian stochastic complexity defined by $E_{X^n}[F_0(X^n)]$ has the following asymptotic form,

$$E_{X^n}[F_0(X^n)] \simeq \lambda\log n - (m-1)\log\log n + O(1), \tag{6}$$

where λ and m are the rational number and the natural number respectively which are determined by the singularities of the true parameter. In regular statistical models, 2λ is equal to the number of parameters and $m = 1$, whereas in non-regular models such as Gaussian mixture models, 2λ is not larger than the number of parameters and $m \geq 1$. This means non-regular models have an advantage in Bayesian learning. From eq.(5), if the asymptotic form of the average normalized Bayesian stochastic complexity is given by eq.(6), the average generalization error is given by

$$E_{X^n}[K(p_0(x)||p(x|X^n))] \simeq \frac{\lambda}{n} + o(\frac{1}{n}). \tag{7}$$

Since the coefficient λ is proportional to the average generalization error, Bayesian learning is more suitable for non-regular models than the maximum likelihood (ML) method.

However, in order to carry out Bayesian learning practically, one computes the Bayesian stochastic complexity or the predictive distribution by integrating over the posterior distribution, which typically cannot be performed analytically.

Hence, approximations must be made. The Laplace approximation is a well-known and simple method that approximates the posterior distribution by a Gaussian distribution. This approach gives reasonable approximation in the case of regular statistical models whose posteriors converge to normal distributions as the sample size n tends to infinity. In contrast, posterior distributions of non-regular models do not converge to normal distributions in general, even as n tends to infinity. Therefore, the Laplace approximation can be insufficient for non-regular models. Markov chain Monte Carlo (MCMC) method can provide a better approximation. It attempts to sample from the exact posterior distribution but typically requires vast computational resources.

As another approximation, the variational Bayesian framework was proposed (Attias, 1999; Beal, 2003; Ghahramani and Beal, 2000).

4. Variational Bayesian Learning

In this section, we outline the variational Bayesian framework and define the variational stochastic complexity.

4.1 The Variational Bayesian Framework

Using the complete likelihood of the data $\{X^n, Y^n\}$, with the corresponding hidden variables $Y^n = \{y_1, \cdots, y_n\}$, we can rewrite the Bayesian stochastic complexity eq.(2) as

$$\begin{aligned} F(X^n) &= -\log \int \sum_{Y^n} \varphi(\theta) \prod_{i=1}^{n} p(x_i, y_i|\theta) d\theta \\ &= -\log \int \sum_{Y^n} p(X^n, Y^n, \theta) d\theta, \end{aligned}$$

where the sum over Y^n ranges over all possible values of all hidden variables.

The variational Bayesian framework starts by upper bounding the Bayesian stochastic complexity. For an arbitrary conditional distribution $q(Y^n, \theta|X^n)$ on the hidden variables and the parameters, the Bayesian stochastic complexity can be upper bounded by applying Jensen's inequality,

$$\begin{aligned} F(X^n) &\leq \sum_{Y^n} \int q(Y^n, \theta|X^n) \log \frac{q(Y^n, \theta|X^n)}{p(X^n, Y^n, \theta)} d\theta \\ &\equiv \overline{F}[q]. \end{aligned}$$

This inequality becomes an equality if and only if $q(Y^n, \theta|X^n) = p(Y^n, \theta|X^n)$, that is, $q(Y^n, \theta|X^n)$ equals the Bayesian posterior distribution. This means that the smaller the functional $\overline{F}[q]$ is, the closer the distribution $q(Y^n, \theta|X^n)$ is to the true Bayesian posterior distribution. The functional $\overline{F}[q]$ is called the variational free energy.

The variational Bayesian approach makes an approximation to ensure a computationally tractable posterior. More specifically, assuming the parameters and the hidden variables are conditionally independent of each other, the variational Bayesian approach restricts the set of $q(Y^n, \theta|X^n)$ to distributions that have the form

$$q(Y^n, \theta|X^n) = Q(Y^n|X^n) r(\theta|X^n), \tag{8}$$

where $Q(Y^n|X^n)$ and $r(\theta|X^n)$ are probability distributions over the hidden variables and the parameters respectively. The distribution $q(Y^n, \theta|X^n)$ that minimizes the functional $\overline{F}[q]$ is termed the optimal variational posterior and generally differs from the true Bayesian posterior.

Minimization of the functional $\overline{F}[q]$ with respect to the distributions $Q(Y^n|X^n)$ and $r(\theta|X^n)$ can be performed by using variational methods. Solving the minimization problem under the constraints $\int r(\theta|X^n) d\theta = 1$ and $\sum_{Y^n} Q(Y^n|X^n) = 1$ gives the following theorem. The proof is well-known(Beal, 2003; Sato, 2001), thus it is omitted in this paper.

Theorem 1 *If the functional $\overline{F}[q]$ is minimized under the constraint eq.(8) then the variational posteriors, $r(\theta|X^n)$ and $Q(Y^n|X^n)$, satisfy*

$$r(\theta|X^n) = \frac{1}{C_r} \varphi(\theta) \exp \langle \log p(X^n, Y^n|\theta) \rangle_{Q(Y^n|X^n)}, \tag{9}$$

and

$$Q(Y^n|X^n) = \frac{1}{C_Q} \exp \langle \log p(X^n, Y^n|\theta) \rangle_{r(\theta|X^n)}, \tag{10}$$

where C_r and C_Q are the normalization constants.[2]

2. Hereafter for an arbitrary distribution $p(x)$, we use the notation $\langle \cdot \rangle_{p(x)}$ for the expected value over $p(x)$.

Note that eq.(9) and eq.(10) give only the necessary condition for $r(\theta|X^n)$ and $Q(Y^n|X^n)$ minimize the functional $\overline{F}[q]$. The variational posteriors that satisfy eq.(9) and eq.(10) are searched by an iterative algorithm. It is known that this algorithm is a natural gradient method when the model is in the general exponential family of models with hidden variables (Sato, 2001).

4.2 Stochastic Complexity of Variational Bayes

We define the variational stochastic complexity $\overline{F}(X^n)$ by the minimum value of the functional $\overline{F}[q]$ attained by the above optimal variational posteriors, that is ,

$$\overline{F}(X^n) = \min_{r,Q} \overline{F}[q].$$

The variational stochastic complexity $\overline{F}(X^n)$ gives an estimate (upper bound) for the true Bayesian stochastic complexity $F(X^n)$, which is the minus log evidence. Therefore, $\overline{F}(X^n)$ is used for the model selection in variational Bayesian learning(Beal, 2003). Moreover, the difference between $\overline{F}(X^n)$ and the Bayesian stochastic complexity $F(X^n)$ is the Kullback information from the optimal variational posterior to the true posterior. That is

$$\overline{F}(X^n) - F(X^n) = \min_{r,Q} K(q(Y^n,\theta|X^n)||p(Y^n,\theta|X^n)).$$

Hence, comparison between $\overline{F}(X^n)$ and $F(X^n)$ shows the accuracy of the variational Bayesian approach as an approximation of true Bayesian learning.

We define the normalized variational stochastic complexity $\overline{F}_0(X^n)$ by

$$\overline{F}_0(X^n) = \overline{F}(X^n) - S(X^n). \tag{11}$$

From Theorem 1, the following lemma is obtained. The proof is given in Appendix.

Lemma 2

$$\overline{F}_0(X^n) = \min_{r(\theta|X^n)} \{K(r(\theta|X^n)||\varphi(\theta)) - (\log C_Q + S(X^n))\}, \tag{12}$$

where

$$C_Q = \sum_{Y^n} \exp \langle \log p(X^n,Y^n|\theta) \rangle_{r(\theta|X^n)}.$$

The variational posteriors $r(\theta|X^n)$ and $Q(Y^n|X^n)$ that satisfy eq.(9) and eq.(10) are parameterized by the variational parameter $\overline{\theta}$ defined by

$$\overline{\theta} = \langle \theta \rangle_{r(\theta|X^n)},$$

if the model $p(x,y|\theta)$ is included in the exponential family(Beal, 2003; Ghahramani and Beal, 2000). Then it is noted that C_Q in eq.(12) is also parameterized by $\overline{\theta}$. Therefore, henceforth we denote $r(\theta|X^n)$ and C_Q as $r(\theta|\overline{\theta})$ and $C_Q(\overline{\theta})$ when they are regarded as functions of the variational parameter $\overline{\theta}$.

We define the variational estimator $\overline{\theta}_{vb}$ of θ by the variational parameter $\overline{\theta}$ that attains the minimum value of the normalized variational stochastic complexity $\overline{F}_0(X^n)$. By this definition, Lemma 2 claims that

$$\overline{\theta}_{vb} = \operatorname*{argmin}_{\overline{\theta}} \{K(r(\theta|\overline{\theta})||\varphi(\theta)) - (\log C_Q(\overline{\theta}) + S(X^n))\}. \tag{13}$$

In variational Bayesian learning, the variational parameter $\overline{\theta}$ is updated iteratively to find the optimal solution $\overline{\theta}_{vb}$. Therefore, our aim is to evaluate the minimum value of the right hand side of eq.(13) as a function of the variational parameter $\overline{\theta}$.

5. Main Results

In this section, we describe two conditions and give the upper and lower bounds of the normalized variational stochastic complexity in Theorem 3.

We assume the following conditions.

(i) The true distribution $p_0(x)$ is an M-dimensional Gaussian mixture model $p(x|\theta_0)$ which has K_0 components and the parameter $\theta_0 = \{a_k^*, \mu_k^*\}_{k=1}^{K_0}$,

$$p(x|\theta_0) = \sum_{k=1}^{K_0} \frac{a_k^*}{\sqrt{2\pi}^M} \exp(-\frac{\|x-\mu_k^*\|^2}{2}),$$

where $x, \mu_k^* \in R^M$. And suppose that the true distribution can be realized by the model, that is, the model $p(x|\theta)$ has K components,

$$p(x|\theta) = \sum_{k=1}^{K} \frac{a_k}{\sqrt{2\pi}^M} \exp(-\frac{\|x-\mu_k\|^2}{2}), \tag{14}$$

and $K \geq K_0$ holds.

(ii) The prior of the parameters is the product of the following two distributions on $\mathbf{a} = \{a_k\}_{k=1}^K$ and $\mu = \{\mu_k\}_{k=1}^K$

$$\varphi(\mathbf{a}) = \frac{\Gamma(K\phi_0)}{\Gamma(\phi_0)^K} \prod_{k=1}^{K} a_k^{\phi_0 - 1}, \tag{15}$$

$$\varphi(\mu) = \prod_{k=1}^{K} \sqrt{\frac{\xi_0}{2\pi}}^M \exp(-\frac{\xi_0 \|\mu_k - \nu_0\|^2}{2}), \tag{16}$$

where $\xi_0 > 0$, $\nu_0 \in R^M$ and $\phi_0 > 0$ are constants called hyperparameters. These are Dirichlet and normal distributions respectively. They are the conjugate prior distributions and are often used in variational Bayesian learning of Gaussian mixture models.

Under these conditions, we prove the following theorem. The proof will appear in the next section.

Theorem 3 (Main Result) *Assume the conditions (i) and (ii). Then the normalized variational stochastic complexity $\overline{F}_0(X^n)$ defined by eq.(11) satisfies*

$$\underline{\lambda} \log n + nH_n(\overline{\theta}_{vb}) + C_1 \leq \overline{F}_0(X^n) \leq \overline{\lambda} \log n + C_2, \tag{17}$$

with probability 1 for an arbitrary natural number n where C_1, C_2 are constants independent of n and the coefficients $\underline{\lambda}$, $\overline{\lambda}$ are given by

$$\underline{\lambda} = \begin{cases} (K-1)\phi_0 + \frac{M}{2} & (\phi_0 \leq \frac{M+1}{2}), \\ \frac{MK+K-1}{2} & (\phi_0 > \frac{M+1}{2}), \end{cases}$$

$$\overline{\lambda} = \begin{cases} (K-K_0)\phi_0 + \frac{MK_0+K_0-1}{2} & (\phi_0 \leq \frac{M+1}{2}), \\ \frac{MK+K-1}{2} & (\phi_0 > \frac{M+1}{2}). \end{cases} \tag{18}$$

Taking expectation over all sets of training samples, we obtain the following corollary.

Corollary 4 *Assume the conditions (i) and (ii). Then the average of the normalized variational stochastic complexity $\overline{F}_0(X^n)$ satisfies*

$$\underline{\lambda}\log n + E_{X^n}[nH_n(\overline{\theta}_{vb})] + C_1 \leq E_{X^n}[\overline{F}_0(X^n)] \leq \overline{\lambda}\log n + C_2.$$

Remark. The following bounds for the variational stochastic complexity $\overline{F}(X^n) = \overline{F}_0(X^n) + S(X^n)$ are immediately obtained from Theorem 3 and Corollary 4,

$$S(X^n) + \underline{\lambda}\log n + nH_n(\overline{\theta}_{vb}) + C_1 \leq \overline{F}(X^n) \leq S(X^n) + \overline{\lambda}\log n + C_2,$$

and

$$nS + \underline{\lambda}\log n + E_{X^n}[nH_n(\overline{\theta}_{vb})] + C_1 \leq E_{X^n}[\overline{F}(X^n)] \leq nS + \overline{\lambda}\log n + C_2,$$

where $S(X^n) = -\sum_{i=1}^n \log p(x_i|\theta_0)$ is the empirical entropy and $S = -\int p(x|\theta_0)\log p(x|\theta_0)dx$ is the entropy.

Since the dimension of the parameter θ is $MK + K - 1$, the penalty term in the Bayesian information criterion (BIC) (Schwarz, 1978) is given by $\lambda_{\rm BIC}\log n$ where

$$\lambda_{\rm BIC} = \frac{MK + K - 1}{2}. \tag{19}$$

Note that, unlike for regular statistical models, the advantage of Bayesian learning for non-regular models is demonstrated by the asymptotic analysis as seen in eq.(6) and eq.(7). Theorem 3 claims that the coefficient $\overline{\lambda}$ of $\log n$ is smaller than $\lambda_{\rm BIC}$ when $\phi_0 \leq (M+1)/2$. This means the normalized variational stochastic complexity $\overline{F}_0(X^n)$ becomes smaller than the BIC and implies that the advantage of non-regular models in Bayesian learning still remains in variational Bayesian learning.

Theorem 3 also shows how the hyperparameters affect the learning process and implies that the hyperparameter ϕ_0 is the only hyperparameter that the leading term of the normalized variational stochastic complexity $\overline{F}_0(X^n)$ depends on. The effects of the hyperparameters are discussed in Section 8.

In the condition (i), we assume that the true distribution is contained in the learner model ($K_0 \leq K$). This assumption is necessary for assessing model selection or hypothesis testing methods and for developing a new method for these tasks. In real-world applications, the true distribution might not be represented by any model with finite components. Also if the model is complex enough to almost contain the true distribution with finite training samples, we need to consider the case when the model is redundant.

6. Proof of Theorem 3

In this section, we prove Theorem 3. First of all, we derive the variational posterior $r(\theta|X^n)$, $Q(Y^n|X^n)$ and the variational parameter $\overline{\theta}$ for the Gaussian mixture model given by eq.(14).

6.1 Variational Posterior for Gaussian Mixture Model

For the complete-data set $\{X^n, Y^n\} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, let

$$\overline{y}_i^k = \langle y_i^k \rangle_{Q(Y^n|X^n)}, \quad n_k = \sum_{i=1}^n \overline{y}_i^k \quad \text{and} \quad \nu_k = \frac{1}{n_k}\sum_{i=1}^n \overline{y}_i^k x_i,$$

where $y_i^k = 1$ if ith datum x_i is from the kth component, if otherwise, $y_i^k = 0$. The variable n_k is the expected number of times data come from the kth component and ν_k is the mean of them. Note that the variables n_k and ν_k satisfy the constraints $\sum_{k=1}^K n_k = n$ and $\sum_{k=1}^K n_k \nu_k = \sum_{i=1}^n x_i$. From eq.(9) and the respective prior eq.(15) and eq.(16), the variational posterior $r(\theta|X^n) = r(\mathbf{a}|X^n) r(\mu|X^n)$ is obtained as the product of the following two distributions,

$$r(\mathbf{a}|X^n) = \frac{\Gamma(n+K\phi_0)}{\prod_{k=1}^K \Gamma(\bar{a}_k(n+K\phi_0))} \prod_{k=1}^K a_k^{\bar{a}_k(n+K\phi_0)-1},$$

and

$$r(\mu|X^n) = \prod_{k=1}^K \frac{1}{\sqrt{2\pi\bar{\sigma}_k^2}^M} \exp\left(\frac{-\|\mu_k - \bar{\mu}_k\|^2}{2\bar{\sigma}_k^2}\right),$$

where

$$\bar{a}_k = \frac{n_k+\phi_0}{n+K\phi_0}, \quad \bar{\sigma}_k^2 = \frac{1}{n_k+\xi_0}, \quad \text{and} \quad \bar{\mu}_k = \frac{n_k\nu_k + \xi_0\nu_0}{n_k+\xi_0}.$$

From eq.(10), the variational posterior $Q(Y^n|X^n)$ is given by

$$Q(Y^n|X^n) = \frac{1}{C_Q} \prod_{i=1}^n \exp\left[y_i^k \{ \Psi(n_k+\phi_0) - \Psi(n+K\phi_0) - \frac{\|x_i - \bar{\mu}_k\|^2}{2} - \frac{M}{2}\left(\log 2\pi + \frac{1}{n_k+\xi_0}\right)\}\right],$$

where $\Psi(x) = \Gamma'(x)/\Gamma(x)$ is the di-gamma(psi) function and we used

$$\langle \log a_k \rangle_{r(\mathbf{a}|X^n)} = \Psi(n_k+\phi_0) - \Psi(n+K\phi_0).$$

The variational parameter $\bar{\theta}$ is given by $\bar{\theta} = \langle\theta\rangle_{r(\theta|X^n)} = \{\bar{a}_k, \bar{\mu}_k\}_{k=1}^K$. It is noted that $r(\theta|X^n)$ and $Q(Y^n|X^n)$ are parameterized by $\bar{\theta}$ since n_k can be replaced by using $\bar{a}_k = \frac{n_k+\phi_0}{n+K\phi_0}$. Henceforth, we denote $r(\theta|X^n)$ and C_Q as $r(\theta|\bar{\theta})$ and $C_Q(\bar{\theta})$.

6.2 Lemmas

Before proving Theorem 3, we show two lemmas where the two terms $K(r(\theta|\bar{\theta})||\varphi(\theta))$ and $(\log C_Q(\bar{\theta}) + S(X^n))$ in Lemma 2 are respectively evaluated. In the proofs (put in Appendix) of the two lemmas, we use inequalities on the di-gamma function $\Psi(x)$ and the log-gamma function $\log\Gamma(x)$, for $x > 0$ (Alzer, 1997),

$$\frac{1}{2x} < \log x - \Psi(x) < \frac{1}{x}, \tag{20}$$

and

$$0 \le \log\Gamma(x) - \{(x-\frac{1}{2})\log x - x + \frac{1}{2}\log 2\pi\} \le \frac{1}{12x}. \tag{21}$$

The inequalities (20) ensure that substituting $\log x$ for $\Psi(x)$ only contributes additive constant terms to the normalized variational stochastic complexity. The substitution for $\log\Gamma(x)$ is given by eq.(21) as well.

Lemma 5

$$\left| K(r(\theta|\bar{\theta})||\varphi(\theta)) - \{G(\bar{\mathbf{a}}) + \frac{\xi_0}{2}\sum_{k=1}^K \|\bar{\mu}_k - \nu_0\|^2\}\right| \le C,$$

holds where C is a constant and the function $G(\bar{\mathbf{a}})$ *of* $\bar{\mathbf{a}} = \{\bar{a}_k\}_{k=1}^K$ *is defined by*

$$G(\bar{\mathbf{a}}) = \frac{MK+K-1}{2}\log n + \{\frac{M}{2} - (\phi_0 - \frac{1}{2})\}\sum_{k=1}^{K}\log \bar{a}_k.$$

Lemma 6

$$\log C_Q(\bar{\theta}) = \sum_{i=1}^{n}\log\Big[\sum_{k=1}^{K}\frac{1}{\sqrt{2\pi}^M}\exp\{\Psi(n_k+\phi_0) - \Psi(n+K\phi_0) - \frac{\|x_i-\bar{\mu}_k\|^2}{2} - \frac{M}{2}\frac{1}{n_k+\xi_0}\}\Big], \quad (22)$$

and

$$nH_n(\bar{\theta}) - \frac{n}{n+K\phi_0} \le -(\log C_Q(\bar{\theta}) + S(X^n)) \le n\overline{H}_n(\bar{\theta}) - \frac{n}{2(n+K\phi_0)}, \quad (23)$$

where $H_n(\bar{\theta})$ *is given by eq.(3) and* $\overline{H}_n(\bar{\theta})$ *is defined by*

$$\overline{H}_n(\bar{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\log\frac{p(x_i|\theta_0)}{\sum_{k=1}^{K}\frac{\bar{a}_k}{\sqrt{2\pi}^M}\exp\{-\frac{\|x_i-\bar{\mu}_k\|^2}{2} - \frac{M+2}{2(n_k+\min\{\phi_0,\xi_0\})}\}}.$$

6.3 Upper and Lower Bounds

Now from the above lemmas, we prove Theorem 3 by showing the upper bound and the lower bound respectively.

(Proof of Theorem 3)

Proof First we show the upper bound in eq.(17).

From Lemma 2, Lemma 5 and Lemma 6, it follows that

$$\overline{F}_0(X^n) \le \min_{\bar{\theta}} T_n(\bar{\theta}) + C, \quad (24)$$

where

$$T_n(\bar{\theta}) = G(\bar{\mathbf{a}}) + \frac{\xi_0}{2}\sum_{k=1}^{K}\|\bar{\mu}_k - \nu_0\|^2 + n\overline{H}_n(\bar{\theta}).$$

From eq.(24), it is noted that the function values of $T_n(\bar{\theta})$ at specific points of the variational parameter $\bar{\theta}$ give the upper bounds of the normalized variational stochastic complexity $\overline{F}_0(X^n)$. Hence, let us consider following two cases.

(I) :

$$\bar{a}_k = a_k^* \ \ (1 \le k \le K_0 - 1), \ \ \bar{a}_k = a_{K_0}^*/(K - K_0 + 1) \ \ (K_0 \le k \le K),$$

$$\bar{\mu}_k = \mu_k^* \ \ (1 \le k \le K_0 - 1), \ \ \bar{\mu}_k = \mu_{K_0}^* \ \ (K_0 \le k \le K),$$

then $n\overline{H}_n(\bar{\theta}) < \frac{K-K_0+1}{\min_k\{a_k^*\}}$ holds and

$$T_n(\bar{\theta}) < \frac{MK+K-1}{2}\log n + C' + O(\frac{1}{n}),$$

where C' is a constant.

(II) :

$$\overline{a}_k = a_k^* \frac{n+K_0\phi_0}{n+K\phi_0} \quad (1 \leq k \leq K_0), \quad \overline{a}_k = \frac{\phi_0}{n+K\phi_0} \quad (K_0+1 \leq k \leq K),$$

$$\overline{\mu}_k = \mu_k^* \quad (1 \leq k \leq K_0), \quad \overline{\mu}_k = \nu_0 \quad (K_0+1 \leq k \leq K),$$

then $n\overline{H}_n(\overline{\theta}) < (K-K_0)\phi_0 + \frac{1}{\min_k\{a_k^*\}} + O(1/n)$ holds and

$$T_n(\overline{\theta}) < \{(K-K_0)\phi_0 + \frac{MK_0+K_0-1}{2}\}\log n + C'' + O(\frac{1}{n}),$$

where C'' is a constant.

From eq.(24), we obtain the upper bound in eq.(17).

Next we show the lower bound in eq.(17). It follows from Lemma 2, Lemma 5 and Lemma 6,

$$\overline{F}_0(X^n) \geq \min_{\overline{\mathbf{a}}}\{G(\overline{\mathbf{a}})\} + nH_n(\overline{\theta}_{vb}) - C - 1. \tag{25}$$

If $\phi_0 > \frac{M+1}{2}$, then

$$G(\overline{\mathbf{a}}) \geq \frac{MK+K-1}{2}\log n - (\frac{M+1}{2} - \phi_0)K\log K, \tag{26}$$

since Jensen's inequality yields that $\sum_{k=1}^K \log \overline{a}_k \leq K\log(\frac{1}{K}\sum_{k=1}^K \overline{a}_k) = K\log(\frac{1}{K})$.

If $\phi_0 \leq \frac{M+1}{2}$, then

$$G(\overline{\mathbf{a}}) \geq \{(K-1)\phi_0 + \frac{M}{2}\}\log n + (\frac{M+1}{2} - \phi_0)(K-1)\log\phi_0 + O(\frac{1}{n}), \tag{27}$$

since $\overline{a}_k \geq \frac{\phi_0}{n+K\phi_0}$ holds for every k and the constraint $\sum_{k=1}^K \overline{a}_k = 1$ ensures that $|\log \overline{a}_k|$ is bounded by a constant independent of n for at least one index k. From eqs.(25),(26) and (27), we obtain the lower bound in eq.(17).

■

7. Experiments

In order to examine the quality of the theoretical bounds given in Theorem 3, we conducted experiments of variational Bayesian learning for Gaussian mixture models using $M = 1$ and $M = 10$ dimensional synthetic data. A set of models with different number of components ($K = 1,2,3,4,5$) was prepared. We applied the variational Bayesian algorithm to each model using the data set generated from the true distribution with $K_0 = 2$ components. The true distribution was set to a Gaussian mixture model with the parameter $a_1^* = a_2^* = 1/2$, $\mu_1^* = -2/\sqrt{M}\cdot\mathbf{1}$ and $\mu_2^* = 2/\sqrt{M}\cdot\mathbf{1}$ where $\mathbf{1}$ is the M-dimensional vector whose all entries are 1. The hyperparameters were set at $\phi_0 = 1.0$, $\nu_0 = 0$ and $\xi_0 = 1.0$. In order to achieve the minimum in eq.(13), the initial value of the variational parameter $\overline{\theta}$ was set around the true parameter, that is, around $\overline{a}_1 = \overline{a}_2 = 1/2$, $\overline{a}_k = 0$ $(k \geq 3)$, $\overline{\mu}_1 = \mu_1^*$, $\overline{\mu}_2 = \mu_2^*$ and $\overline{\mu}_k = 0$ $(k \geq 3)$. Two sample sets with the size $n = 1000$ and $n = 100$ were prepared. For each data set, the normalized variational stochastic complexity (the inside of the braces in eq.(13)) was

calculated when the variational Bayesian algorithm converged. Denoting the results for respective data sets by $\overline{F}_0(X^{1000})$ and $\overline{F}_0(X^{100})$, we calculated

$$\lambda_{\rm VB} = (\overline{F}_0(X^{1000}) - \overline{F}_0(X^{100}))/\log 10 \tag{28}$$

to estimate the coefficient of the leading term of the normalized variational stochastic complexity $\overline{F}_0(X^n)$. We averaged the values of $\lambda_{\rm VB}$ over 100 draws of sample sets. The results of the averages of $\lambda_{\rm VB}$ and the coefficient $\overline{\lambda}$ given by eq.(18) are presented in Figure 1 against the number K of components for the case of (a)$M = 1$ and (b)$M = 10$. In Figure 1, an upper bound of the coefficient of the Bayesian stochastic complexity and $\lambda_{\rm BIC}$ given by eq.(19) are also plotted for the comparison of variational Bayesian learning with true Bayesian learning in the next section. The variational Bayesian algorithm gave $\lambda_{\rm VB}$ that coincide with the coefficient $\overline{\lambda}$. This implies the upper bound in eq.(17) is tight.

We also calculated the generalization error defined by $K(p(x|\theta_0)||\langle p(x|\theta)\rangle_{r(\theta|X^n)})$, where $\langle p(x|\theta)\rangle_{r(\theta|X^n)}$ is the predictive distribution in variational Bayesian learning. In the case of the Gaussian mixture model, it is given by $\langle p(x|\theta)\rangle_{r(\theta|X^n)} = \sum_{k=1}^{K} \frac{\overline{a}_k}{\sqrt{2\pi(1+\overline{\sigma}_k^2)}^M} \exp(\frac{-\|x-\overline{\mu}_k\|^2}{2(1+\overline{\sigma}_k^2)})$. The generalization error, multiplied by n for scaling purposes, was approximated by

$$\lambda_{\rm G} = \frac{n}{n'} \sum_{i=1}^{n'} \log \frac{p(x_i'|\theta_0)}{\langle p(x_i'|\theta)\rangle_{r(\theta|X^n)}}, \tag{29}$$

with $n' = 10000$ test data $\{x_i'\}_{i=1}^{n'}$ generated from the true distribution. The results of the averages of $\lambda_{\rm G}$ over 100 draws of the data sets with the size $n = 1000$ are also plotted in Figure 1. The results of the averages of $\lambda_{\rm VB}$ and $\lambda_{\rm G}$ showed different behavior. More specifically, $\lambda_{\rm G}$ increased little while $\lambda_{\rm VB}$ grew proportionally to the number K of components. From eq.(6) and eq.(7), $\lambda_{\rm VB}$ and $\lambda_{\rm G}$ should have shown similar behavior if there were the same relation between the average normalized variational stochastic complexity and the average generalization error as in Bayesian learning. These results imply that in variational Bayesian learning, unlike in Bayesian learning, the coefficient of the average generalization error differs from that of the average variational stochastic complexity $E_{X^n}[\overline{F}(X^n)]$.

8. Discussion

In this paper, we showed upper and lower bounds of the variational stochastic complexity of the Gaussian mixture models. We discuss five topics.

8.1 Lower Bound

Let us discuss the lower bound. The lower bound in eq.(17) can be improved to give

$$\overline{F}_0(X^n) \geq \overline{\lambda}\log n + nH_n(\overline{\theta}_{vb}) + C_1, \tag{30}$$

if the consistency of the variational estimator $\overline{\theta}_{vb}$ is proven. Note that the coefficient $\overline{\lambda}$ is the same as that of the upper bound given in Theorem 3. The consistency means that the mixing coefficient $\overline{a}_k$ does not tend to zero for at least K_0 components and they are always used to learn the K_0 true components when the sample size n is sufficiently large. We conjecture that the variational estimator

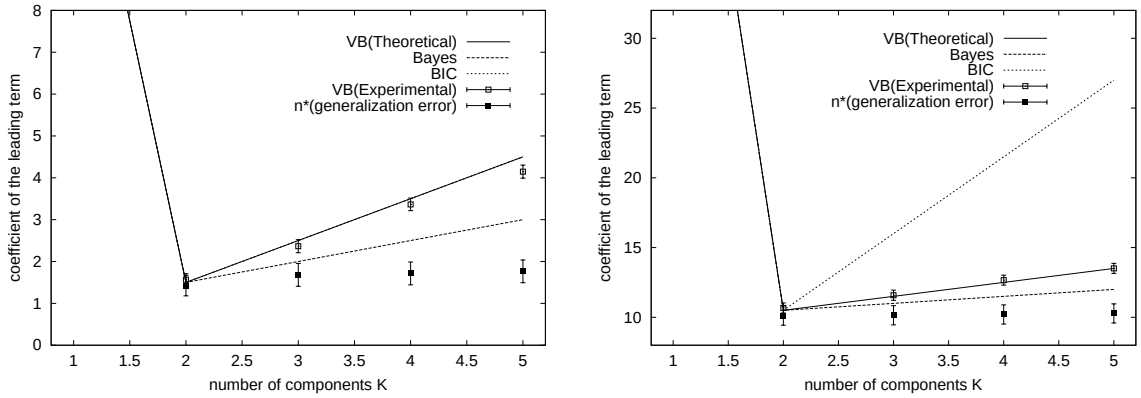


Figure 1: The coefficients of the stochastic complexities for the number K of components with $K_0 = 2$, $\phi_0 = 1$ and (a) $M = 1$, (b) $M = 10$. The solid line is $\overline{\lambda}$ of the variational Bayes eq.(18), the dashed line is the upper bound of λ in true Bayesian learning eq.(32) and the dotted line is $\lambda_{\rm BIC}$ of the BIC eq.(19). The open squares with error bars are the results of the averages of $\lambda_{\rm VB}$ eq.(28) and the full squares with error bars are the results of the averages of $\lambda_{\rm G}$ eq.(29). The error bars show 95% confidence intervals.

is consistent and the inequality (30) holds for the Gaussian mixture model. However, little has been known so far about the behavior of the variational estimator. Analyzing its behavior and investigating the consistency are important undertakings.

Furthermore, on the left hand side of eq.(17), $nH_n(\overline{\theta}_{vb})$ is a kind of training error. If the maximum likelihood estimator exists, it is lower bounded by

$$\min_{\theta} nH_n(\theta) = \min_{\theta} \sum_{i=1}^{n} \log \frac{p(x_i|\theta_0)}{p(x_i|\theta)},$$

which is the (maximum) likelihood ratio statistic with sign inversion. It is known that the likelihood ratio statistics of some non-regular models diverge to infinity as n grows and that the divergence of the likelihood ratio makes the generalization performance worse in the maximum likelihood estimation. In the case of the Gaussian mixture model, it is conjectured that the likelihood ratio diverges in the order of $\log\log n$ (Hartigan, 1985). Although this has not been proved, it suggests that the upper bound in eq.(17) is tight. More specifically, if eq.(30) holds and the order of divergence of the likelihood ratio is smaller than $\log n$, that is, $E_{X^n}[\min_\theta nH_n(\theta)] = o(\log n)$, then it immediately follows from Corollary 4 that

$$E_{X^n}[\overline{F}_0(X^n)]/\log n \to \overline{\lambda} \quad (n \to \infty). \tag{31}$$

This was suggested also by the experimental results presented in the previous section.

8.2 Comparison to Bayesian Learning

We compare the normalized variational stochastic complexity shown in Theorem 3 with the one in true Bayesian learning assuming eq.(31) holds. The Bayesian stochastic complexities of several

non-regular models have been clarified in some recent studies. For the Gaussian mixture model in particular, the following upper bound on the coefficient of the average normalized Bayesian stochastic complexity $E_{X^n}[F_0(X^n)]$ described as eq.(6) is known (Watanabe et al., 2004),

$$\lambda \leq (MK_0 + K - 1)/2, \tag{32}$$

under the same condition about the true distribution and the model as the condition (i) described in Section 5 and certain conditions about the prior distribution. Since these conditions about the prior are satisfied by putting $\phi_0 = 1$ in the condition (ii) of Theorem 3, we can compare the stochastic complexities in this case. Putting $\phi_0 = 1$ in eq.(18), we have

$$\overline{\lambda} = K - K_0 + (MK_0 + K_0 - 1)/2. \tag{33}$$

Let us compare this $\overline{\lambda}$ of variational Bayesian learning to λ in eq.(32) of true Bayesian learning.

For any M,

$$\overline{\lambda} - \lambda \geq (K - K_0)/2$$

holds. This implies that the more redundant components the model has, the more variational Bayesian learning differs from true Bayesian learning. However the difference $(K - K_0)/2$ is rather small since it is independent of the dimension M of the input space. This implies the variational posterior is close to the true Bayesian posterior. Moreover, it is noted that when $M = 1$, that is, the input is one-dimensional, $2\overline{\lambda}$ is equal to $2K - 1$ that is the number of the parameters of the model. Hence the Bayesian information criterion (BIC) (Schwarz, 1978) and the minimum description length (MDL) (Rissanen, 1986) correspond to $\overline{\lambda} \log n$ when $M = 1$.

Figure 1 shows the coefficients $\overline{\lambda}$, λ_{BIC} and the upper bound of the coefficient λ of the Bayesian stochastic complexity with respect to the number K of components for the case when $K_0 = 2$, $\phi_0 = 1$ and (a) $M = 1$ and (b) $M = 10$. In (a) of Figure 1, $\overline{\lambda}$ (solid line) and λ_{BIC} (dotted line) coincide. It is noted that $\overline{\lambda}$ of variational Bayesian learning eq.(33) relatively approaches the upper bound in Bayesian learning eq.(32) and becomes far smaller than that of BIC eq.(19) as the dimension M becomes larger.

8.3 Stochastic Complexity and Generalization

We have discussed how much the variational posterior differs from the true Bayesian posterior by comparing the stochastic complexities. In variational Bayesian learning, there is no apparent relationship between the average variational stochastic complexity and the average generalization error unlike in Bayesian learning where their leading terms are given by the same coefficient λ as in eq.(6) and eq.(7). This was also observed experimentally by the different behavior of λ_{VB} and λ_{G} in the previous section. Hence, assessing the generalization performance of the Gaussian mixture model in variational Bayesian learning is an important issue to be addressed. The term $(\log C_Q(\overline{\theta}) + S(X^n))$ in Lemma 6 may diverge to infinity as the likelihood ratio statistic in the maximum likelihood method as mentioned above. It would be important to clarify how this term affects the generalization performance in variational Bayesian learning.

8.4 Effect of Hyperparameters

Let us discuss the effects of the hyperparameters. From Theorem 3, only the hyperparameter ϕ_0 affects the leading term of the normalized variational stochastic complexity $\overline{F}_0(X^n)$ and the other

hyperparameters ξ_0 and ν_0 affect only the lower order terms. This is due to the influence of the hyperparameters on the prior probability density around the true parameters. Consider the case when $K_0 < K$. In this case, for a parameter that gives the true distribution, either of the followings holds, $a_k = 0$ for some k or $\mu_i = \mu_j$ for some pair (i, j). The prior distribution $\varphi(\mathbf{a})$ given by eq.(15) can drastically change the probability density around the points where $a_k = 0$ for some k by changing the hyperparameter ϕ_0 while the prior distribution $\varphi(\mu)$ given by eq.(16) always takes positive values for any values of the hyperparameters ξ_0 and ν_0.

We also point out that Theorem 3 shows how the hyperparameter ϕ_0 influence variational Bayesian learning. The coefficients $\underline{\lambda}$ and $\overline{\lambda}$ in eq.(18) are divided into two cases. These cases correspond to whether $\phi_0 \leq (M+1)/2$ holds, indicating that the influence of the hyperparameter ϕ_0 in the prior $\varphi(\mathbf{a})$ appears depending on the dimension M of the input space. More specifically, only when $\phi_0 \leq (M+1)/2$, the prior distribution reduces redundant components; otherwise it uses all the components.

8.5 Applications of the Bounds

Finally, let us give examples of how to use the theoretical bounds given in Theorem 3 and discuss issues to be addressed.

Comparing the theoretical bounds in eq.(17) with experimental results, one can investigate the properties of the actual iterative algorithm in variational Bayesian learning. Although the actual iterative algorithm gives the variational posterior that satisfies eq.(9) and eq.(10), it may converge to local minima of the functional $\overline{F}[q]$. Remember that eq.(9) and eq.(10) are just a necessary condition for $\overline{F}[q]$ to be minimized. One can examine experimentally whether the algorithm converges to the optimal variational posterior that minimizes the functional instead of local minima by comparing the experimental results with the theoretical bounds. Moreover, the theoretical bounds would enable us to compare the accuracy of variational Bayesian learning with that of the Laplace approximation or the MCMC method. However, in order to make such comparisons more accurately, one will need not only the leading term but also the lower order terms of the asymptotic form of the variational stochastic complexity. Giving the more accurate asymptotic form is important for such comparisons.

The Gaussian mixture model is included in general exponential family models with hidden variables (Sato, 2001) and furthermore, in general graphical models to which the variational Bayesian framework can be applied (Attias, 1999). Analyzing the variational stochastic complexities in the more general cases would be an important undertaking.

Furthermore, as mentioned in Section 4, the variational stochastic complexity $\overline{F}(X^n)$ is used as a criterion for model selection in variational Bayesian learning. Theorem 3 shows how accurately one can estimate the Bayesian stochastic complexity $F(X^n)$, the negative log of the Bayesian evidence, by its upper bound $\overline{F}(X^n)$. By the above comparison to Bayesian learning, it is expected that $\overline{F}(X^n)$ provides a rather good approximation to $F(X^n)$. This gives a theoretical justification for its use in model selection. Our result is important for developing effective model selection methods using $\overline{F}(X^n)$.

9. Conclusion

In this paper, we derived upper and lower bounds of the variational stochastic complexity of the Gaussian mixture models. Using the derived bounds, we discussed the influence of the hyperparameters and the accuracy of variational Bayesian learning as an approximation of true Bayesian

learning. These bounds can be used for evaluation and optimization of learning algorithms based on the variational Bayesian approximation.

Acknowledgments

The authors would like to thank Dr.Masa-aki Sato, Dr.Shin Ishii and Dr.Kenji Fukumizu for their helpful comments. This work was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for JSPS Fellows 16-4637 and for Scientific Research 15500130, 2005. An early version of this paper has been presented at IEEE conference on Cybernetics and Intelligent Systems (Watanabe and Watanabe, 2004).

Appendix A.

Proof of Lemma 2

Proof From the restriction of the variational Bayesian approximation eq.(8), $\overline{F}(X^n)$ can be divided into two terms,

$$\overline{F}(X^n) = \min_{r,Q}\Big[\langle \log \frac{r(\theta|X^n)}{\varphi(\theta)}\rangle_{r(\theta|X^n)} + \langle \log \frac{Q(Y^n|X^n)}{p(X^n,Y^n|\theta)}\rangle_{r(\theta|X^n)Q(Y^n|X^n)}\Big].$$

Since the optimal variational posteriors satisfy eq.(9) and eq.(10), if the variational posterior $Q(Y^n|X^n)$ is optimized, then

$$\langle \log \frac{Q(Y^n|X^n)}{p(X^n,Y^n|\theta)}\rangle_{r(\theta|X^n)Q(Y^n|X^n)} = -\log C_Q$$

holds. Thus we obtain eq.(12). ■

Proof of Lemma 5

Proof Calculating the Kullback information between the posterior and the prior, we obtain

$$K(r(\mathbf{a}|\bar{\mathbf{a}})||\varphi(\mathbf{a})) = \sum_{k=1}^{K} h(n_k) - n\Psi(n+K\phi_0) + \log\Gamma(n+K\phi_0) + \log\frac{\Gamma(\phi_0)^K}{\Gamma(K\phi_0)}, \tag{34}$$

where we use the notation $h(x) = x\Psi(x+\phi_0) - \log\Gamma(x+\phi_0)$. Similarly,

$$K(r(\mu|\bar{\mu})||\varphi(\mu)) = \sum_{k=1}^{K}\frac{M}{2}\log\frac{n_k+\xi_0}{\xi_0} - \frac{KM}{2} + \frac{1}{2}\xi_0\sum_{k=1}^{K}\{\frac{M}{n_k+\xi_0} + \|\bar{\mu}_k-\nu_0\|^2\}. \tag{35}$$

By using inequalities (20) and (21), we obtain

$$h(x) = -(\phi_0 - \frac{1}{2})\log(x+\phi_0) + x + O(1).$$

Thus we have, from eqs.(34),(35) and $K(r(\theta|\bar{\theta})||\varphi(\theta)) = K(r(\mathbf{a}|\bar{\mathbf{a}})||\varphi(\mathbf{a})) + K(r(\mu|\bar{\mu})||\varphi(\mu))$,

$$\Big|K(r(\theta|\bar{\theta})||\varphi(\theta)) - \Big\{G(\bar{\mathbf{a}}) + \frac{\xi_0}{2}\sum_{k=1}^{K}\|\bar{\mu}_k-\nu_0\|^2\Big\}\Big| \le C,$$

where C is a constant since $\frac{1}{n+\xi_0} < \frac{1}{n_k+\xi_0} < \frac{1}{\xi_0}$. ■

Proof of Lemma 6

Proof

$$\begin{aligned} C_Q(\bar{\theta}) &= \prod_{i=1}^{n} \sum_{y_i} \exp \langle \log p(x_i, y_i|\theta) \rangle_{r(\theta|\bar{\theta})} \\ &= \prod_{i=1}^{n} \sum_{k=1}^{K} \frac{1}{\sqrt{2\pi}^M} \exp\{\Psi(n_k+\phi_0) - \Psi(n+K\phi_0) - \frac{\|x_i-\bar{\mu}_k\|^2}{2} - \frac{M}{2}\frac{1}{n_k+\xi_0}\}. \end{aligned} \tag{36}$$

Thus we have eq.(22).

Using again the inequalities (20), we obtain

$$-\log C_Q(\bar{\theta}) \leq -\sum_{i=1}^{n} \log\Big[\sum_{k=1}^{K} \frac{\bar{a}_k}{\sqrt{2\pi}^M} \exp\{-\frac{\|x_i-\bar{\mu}_k\|^2}{2} - \frac{M+2}{2(n_k+\min\{\phi_0,\xi_0\})}\}\Big] - \frac{n}{2(n+K\phi_0)},$$

and

$$-\log C_Q(\bar{\theta}) \geq -\sum_{i=1}^{n} \log\Big[\sum_{k=1}^{K} \frac{\bar{a}_k}{\sqrt{2\pi}^M} \exp\{-\frac{\|x_i-\bar{\mu}_k\|^2}{2}\}\Big] - \frac{n}{n+K\phi_0},$$

which give the upper and lower bounds in eq.(23) respectively. ■

References

H. Akaike. Likelihood and bayes procedure. In *Bayesian Statistics*, pages 143–166, Valencia, Spain, 1980. (Bernald J.M. eds.), University Press.

H. Alzer. On some inequalities for the Gamma and Psi functions. *Mathematics of computation*, 66 (217):373–389, 1997.

H. Attias. Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of Uncertainty in Artificial Intelligence(UAI'99)*, 1999.

M. J. Beal. *Variational Algorithms for approximate Bayesian inference*. PhD thesis, University College London, 2003.

D. Dacunha-Castelle and E. Gassiat. Testing in locally conic models, and application to mixture models. *Probability and Statistics*, 1:285–317, 1997.

Z. Ghahramani and M. J. Beal. Graphical models and variational methods. *Advanced Mean Field Methods – Theory and Practice, eds. D. Saad and M. Opper, MIT Press*, 2000.

J. A. Hartigan. A failure of likelihood asymptotics for normal mixtures. In *Proceedings of the Berkeley Conference in Honor of J.Neyman and J.Kiefer*, pages 807–810, 1985.

E. Levin, N. Tishby, and S. A. Solla. A statistical approaches to learning and generalization in layered neural networks. *Proc. of IEEE*, 78(10):1568–1674, 1990.

D. J. Mackay. Bayesian interpolation. *Neural Computation*, 4(2):415–447, 1992.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14(3):1080–1100, 1986.

M. Sato. Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681, 2001.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

K. Watanabe and S. Watanabe. Lower bounds of stochastic complexities in variational bayes learning of gaussian mixture models. In *Proceedings of IEEE conference on Cybernetics and Intelligent Systems (CIS04)*, pages 99–104, Singapore, 2004.

S. Watanabe. Algebraic analysis for non-identifiable learning machines. *Neural Computation*, 13 (4):899–933, 2001.

S. Watanabe, K. Yamazaki, and M. Aoyagi. Kullback information of normal mixture is not an analytic function. *Technical Report of IEICE (in Japanese)*, NC2004-50:41–46, 2004.

K. Yamazaki and S. Watanabe. Singularities in mixture models and upper bounds of stochastic complexity. *International Journal of Neural Networks*, 16:1023–1038, 2003a.

K. Yamazaki and S. Watanabe. Stochastic complexity of bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence(UAI'03)*, 2003b.

Journal of Machine Learning Research 7 (2006) 645–664 Submitted 7/05; Revised 12/05; Published 5/06

Pattern Recognition for Conditionally Independent Data

Daniil Ryabko DANIIL@RYABKO.NET
IDSIA
Galleria 2
6928 Manno-Lugano, Switzerland
and
Computer Learning Research Centre
Royal Holloway, University of London
Egham TW20 0EX, UK

Editor: Peter Bartlett

Abstract

In this work we consider the task of relaxing the i.i.d. assumption in pattern recognition (or classification), aiming to make existing learning algorithms applicable to a wider range of tasks. Pattern recognition is guessing a discrete label of some object based on a set of given examples (pairs of objects and labels). We consider the case of deterministically defined labels. Traditionally, this task is studied under the assumption that examples are independent and identically distributed. However, it turns out that many results of pattern recognition theory carry over a weaker assumption. Namely, under the assumption of conditional independence and identical distribution of objects, while the only assumption on the distribution of labels is that the rate of occurrence of each label should be above some positive threshold.

We find a broad class of learning algorithms for which estimations of the probability of the classification error achieved under the classical i.i.d. assumption can be generalized to the similar estimates for case of conditionally i.i.d. examples.

1. Introduction

Pattern recognition (or classification) is, informally, the following task. There is a finite number of classes of some complex objects. A predictor is learning to classify the objects, based only on examples of labelled objects. The formal model of the task used most widely is described, for example, in (Vapnik, 1998), and can be briefly introduced as follows (we will later refer to it as "the i.i.d. model"). The objects $x \in \mathbf{X}$ are drawn independently and identically distributed (i.i.d.) according to some unknown (but fixed) probability distribution $P(x)$. The labels $y \in \mathbf{Y}$ are given for each object according to some (also unknown but fixed) function[1] $\eta(x)$. The space $\mathbf{Y}$ of labels is assumed to be finite (often binary). The task is to construct the best predictor for the labels, based on the data observed, i.e. actually to "learn" $\eta(x)$.

This task is usually considered in either of the following two settings. In the off-line setting a (finite) set of examples is divided into two finite subsets, the training set and the testing set. A predictor is constructed based on the first set and then is used to classify the objects from the second. In the online setting a predictor starts by classifying the first object with zero knowledge; then it is

1. Often (e.g. in (Vapnik, 1998)) a more general situation is considered, the labels are drawn according to some probability distribution $P(y|x)$, i.e. each object can have more than one possible label.

given the correct label and (having "learned" this information) proceeds with classifying the second object, the correct second label is given, and so on.

Weakness of the model: an example. Many algorithms were developed for solving pattern recognition tasks (see Devroye, Györfi, Lugosi, 1996; Vapnik, 1998; Kearns, Vazirani, 1994, for the most widely used methods). However, the i.i.d. assumption, which is central in the model, is too tight for many applications. It turns out that it is also too tight for a wide range of methods developed under the assumptions of the model: they work nearly as well under weaker conditions.

First consider the following example, which provides intuition for the probabilistic model we introduce. Suppose we are trying to recognise a printed or hand-written text. Obviously, letters in the text are dependent (for instance, we strongly expect to meet "u" after "q"). Observe also that a written text is not Markovian and, moreover, can exhibit arbitrarily long range dependencies. This seemingly implies that pattern recognition methods can not be applied to this task, which is one of their classical applications.

However, a sequence of images which forms a written text has several properties, which in fact will be shown to be sufficient for learning. First, the object-label dependence does not change in time. That is, an image of a letter which in the beginning of the text means, say, "a", to the end of the text will not be interpreted as, say, "e". Moreover, if we extract from the original sequence all letters labelled with (for instance) "a", the resulting sequence (of images of "a") will be i.i.d. Finally, the rate of occurrence of each label keeps above some positive threshold. In our example, we expect the rate of occurrence of each letter to be, say, somewhere between 1% and 99% of all letters, with some feasible probability (depending on the size of the text).

Thus, given labels, objects are independent. This holds exactly for a typewritten text. For a text on a journal page this condition is sometimes violated because of such image-image dependencies as ligatures (like " ff "). In a hand-written text different pairs of letters are connected differently and so the condition does not hold, but still seems more adequate than the pure i.i.d. condition.

Conditional i.i.d. model. These intuitive ideas lead us to the following model (to which we refer as "the conditional model"). The labels $y \in \mathbf{Y}$ are drawn according to some unknown (but fixed) distribution over the set of all infinite sequences of labels. There can be any type of dependence between labels; moreover, we can assume that we are dealing with any (fixed) combinatorial sequence of labels. However, in this sequence the rate of occurrence of each label should keep above some positive threshold. For each label y the corresponding object $x \in \mathbf{X}$ is generated according to some (unknown but fixed) probability distribution $P(x|y)$. All the rest is as in the i.i.d. model.

The main difference from the i.i.d. model is that in the conditional model we made the distribution of labels primal; having done that we can relax the requirement of independence of objects to the conditional independence.

Results. The main result of the paper is *not* in constructing an algorithm for the proposed model. Rather, we show that any reasonable already known algorithm designed to work in the i.i.d. setting also works under the strictly weaker conditionally i.i.d. assumption. An implication is that the i.i.d. assumption for pattern recognition is, to a considerable extent, redundant.

Moreover, we provide a tool for obtaining estimations of probability of error of a predictor in the conditional model from estimations of the probability of error in the i.i.d. model. The general theorems about extending results concerning performance of a predictor to the conditional model are illustrated on two classes of predictors.

First, we extend weak consistency results concerning partitioning and nearest neighbour estimates from the i.i.d. model to the conditional model.

Second, we use some results of Vapnik-Chervonenkis theory to estimate performance in the conditional model (on finite amount of data) of predictors minimizing empirical risk, and also obtain some strong consistency results.

These results are obtained as applications of the following general statement. The only assumption on a predictor under which it works in the conditional model as well as in the i.i.d. model is what we call *tolerance to data*: in any large data set there is no small subset which strongly changes the probability of error. This property should also hold with respect to permutations. This assumption on a predictor should be valid in the i.i.d. model. Thus, the results achieved in the i.i.d. model can be extended to the conditional model; this concerns distribution–free results as well as distribution–specific, results on the performance on finite samples as well as asymptotic results.

Further examples. As another example of pattern recognition task which does not comply with the i.i.d. (or Markov) assumption, but is more adequately modelled by the conditional i.i.d. assumption, consider the problem of medical diagnostics. The problem is to diagnose a certain disease based on the set of symptoms; for simplicity, consider binary labels (ill versus not ill). Since many diseases have yearly or other dynamics (e.g. epidemics), the sequence of sets of symptoms of patients entering a clinic can not be considered i.i.d. However, the sequence of sets of symptoms of ill patients does not reflect such dynamics, and can be considered close to i.i.d. In other words, we expect the distribution of symptoms to be determined only by the fact that the patient is ill or that (s)he is not. Note however, that there are certain types of dependencies between sets of symptoms which can violate our condition, for example, if a family comes for diagnostics together; yet the conditional i.i.d. model seems to be more adequate here than just i.i.d. or than Markov condition, since it allows for more dependencies present in the problem.

The same argument applies for any task which would be i.i.d. if it was not for the fluctuations of the class probability, such as an example from (Duda, Hart, Stork, 2001) of classifying fish species by a photographic image: one can imagine that at different times and in different areas the proportion of species among the fish caught is different.

It should also be mentioned that in such popular practical tasks as speech recognition the label-label dependencies, which we show to be tolerated by pattern recognition methods, can be and actually are exploited. Thus, pattern recognition methods are used in conjunction with sequence prediction algorithms, and here our results can be considered a further theoretical justification of the use of the pattern recognition component.

Related work. Various approaches to relaxing the i.i.d. assumption in learning tasks have been proposed in the literature. Thus, in (Kulkarni, Posner , Sandilya, 2002; Kulkarni, Posner, 1995) the authors study the nearest neighbour and kernel estimators for the task of regression estimation with continuous regression function, under the assumption that labels are conditionally independent given their objects, while objects form any individual sequence. The probabilistic assumption is weaker than ours but continuity of regression function holds only in trivial cases of the classification task we consider. A similar approach is taken in (Morvai, Kulkarni, Nobel, 1999), where a regression estimation scheme is proposed which is consistent for any individual stable sequence of object-label pairs (no probabilistic assumptions), assuming that there is a known upper bound on the variation of regression function.

There are also several approaches in which different types of assumptions on the joint distribution of objects and labels are made; then the authors construct a predictor or a class of predictors, to work well under the assumptions made. Thus, in (Gamarnik, 2003) and (Aldous, Vazirani, 1990) a generalisation of PAC approach to Markov chains with finite or countable state space is presented.

The estimates of probability of error are constructed for this cases, under the assumption that the optimal rule generating examples belongs to a pre-specified class of decision rules. There is also a track of research on prediction under the assumption that the distribution generating examples is stationary or stationary and ergodic. The basic difference from our learning task, apart from different probabilistic assumption, is in that we are only concerned with object-label dependence, while in predicting ergodic sequences it is label-label (time series) dependence that is of primary interest. On this task see (Ryabko, 1988; Algoet, 1999; Morvai, Yakowitz, Algoet, 1997; Nobel, 1999) and references therein. Observe also that none of these probabilistic concepts (Markov assumption, stationarity, ergodicity) is comparable with our conditional i.i.d. assumption, in the sense that none of them is either weaker nor stronger than the conditional i.i.d. assumption.

Another approach is taken in (Helmbold, Long, 1991; Bartlett, Ben-David, Kulkarni, 1996) where the PAC model is generalised to allow concepts changing over time. Here the methodology is proposed to track time series dependencies, that is, the authors find some classes of dependencies which can be exploited for learning. Again the difference with our approach is that we try to find a (broad) class of problems where the time series dependence can be ignored by any reasonable pattern recognition method rather than constructing methods to use some specific dependencies of this kind.

2. Definitions and General Results

Consider a sequence of *examples* $(x_1,y_1),(x_2,y_2),\ldots$; each example $z_i := (x_i,y_i)$ consists of an *object* $x_i \in \mathbf{X}$ and a *label* $y_i := \eta(x_i) \in \mathbf{Y}$, where $\mathbf{X}$ is a measurable space called an *object space*, $\mathbf{Y} := \{0,1\}$ is called a *label space* and $\eta : \mathbf{X} \to \mathbf{Y}$ is some deterministic function. For simplicity we made the assumption that the space $\mathbf{Y}$ is binary, but all results easily extend to the case of any finite space $\mathbf{Y}$. The notation $\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$ is used for the measurable space of examples. Objects are drawn according to some probability distribution $\mathbf{P}$ on $\mathbf{X}^\infty$ (and labels are defined by η). Thus we consider only the case of deterministically defined labels (that is, the noise-free model); in section 5 we discuss possible generalisations.

The notation $\mathbf{P}$ is used for distributions on $\mathbf{X}^\infty$ while the symbol P is reserved for distributions on $\mathbf{X}$. In the latter case P^∞ denotes the i.i.d. distribution on $\mathbf{X}^\infty$ generated by P. Correspondingly we will use symbols $\mathbf{E}$, E and $\mathbf{E}^\infty$ for expectations over spaces $\mathbf{X}^\infty$ and $\mathbf{X}$. Letters x,y,z (with indices) will be used for elements of spaces $\mathbf{X},\mathbf{Y},\mathbf{Z}$ correspondingly, while letters X,Y,Z are reserved for random variables on these spaces.

The traditional assumption about the distribution $\mathbf{P}$ generating objects is that examples are independently and identically distributed (i.i.d.) according to some distribution P on $\mathbf{X}$ (i.e. $\mathbf{P} = P^\infty$).

Here we replace this assumption with the following two conditions.

First, for any $n \in \mathbb{N}$ and for any measurable set $A \subset \mathbf{X}$

$$\mathbf{P}(X_n \in A \mid Y_n, X_1, Y_1, \ldots, X_{n-1}, Y_{n-1}) = \mathbf{P}(X_n \in A \mid Y_n) \tag{1}$$

(i.e. some versions of conditional probabilities coincide). This condition looks very much like Markov condition which requires that each object depends on the past only through its immediate predecessor. The condition (1) says that each object depends on the past only through its label.

Second, for any $y \in \mathbf{Y}$, for any $n_1,n_2 \in \mathbb{N}$ and for any measurable set $A \subset \mathbf{X}$

$$\mathbf{P}(X_{n_1} \in A \mid Y_{n_1} = y) = \mathbf{P}(X_{n_2} \in A \mid Y_{n_2} = y) \tag{2}$$

(i.e. the process is uniform in time; (1) allows dependence in n).

Note that the first condition means that objects are conditionally independent given labels (on conditional independence see Dawid, 1979). Under the conditions (1) and (2) we say that *objects are conditionally independent and identically distributed* (conditionally i.i.d.).

For each $y \in \mathbf{Y}$ denote the distribution $\mathbf{P}(X_n \mid Y_n = y)$ by P_y (it does not depend on n by (2)). Clearly, the distributions P_0 and P_1 define some distributions P on $\mathbf{X}$ up to a parameter $p \in [0,1]$. That is, $P_p(A) = pP_1(A) + (1-p)P_0(A)$ for any measurable set $A \subset \mathbf{X}$ and for each $p \in [0,1]$. Thus with each distribution $\mathbf{P}$ satisfying the assumptions (1) and (2) we will associate a family of distributions P_p, $p \in [0,1]$.

The assumptions of the conditional model can be also interpreted as follows. Assume that we have some individual sequence $(y_n)_{n\in\mathbb{N}}$ of labels and two probability distributions P_0 and P_1 on $\mathbf{X}$, such that there exists sets X_0 and X_1 in $\mathbf{X}$ such that $P_1(X_1) = P_0(X_0) = 1$ and $P_0(X_1) = P_1(X_0) = 0$ (i.e. X_0 and X_1 define some function η). Each example $x_n \in \mathbf{X}$ is drawn according to the distribution P_{y_n}; examples are drawn independently of each other.

A *predictor* is a measurable function $\Gamma_n := \Gamma(x_1, y_1, \dots, x_n, y_n, x_{n+1})$ taking values in $\mathbf{Y}$ (more formally, a family of functions indexed by n).

The probability of error of a predictor Γ on each step n is defined as

$$\mathrm{err}_n(\Gamma, \mathbf{P}, z_1, \dots, z_n) := \mathbf{P}\big\{(x,y) \in \mathbf{Z} : y \neq \Gamma_n(z_1, \dots, z_n, x)\big\}$$

(z_i, $1 \le i \le n$ are fixed and the probability is taken over z_{n+1}). We will sometimes omit some of the arguments of err_n when it can cause no confusion; in particular, we will often use a short notation $\mathbf{P}(\mathrm{err}_n(\Gamma, Z_1, \dots, Z_n) > \varepsilon)$ and an even shorter one $\mathbf{P}(\mathrm{err}_n(\Gamma) > \varepsilon)$ in place of

$$\mathbf{P}\big\{z_1, \dots, z_n : \mathrm{err}_n(\Gamma, \mathbf{P}, z_1, \dots, z_n) > \varepsilon\big\}.$$

For a pair of distributions P_0 and P_1 and any $\delta \in (0, 1/2)$ define

$$\bigtriangledown_\delta(P_0, P_1, n, \varepsilon) := \sup_{p \in [\delta, 1-\delta]} P_p^\infty(\mathrm{err}_n(\Gamma) > \varepsilon), \tag{3}$$

that is, we consider the supremum of the probability of error over all class label probabilities.

For a predictor Γ and a distribution P on $\mathbf{X}$ define

$$\Delta(P, n, z_1, \dots, z_n) := \max_{j \le \varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\}} |\,\mathrm{err}_n(\Gamma, P^\infty, z_1, \dots, z_n) - \mathrm{err}_{n-j}(\Gamma, P^\infty, z_{\pi(1)}, \dots, z_{\pi(n-j)})|.$$

Define the *tolerance to data* of Γ as

$$\Delta(P, n, \varepsilon) := P^n\big(\Delta(P, n, Z_1, \dots, Z_n) > \varepsilon\big) \tag{4}$$

for any $n \in \mathbb{N}$, any $\varepsilon > 0$ and $\varkappa_n := \sqrt{n \log n}$ (see the end of Section 5 for the discussion of the choice of the constants $\varkappa_n$). Furthermore, for a pair of distributions P_0 and P_1 and any $\delta \in (0, 1/2)$ define

$$\Delta_\delta(P_0, P_1, n, \varepsilon) := \sup_{p \in [\delta, 1-\delta]} \Delta(P_p, n, \varepsilon). \tag{5}$$

Tolerance to data means, in effect, that in any typical large portion of data there is no small portion that changes strongly the probability of error. This property should also hold with respect to permutations.

We will also use another version of tolerance to data, in which instead of removing some examples we replace them with an arbitrary sample $z'_j,\dots,z'_n$ consistent with η:

$$\bar{\Delta}(P,z_1,\dots,z_n) := \sup_{j<\varkappa_n;\pi:\{1,\dots,n\}\to\{1,\dots,n\};z'_{n-j},\dots,z'_n} |\operatorname{err}_n(\Gamma,P^\infty,z_1,\dots,z_n)-\operatorname{err}_n(\Gamma,P^\infty,\zeta_1,\dots,\zeta_n)|,$$

where $\zeta_{\pi(i)} := z_{\pi(i)}$ if $i<n-j$ and $\zeta_{\pi(i)} := z'_i$ otherwise; the maximum is taken over all z'_i, $n-j<i\le n$ consistent with η. Define

$$\bar{\Delta}(P,n,\varepsilon) := P^n\big(\bar{\Delta}(P,n,Z_1,\dots,Z_n)>\varepsilon\big)$$

and

$$\bar{\Delta}_\delta(P_0,P_1,n,\varepsilon) := \sup_{p\in[\delta,1-\delta]}\bar{\Delta}(P_p,n,\varepsilon).$$

The same notational convention will be applied to Δ and $\bar{\Delta}$ as to err_n.

Various notions similar to tolerance to data have been studied in literature. Perhaps first they appeared in connection with deleted or condensed estimates (see e.g. Rogers, Wagner, 1988), and were later called stability (see Bousquet, Elisseeff, 2002; Kearns, Ron, 1999, for present studies of different kinds of stability, and for extensive overviews). Naturally, such notions arise when there is a need to study the behaviour of a predictor when some of the training examples are removed. These notions are much similar to what we call tolerance to data, only we are interested in the maximal deviation of probability of error while usually it is the average or minimal deviations that are estimated.

A predictor developed to work in the off-line setting should be, loosely speaking, tolerant to small changes in a training sample. The next theorem shows under which conditions this property of a predictor can be utilized.

Theorem 1 *Suppose that a distribution* $\mathbf{P}$ *generating examples is such that the objects are conditionally i.i.d., i.e.* $\mathbf{P}$ *satisfies (1) and (2). Fix some* $\delta\in(0,1/2]$, *let* $p(n) := \frac{1}{n}\#\{i\le n : Y_i=1\}$ *and* $C_n := \mathbf{P}(\delta\le p(n)\le 1-\delta)$ *for each* $n\in\mathbb{N}$. *Let also* $\alpha_n := \frac{1}{1-1/\sqrt{n}}$. *For any predictor* Γ *and any* $\varepsilon>0$ *we have*

$$\mathbf{P}(\operatorname{err}_n(\Gamma)>\varepsilon)\le C_n^{-1}\alpha_n\big(\bigtriangledown_\delta(P_0,P_1,n+\varkappa_n,\delta\varepsilon/2) + \Delta_\delta(P_0,P_1,n+\varkappa_n,\delta\varepsilon/2)\big)+(1-C_n), \quad (6)$$

and

$$\mathbf{P}(\operatorname{err}_n(\Gamma)>\varepsilon)\le C_n^{-1}\alpha_n\big(\bigtriangledown_\delta(P_0,P_1,n,\delta\varepsilon/2)+\bar{\Delta}_\delta(P_0,P_1,n,\delta\varepsilon/2)\big)+(1-C_n). \quad (7)$$

The theorem says that if we know with some confidence C_n that the rate of occurrence of each label is not less than some (small) δ, and have some bounds on the error rate and tolerance to data of a predictor in the i.i.d. model, then we can obtain bounds on its error rate in the conditional model.

The proofs for this section can be found in Appendix A. The intuition behind the proof of the theorem is as follows. First we fix some individual sample of n labels (without objects) and consider the behaviour of the predictor conditional on this sample. Fixing the labels allows us to pass from the conditional i.i.d. case to i.i.d. and to use error estimates for this case. Then, using tolerance

to data, we compare the behaviour of the predictor on any two different, but typical for a certain i.i.d. distribution, samples of labels. This allows us to estimate the probability of error on any (typical) sample and so to pass back to the conditional i.i.d. case.

Thus we have a tool for estimating the performance of a predictor on each finite step n. In Section 4 we will show how this result can be applied to predictors minimizing empirical risk. However, if we are only interested in asymptotic results the formulations can be somewhat simplified.

Consider the following asymptotic condition on the frequencies of labels. Define $p(n) := \frac{1}{n}\#\{i \le n : Y_i = 1\}$. We say that the *rates of occurrence of labels are bounded from below* if there exist such δ, $0 < \delta < 1/2$ that

$$\lim_{n\to\infty} \mathbf{P}(p(n) \in [\delta, 1-\delta]) = 1. \tag{8}$$

As the condition (8) means $C_n \to 1$ we can derive from Theorem 1 the following corollary.

Corollary 2 *Suppose that a distribution* $\mathbf{P}$ *satisfies (1), (2), and (8) for some* $\delta \in (0, 1/2]$. *Let* Γ *be such a predictor that*

$$\lim_{n\to\infty} \bigtriangledown_\delta(P_0, P_1, n, \varepsilon) = 0 \tag{9}$$

and either

$$\lim_{n\to\infty} \Delta_\delta(P_0, P_1, n, \varepsilon) = 0 \tag{10}$$

or

$$\lim_{n\to\infty} \bar{\Delta}_\delta(P_0, P_1, n, \varepsilon) = 0 \tag{11}$$

for any $\varepsilon > 0$. *Then*

$$\mathbf{E}(\mathrm{err}_n(\Gamma, \mathbf{P}, Z_1, \ldots, Z_n)) \to 0.$$

In Section 3 we show how this statement can be applied to prove weak consistence of some classical nonparametric predictors in the conditional model.

3. Application to Classical Nonparametric Predictors

In this section we will consider two types of classical nonparametric predictors: partitioning and nearest neighbour classifiers.

The nearest neighbour predictor assigns to a new object x_{n+1} the label of its nearest neighbour among $x_1, \ldots, x_n$:

$$\Gamma_n(x_1, y_1, \ldots, x_n, y_n, x_{n+1}) := y_j,$$

where $j := \mathrm{argmin}_{i=1,\ldots,n} \|x - x_i\|$.

For i.i.d. distributions this predictor is also consistent, i.e.

$$E^\infty(\mathrm{err}_n(\Gamma, P^\infty)) \to 0,$$

for any distribution P on $\mathbf{X}$ (see Devroye, 1981).

We generalise this result as follows.

Theorem 3 *Let* Γ *be the nearest neighbour classifier. Let* $\mathbf{P}$ *be some distribution on* $\mathbf{X}^\infty$ *satisfying (1), (2) and (8). Then*

$$\mathbf{E}(\mathrm{err}_n(\Gamma, \mathbf{P})) \to 0.$$

The proofs for this section can be found in Appendix B.

A partitioning predictor on each step n partitions the object space $\mathbf{X} = \mathbb{R}^d$, $d \in \mathbb{N}$ into disjoint cells $A_1^n, A_2^n, \ldots$ and classifies in each cell according to the majority vote:

$$\Gamma(z_1, \ldots, z_n, x) := \begin{cases} 0 & \text{if } \sum_{i=1}^n I_{y_i=1} I_{x_i \in A(x)} \leq \sum_{i=1}^n I_{y_i=0} I_{x_i \in A(x)} \\ 1 & \text{otherwise,} \end{cases}$$

where $A(x)$ denotes the cell containing x. Define

$$\operatorname{diam}(A) := \sup_{x,y \in A} \|x - y\|$$

and

$$N(x) := \sum_{i=1}^n I_{x_i \in A(x)}.$$

It is a well known result (see, e.g. (Devroye, Györfi, Lugosi, 1996)) that a partitioning predictor is weakly consistent, provided certain regulatory conditions on the size of cells. More precisely, let Γ be a partitioning predictor such that $\operatorname{diam}(A(X)) \to 0$ in probability and $N(X) \to \infty$ in probability. Then for any distribution P on $\mathbf{X}$

$$E^\infty(\operatorname{err}_n(\Gamma, P^\infty)) \to 0.$$

We generalise this result to the case of conditionally i.i.d. examples as follows.

Theorem 4 *Let Γ be a partitioning predictor such that $\operatorname{diam}(A(X)) \to 0$ in probability and $N(X) \to \infty$ in probability, for any distribution generating i.i.d. examples. Then*

$$\mathbf{E}(\operatorname{err}_n(\Gamma, \mathbf{P})) \to 0$$

for any distribution $\mathbf{P}$ on $\mathbf{X}^\infty$ satisfying (1), (2) and (8).

Observe that we only generalise results concerning weak consistency of (one) nearest neighbour and non-data-dependent partitioning rules. More general results exist (see e.g. Devroye et. al., 1994),(Lugosi, Nobel, 1996), in particular for data-dependent rules. However, we do not aim to generalise state-of-the-art results in nonparametric classification, but rather to illustrate that weak consistency results can be extended to the conditional model.

4. Application to Empirical Risk Minimisation

In this section we show how to estimate the performance of a predictor minimising empirical risk (over certain class of functions) using Theorem 1. To do this we estimate the tolerance to data of such predictors, using some results from Vapnik-Chervonenkis theory. For overviews of Vapnik-Chervonenkis theory see (Vapnik, Chervonenkis, 1974; Vapnik, 1998; Devroye, Györfi, Lugosi, 1996).

Let $\mathbf{X} = \mathbb{R}^d$ for some $d \in \mathbb{N}$ and let $\mathcal{C}$ be a class of measurable functions of the form $\varphi : \mathbf{X} \to \mathbf{Y} = \{0, 1\}$, called *decision functions*. For a probability distribution P on $\mathbf{X}$ define $\operatorname{err}(\varphi, P) := P(\varphi(X_i) \neq Y_i)$. If the examples are generated i.i.d. according to some distribution P, the aim is to find a function φ from $\mathcal{C}$ for which $\operatorname{err}(\varphi, P)$ is minimal:

$$\varphi_P = \operatorname{argmin}_{\varphi \in \mathcal{C}} \operatorname{err}(\varphi, P).$$

In the theory of empirical risk minimisation this function is approximated by the function

$$\varphi_n^* := \arg\min_{\varphi\in\mathcal{C}} \overline{\mathrm{err}}_n(\varphi)$$

where $\overline{\mathrm{err}}_n(\varphi) := \sum_{i=1}^n I_{\varphi(X_i)\neq Y_i}$ is the empirical error functional, based on a sample (X_i,Y_i), $i = 1,\ldots,n$. Thus, $\Gamma_n(z_1,\ldots,z_n,x_{n+1}) := \varphi_n^*(x_{n+1})$ is a predictor minimising empirical risk over the class of functions $\mathcal{C}$.

One of the basic results of Vapnik-Chervonenkis theory is the estimation of the difference of probabilities of error between the best possible function in the class (φ_P) and the function which minimises empirical error:

$$P\big(\mathrm{err}_n(\Gamma,P^\infty) - \mathrm{err}(\varphi_P,P) > \varepsilon\big) \le 8\mathcal{S}(\mathcal{C},n)e^{-n\varepsilon^2/128},$$

where the symbol $\mathcal{S}(\mathcal{C},n)$ is used for the n-th shatter coefficient of the class $\mathcal{C}$:

$$\mathcal{S}(\mathcal{C},n) := \max_{A:=\{x_1,\ldots,x_n\}\subset\mathbf{X}} \#\{C\cap A : C\in\mathcal{C}\}.$$

Thus,

$$P(\mathrm{err}_n(\Gamma) > \varepsilon) \le I_{\mathrm{err}(\varphi_P,P)>\varepsilon/2} + 8\mathcal{S}(\mathcal{C},n)e^{-n\varepsilon^2/512}.$$

A particularly interesting case is when the optimal rule belongs to $\mathcal{C}$, i.e. when $\eta\in\mathcal{C}$. This situation was investigated in e.g. (Valiant, 1984; Blumer et. al., 1989). Obviously, in this case $\varphi_P\in\mathcal{C}$ and $\mathrm{err}(\varphi_P,P) = 0$ for any P. Moreover, a better bound exists (see Vapnik, 1998; Blumer et. al., 1989; Devroye, Györfi, Lugosi, 1996)

$$P(\mathrm{err}_n(\Gamma,P) > \varepsilon) \le 2\mathcal{S}(\mathcal{C},n)e^{-n\varepsilon/2}.$$

Theorem 5 *Let $\mathcal{C}$ be a class of decision functions and let Γ be a predictor which for each $n\in\mathbb{N}$ minimises $\overline{\mathrm{err}}_n$ over $\mathcal{C}$ on the observed examples $(z_1,\ldots,z_n)$. Fix some $\delta\in(0,1/2]$, let $p(n) := \frac{1}{n}\#\{i\le n : Y_i = 0\}$ and $C_n := \mathbf{P}(\delta\le p(n)\le 1-\delta)$ for each $n\in\mathbb{N}$. Assume $n > 4/\varepsilon^2$ and let $\alpha_n := \frac{1}{1-1/\sqrt{n}}$. We have*

$$\Delta_\delta(P_0,P_1,n,\varepsilon) \le 16\mathcal{S}(\mathcal{C},n)e^{-n\varepsilon^2/512}. \tag{12}$$

(which does not depend on the distributions P_0, P_1 and δ) and

$$\mathbf{P}(\mathrm{err}_n(\Gamma,\mathbf{P}) > \varepsilon) \le I_{2\,\mathrm{err}(\varphi_{P_{1/2}},P_{1/2})>\varepsilon/2} + 16\alpha_n C_n^{-1}\mathcal{S}(\mathcal{C},n)e^{-n\delta^2\varepsilon^2/2048} + (1-C_n). \tag{13}$$

If in addition $\eta\in\mathcal{C}$ then

$$\Delta(n,\varepsilon) \le 4\mathcal{S}(\mathcal{C},2n)2^{-n\varepsilon/8} \tag{14}$$

and

$$\mathbf{P}(\mathrm{err}_n(\Gamma,\mathbf{P}) > \varepsilon) \le 4\alpha_n C_n^{-1}\mathcal{S}(\mathcal{C},n)e^{-n\delta\varepsilon/16} + (1-C_n). \tag{15}$$

Thus, if we have bounds on the VC dimension of some class of classifiers, we can obtain bounds on the performance of predictors minimising empirical error for the conditional model.

Next we show how strong consistency results can be achieved in the conditional model. For general strong universal consistency results (with examples) see (Lugosi, Zeger, 1995; Vapnik, 1998; Vapnik, Chervonenkis, 1974).

Denote the VC dimension of $\mathcal{C}$ by $V(\mathcal{C})$:

$$V(\mathcal{C}) := \max\{n \in \mathbb{N} : s(\mathcal{C}, n) = 2^n\}.$$

Using Theorem 5 and Borel-Cantelli lemma, we obtain the following corollary.

Corollary 6 *Let $\mathcal{C}^k$, $k \in \mathbb{N}$ be a sequence of classes of decision functions with finite VC dimension such that $\lim_{k\to 0} \inf_{\varphi\in\mathcal{C}^k} \mathrm{err}(\varphi, P) = 0$ for any distribution P on $\mathbf{X}$. If $k_n \to \infty$ and $\frac{V(\mathcal{C}^{k_n})\log n}{n} \to 0$ as $n \to \infty$ then*

$$\mathrm{err}(\Gamma, \mathbf{P}) \to 0 \ \mathbf{P}\text{–}a.s.$$

where Γ is a predictor which in each trial n minimises empirical risk over $\mathcal{C}^{k_n}$ and $\mathbf{P}$ is any distribution satisfying (1), (2) and $\sum_{n=1}^{\infty}(1-C_n) < \infty$.

In particular, if we use bound on the VC dimension on classes of neural networks provided in (Baum, Haussler, 1989) then we obtain the following corollary.

Corollary 7 *Let Γ be a classifier that minimises the empirical error over the class $\mathcal{C}^{(k)}$, where $\mathcal{C}^{(k)}$ is the class of neural net classifiers with k nodes in the hidden layer and the threshold sigmoid, and $k \to \infty$ so that $k\log n/n \to 0$ as $n \to \infty$. Let $\mathbf{P}$ be any distribution on $\mathbf{X}^\infty$ satisfying (1) and (2) such that $\sum_{n=1}^{\infty}(1-C_n) < \infty$. Then*

$$\lim_{n\to\infty} \mathrm{err}_n(\Gamma) = 0 \ \ \mathbf{P}\text{–}a.s.$$

5. Discussion

We have introduced "conditionally i.i.d." model for pattern recognition which generalises the commonly used i.i.d. model. Naturally, a question arises whether our conditions on the distributions and on predictors are necessary, or they can be yet more generalised in the same direction. In this section we discuss the conditions of the new model from this point of view.

The first question is, can the same results be obtained without assumptions on tolerance to data? The following negative example shows that some **bounds on tolerance to data are necessary**.

Remark 8 *There exists a distribution $\mathbf{P}$ on $\mathbf{X}^\infty$ satisfying (1) and (2) such that $\mathbf{P}(|p_n - 1/2| > 3/n) = 0$ for any n (i.e. $C_n = 1$ for any $\delta \in (0, 1/2)$ and $n > \frac{3}{(1/2-\delta)}$) and a predictor Γ such that $P_p^n(\mathrm{err}_n > 0) \le 2^{1-n}$ for any $p \in [\delta, 1-\delta]$ and $\mathbf{P}(\mathrm{err}_n = 1) = 1$ for $n > 1$.*

Proof Let $\mathbf{X} = \mathbf{Y} = \{0,1\}$. We define the distributions P_y as $P_y(X = y) = 1$, for each $y \in \mathbf{Y}$ (i.e. $\eta(x) = x$ for each x). The distribution $\mathbf{P}|_{\mathbf{Y}^\infty}$ is defined as a Markov distribution with transition probability matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, i.e. it always generates sequences of labels $\ldots 01010101 \ldots$.

We define the predictor Γ as follows

$$\Gamma_n := \begin{cases} 1 - x_n & \text{if } |\#\{i < n : y_i = 0\} - n/2| \le 1, \\ x_n & \text{otherwise.} \end{cases}$$

So, in the case when the distribution $\mathbf{P}$ is used to generate the examples, Γ is always seeing either $n-1$ zeros and n ones, or n zeros and n ones which, consequently, will lead it to always predict the

wrong label. It remains to note that this is very improbable in the case of an i.i.d. distribution. ■

Another point is **the requirement on the frequencies of labels**. In particular, the assumption (8) might appear redundant: if the rate of occurrence of some label tends to zero, can we just ignore this label without affecting the asymptotic? It appears that this is not the case, as the following example illustrates.

Remark 9 *There exist a distribution* $\mathbf{P}$ *on* $\mathbf{X}^\infty$ *which satisfies (1) and (2) but for which the nearest neighbour predictor is not consistent, i.e. the probability of error does not tend to zero.*

Proof Let $\mathbf{X} = [0,1]$, let $\eta(x) = 0$ if x is rational and $\eta(x) = 1$ otherwise. The distribution P_1 is uniform on the set of irrational numbers, while P_0 is any distribution such that $P(x) \neq 0$ for any rational x. (This construction is due to T. Cover.) The nearest neighbour predictor is consistent for any i.i.d. distribution which agrees with the definition, i.e. for any $p = P(Y = 1) \in [0,1]$.

Next we construct the distribution $\mathbf{P}|_{\mathbf{Y}^\infty}$. Fix some ε, $0 < \varepsilon < 1$. Assume that according to $\mathbf{P}$ the first label is always 1, (i.e. $\mathbf{P}(y_1 = 1) = 1$; the object is an irrational number). Next k_1 labels are always 0 (rationals), then follows 1, then k_2 zeros, and so on. It is easy to check that there exists such sequence $k_1, k_2, \ldots$ that with probability at least ε we have

$$\max_{i<n:\ X_i \text{ is irrational}} P_1\{x : X_i \text{ is the nearest neighbour of } x\} \leq \frac{1-\varepsilon}{m(n)},$$

where $m(n)$ is the total number of irrational objects up to the trial n. On each step n such that $n = t + \sum_{j=1}^{t} k_t$ for some $t \in \mathbb{N}$ (i.e. on each irrational object) we have

$$\mathbf{E}(\mathrm{err}_n(\Gamma, \mathbf{P})) \quad \geq \quad \varepsilon \left(1 - \sum_{j<n:\ X_j \text{ is irrational}} \mathbf{P}(X_j \text{ is the nearest neighbour of } X)\right) \quad \geq \quad \varepsilon^2$$

As irrational objects are generated infinitely often (that is, with intervals k_i), the probability of error does not tend to zero. ■

Another question is whether the results can be generalised to the case of **non-deterministically defined labels**, which is often considered in literature. It should be noted that we consider the task of learning object-label dependence, ignoring the label-label dependence (and prohibiting any dependence apart from these). On one hand, it allows us to consider any sort of label-label dependence. On the other hand, the best bound on the probability of error we can obtain is the maximum of the class-conditional probabilities of error (as nothing is known about the probability of the next label), and not the so-called Bayes error, which is the best achievable bound in the i.i.d. case. Thus, if we want to consider stochastically defined labels, we should restrict our attention to class-conditional probabilities of error. On this way also some obstacles can be met. In particular, the function η, which in this case is defined as $\eta(x) := \mathbf{P}(Y_n = 1 | X_n = x)$ should not depend on n, which will require more restrictive definition of constants C_n and the condition (8). We leave this question for further investigation.

As it was mentioned in Section 2, for the sake of simplicity of notations, all results are formulated for the case of binary labels $\mathbf{Y} = \{0,1\}$; however, they can be easily extended to **the case of**

any finite label space. Indeed, to pass to the general case only the following changes should be made. With each distribution satisfying the conditions of the model (1) and (2) we associate (not two but) $|\mathbf{Y}|$ distributions P_a, $a \in \mathbf{Y}$, defined by $\mathbf{P}(X_n|Y_n = a)$ (which does not depend on n). Analogously to the binary case, these distributions are used to define (in the natural way) the family of distributions P_q (cf. P_p of Section 2), where q stands for any probability distribution over the set $\mathbf{Y}$. The definitions (3) and (4) take the form

$$\bigtriangledown_\delta(P_0,P_1,n,\varepsilon) := \sup_q P_q^\infty(\mathrm{err}_n(\Gamma) > \varepsilon) \tag{16}$$

and

$$\Delta_\delta(P_0,P_1,n,\varepsilon) := \sup_q \Delta(P_q,n,\varepsilon), \tag{17}$$

where the supremums are with respect to all distributions q such that $\min_{a\in\mathbf{Y}} q(a) \geq \delta$. All theorems retain their form with $p(n,a)$ (instead of $p(n)$)defined as $\frac{1}{n}\#\{i \leq n : y_i = a\}$ and C_n as $P(\min_{a\in\mathbf{Y}} p(n,a) \geq \delta)$. It is easy to alter the proofs according to these changes. However, as it can be seen, the notation becomes significantly more cumbersome.

Finally, the choice of the constants $\varkappa_n$ requires clarification. We have fixed these constants for the sake of simplicity of notations, however, they can be made variable, as long as $\varkappa_n$ obeys the following condition.

$$\lim_{n\to\infty}\{n|p_n - p| \leq \varkappa_n\} = 0$$

almost surely for any $p \in (0,1)$ and any probability distribution P on $\mathbf{X}$ such that $P(y = 1) = p$, where $p_n := \frac{1}{n}\#\{i \leq n : Y_i = 0\}$.

Acknowledgements

Parts of the results were reported on International Conference on Machine Learning, 2004, (Ryabko, 2004a), and on 15th International Conference on Algorithmic Learning Theory, 2004, (Ryabko, 2004b). The main results were obtained while the author was a Ph.D. student at Royal Holloway, University of London. The research was partially supported by the Swiss NSF grant 200020-107616.

Appendix A: Proofs for Section 2

Before proceeding with the proof of Theorem 1 we give some definitions and supplementary facts.

Define the conditional probabilities of error of Γ as follows

$$\mathrm{err}_n^0(\Gamma,\mathbf{P},z_0,\dots,z_n) := \mathbf{P}(Y_{n+1} \neq \Gamma(z_1,\dots,z_n,X_{n+1})|Y_{n+1} = 0),$$

$$\mathrm{err}_n^1(\Gamma,\mathbf{P},z_0,\dots,z_n) := \mathbf{P}(Y_{n+1} \neq \Gamma(z_1,\dots,z_n,X_{n+1})|Y_{n+1} = 1),$$

(with the same notational convention as used with the definition of $\mathrm{err}_n(\Gamma)$). In words, for each $y \in \mathbf{Y} = \{0,1\}$ we define err_n^y as the probability of all $x \in \mathbf{X}$, such that Γ makes an error on n'th trial, given that $Y_{n+1} = y$ and fixed $z_1,\dots,z_n$.

For any $\mathbf{y} := (y_1,y_2,\dots) \in \mathbf{Y}^\infty$, define $\mathbf{y}_n := (y_1,\dots,y_n)$ and $p_n(\mathbf{y}) := \frac{1}{n}\#\{i \leq n : y_i = 0\}$, for $n > 1$.

Clearly (from the assumption (1)) the random variables $X_1,\dots,X_n$ are mutually conditionally independent given $Y_1,\dots,Y_n$, and by (2) they are distributed according to P_{Y_i}, $1\le i\le n$. Hence, the following statement is valid.

Lemma 10 *Fix some $n>1$ and some $\mathbf{y}\in\mathbf{Y}^\infty$ such that $\mathbf{P}((Y_1,\dots Y_{n+1})=\mathbf{y}_{n+1})\ne 0$. Then*

$$\mathbf{P}\bigl(\operatorname{err}_n^{y_{n+1}}(\Gamma) \;>\; \varepsilon \,\big|\, (Y_1,\dots,Y_n) \;=\; \mathbf{y}_n\bigr) \;=\; P_p^n\bigl(\operatorname{err}_n^{y_{n+1}}(\Gamma) \;>\; \varepsilon \,\big|\, (Y_1,\dots,Y_n) \;=\; \mathbf{y}_n\bigr)$$

for any $p\in(0,1)$.

Where, accordingly to the notational conventions made above,

$$\mathbf{P}\bigl(\operatorname{err}_n^{y_{n+1}}(\Gamma)>\varepsilon \,\big|\, (Y_1,\dots,Y_n)=\mathbf{y}_n\bigr)=\mathbf{P}\bigl\{x_1,\dots,x_n : \operatorname{err}_n(\Gamma,\mathbf{P},x_1,y_1,\dots,x_n,y_n)>\varepsilon\bigr\};$$

that is, having fixed the labels, we consider probability over objects only.

Proof of Theorem 1. Fix some $n>1$, some $y\in\mathbf{Y}$ and such $\mathbf{y}^1\in\mathbf{Y}^\infty$ that $\delta\le p_n(\mathbf{y}^1)\le 1-\delta$ and $\mathbf{P}((Y_1,\dots,Y_n)=\mathbf{y}^1_n)\ne 0$. Let $p:=p_n(\mathbf{y}^1)$. We will find bounds on $\mathbf{P}\bigl(\operatorname{err}_n(\Gamma)>\varepsilon \mid (Y_1,\dots,Y_n)=\mathbf{y}^1_n\bigr)$, first in terms of Δ and then in terms of Δ.

Lemma 10 allows us to pass to the i.i.d. case:

$$\mathbf{P}\bigl(\operatorname{err}_n^y(\Gamma,X_1,y_1^1,\dots,X_n,y_n^1,X_{n+1}) \;>\; \varepsilon\bigr) \;=\; P_p^n\bigl(\operatorname{err}_n^y(\Gamma,X_1,y_1^1,\dots,X_n,y_n^1,X_{n+1}) \;>\; \varepsilon\bigr)$$

for any y such that $\mathbf{P}(Y_1=y_1^1,\dots,Y_n=y_n^1,Y_{n+1}=y)\ne 0$ (recall that we use upper-case letters for random variables and lower-case for fixed variables, so that the probabilities in the above formula are labels-conditional).

Clearly, for $\delta\le p\le 1-\delta$ we have $\operatorname{err}_n(\Gamma,P_p)\le\max_{y\in\mathbf{Y}}(\operatorname{err}_n^y(\Gamma,P_p))$, and if $\operatorname{err}_n(\Gamma,P_p)<\varepsilon$ then $\operatorname{err}_n^y(\Gamma,P_p)<\varepsilon/\delta$ for each $y\in\mathbf{Y}$.

Let m be such number that $m-\varkappa_m=n$. For any $\mathbf{y}^2\in\mathbf{Y}^\infty$ such that $|mp_m(\mathbf{y}^2)-mp|\le\varkappa_m/2$ there exist such mapping $\pi:\{1,\dots,n\}\to\{1,\dots,m\}$ that $y^2_{\pi(i)}=y^1_i$ for any $i\le n$. Define random variables $X'_1\dots X'_m$ as follows: $X'_{\pi(i)}:=X_i$ for $i\le n$, while the rest $\varkappa_m$ of X'_i are some random variables independent from $X_1,\dots,X_n$ and from each other, and distributed according to P_p (a "ghost sample"). We have

$$\begin{aligned}
&P_p^n\bigl(\operatorname{err}_n^y(X_1,y_1^1,\dots,X_n,y_n^1)>\varepsilon\bigr)\\
&\quad= P_p^m\Bigl(\operatorname{err}_n^y(X_1,y_1^1,\dots,X_n,y_n^1)-\operatorname{err}_n^y(X'_1,y_1^2,\dots,X'_m,y_m^2)+\operatorname{err}_n^y(X'_1,y_1^2,\dots,X'_m,y_m^2)>\varepsilon\Bigr)\\
&\qquad\le P_p^m\Bigl(\bigl|\operatorname{err}_n^y(X'_1,y_1^2,\dots,X'_n,y_n^2)-\operatorname{err}_n^y(X_1,y_1^1,\dots,X_n,y_n^1)\bigr|>\varepsilon/2\Bigr)\\
&\qquad\qquad+P_p^n\Bigl(\operatorname{err}_n^y(X'_1,y_1^2,\dots,X'_n,y_n^2)>\varepsilon/2\Bigr).
\end{aligned}$$

Observe that $\mathbf{y}^2$ was chosen arbitrarily (among sequences for which $|mp_m(\mathbf{y}^2)-mp|\le\varkappa_m/2$) and $(X_1,y_1^1,\dots,X_ny_n^1)$ can be obtained from $(X'_1,y_1^2,\dots,X'_my_m^2)$ by removing at most $\varkappa_m$ elements and applying some permutation. Thus the first term is bounded by

$$\begin{aligned}
P_p^m\Bigl(&\max_{j\le\varkappa_m;\ \pi:\{1,\dots,m\}\to\{1,\dots,m\}}\bigl|\operatorname{err}_m^y(\Gamma,Z_1,\dots,Z_m)-\\
&\qquad\operatorname{err}_{m-j}^y(\Gamma,Z_{\pi(1)},\dots,Z_{\pi(m-j)})\bigr|>\varepsilon/2 \,\big|\, |mp(m)-mp|\le\varkappa_m/2\Bigr)\\
&\qquad\le\frac{\Delta(P_p,m,\delta\varepsilon/2)}{P_p^n(|mp(m)-mp|\le\varkappa_m)}\le\frac{1}{1-1/\sqrt{m}}\Delta(P_p,m,\delta\varepsilon/2),
\end{aligned}$$

and the second term is bounded by $\frac{1}{1-1/\sqrt{m}}P_p^m(\mathrm{err}_m(\Gamma) > \delta\varepsilon/2)$. Hence

$$P_p^n\big(\mathrm{err}_n^y(X_1,y_1^1,\ldots,X_n,y_n^1) \;>\; \varepsilon\big) \;\le\; \alpha_n\big(\Delta(P_p,m,\delta\varepsilon/2) \;+\; P_p^m(\mathrm{err}_m(\Gamma) \;>\; \delta\varepsilon/2)\big). \quad (18)$$

Next we establish a similar bound in terms of $\bar{\Delta}$. For any $\mathbf{y}_n^2 \in \mathbf{Y}^n$ such that $|np_n(\mathbf{y}^2) - np| \le \varkappa_n/2$ there exist such permutations π_1,π_2 of the set $\{1,\ldots,n\}$ that $y^1_{\pi_1(i)} = y^2_{\pi_2(i)}$ for any $i \le n - \delta\varkappa_n$. Denote $n-\delta\varkappa_n$ by n' and define random variables $X'_1 \ldots X'_n$ as follows: $X'_{\pi_2(i)} := X_{\pi_1(i)}$ for $i \le n'$, while for $n' < i \le n$ X'_i are some "ghost" random variables independent from $X_1,\ldots,X_n$ and from each other, and distributed according to P_p. We have

$$\begin{aligned}
&P_p^n\big(\mathrm{err}_n^y(X_1,y_1^1,\ldots,X_n,y_n^1) > \varepsilon\big)\\
&\qquad \le P_p^{n+\varkappa_n}\Big(\big|\mathrm{err}_n^y(X'_1,y_1^2,\ldots,X'_n,y_n^2) - \mathrm{err}_n^y(X_1,y_1^1,\ldots,X_n,y_n^1)\big| > \varepsilon/2\Big)\\
&\qquad\qquad + P_p^n\Big(\mathrm{err}_n^y(X'_1,y_1^2,\ldots,X'_n,y_n^2) > \varepsilon/2\Big),
\end{aligned}$$

Again, $\mathbf{y}^2$ was chosen arbitrarily (among sequences for which $|np_n(\mathbf{y}^2) - np| \le \varkappa_n/2$) and

$$(X_1,y_1^1,\ldots,X_ny_n^1)$$

differs from

$$(X'_1,y_1^2,\ldots,X'_ny_n^2)$$

in at most $\varkappa_n$ elements, up to some permutation. Thus the first term is bounded by

$$\begin{aligned}
P_p^n\Big(&\sup_{j<\varkappa_n;\pi:\{1,\ldots,n\}\to\{1,\ldots,n\};z'_{n-j},\ldots,z'_n} |\,\mathrm{err}_n^y(Z_1,\ldots,Z_n)\\
&\qquad - \mathrm{err}_n^y(\zeta_1,\ldots,\zeta_n)| > \varepsilon/2 \,\big|\, |np(n) - np| \le \varkappa_n/2\Big) \le \alpha_n\bar{\Delta}(P_p,n,\delta\varepsilon/2),
\end{aligned}$$

and the second term is bounded by $\alpha_n P_p^n(\mathrm{err}_n(\Gamma) > \delta\varepsilon/2)$. Hence

$$P_p^n\big(\mathrm{err}_n^y(X_1,y_1^1,\ldots,X_n,y_n^1) \;>\; \varepsilon\big) \;\le\; \alpha_n\big(\bar{\Delta}(P_p,n,\delta\varepsilon/2) \;+\; P_p^n(\mathrm{err}_n(\Gamma) \;>\; \delta\varepsilon/2)\big). \quad (19)$$

Finally, as $\mathbf{y}^1$ was chosen arbitrarily among sequences $\mathbf{y} \in \mathbf{Y}^\infty$ such that $n\delta \le p_n(\mathbf{y}^1) \le n(1-\delta)$ from (18) and (19), we obtain (6) and (7). □

Appendix B: Proofs for Section 3

The first part of the proof is common for theorems 3 and 4. Let us fix some distribution $\mathbf{P}$ satisfying conditions of the theorems. It is enough to show that

$$\sup_{p\in[\delta,1-\delta]} E^\infty(\mathrm{err}_n(\Gamma,P_p,Z_1,\ldots,Z_n)) \to 0$$

and

$$\sup_{p\in[\delta,1-\delta]} E^\infty(\bar{\Delta}(P_p,n,Z_1,\ldots,Z_n)) \to 0$$

for nearest neighbour and partitioning predictor, and apply Corollary 2.

Observe that both predictors are symmetric, i.e. do not depend on the order of $Z_1,\dots,Z_n$. Thus, for any $z_1,\dots,z_n$

$$\bar{\Delta}(P_p,n,z_1,\dots,z_n) = \sup_{j\le\varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\},z'_{n-j},\dots,z'_n} |\operatorname{err}_n(\Gamma,P_p,z_1,\dots,z_n) - \operatorname{err}_n(\Gamma,P_p,z_{\pi(1)},\dots,z_{\pi(n-j)},z'_{n-j},\dots,z'_n)|,$$

where the maximum is taken over all z'_i consistent with η, $n-j\le i\le n$. Define also the class-conditional versions of $\bar{\Delta}$:

$$\bar{\Delta}^y(P_p,n,z_1,\dots,z_n) := \sup_{j\le\varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\},z'_{n-j},\dots,z'_n} |\operatorname{err}^y_n(\Gamma,P_p,z_1,\dots,z_n) - \operatorname{err}^y_n(\Gamma,P_p,z_{\pi(1)},\dots,z_{\pi(n-j)},z'_{n-j},\dots,z'_n)|.$$

Note that (omitting $z_1,\dots,z_n$ from the notation) $\operatorname{err}_n(\Gamma,P_p)\le\operatorname{err}^0_n(\Gamma,P_p)+\operatorname{err}^1_n(\Gamma,P_p)$ and $\bar{\Delta}(P_p,n)\le\bar{\Delta}^0(P_p,n)+\bar{\Delta}^1(P_p,n)$. Thus, it is enough to show that

$$\sup_{p\in[\delta,1-\delta]} E^\infty(\operatorname{err}^1_n(\Gamma,P_p))\to 0 \tag{20}$$

and

$$\sup_{p\in[\delta,1-\delta]} E^\infty(\bar{\Delta}^1(P_p,n))\to 0. \tag{21}$$

Observe that for each of the predictors in question the probability of error given that the true label is 1 will not decrease if an arbitrary (possibly large) portion of training examples labelled with ones is replaced with an arbitrary (but consistent with η) portion of the same size of examples labelled with zeros. Thus, for any n and any $p\in[\delta,1-\delta]$ we can decrease the number of ones in our sample (by replacing the corresponding examples with examples from the other class) down to (say) $\delta/2$, not decreasing the probability of error on examples labelled with 1. So,

$$E^\infty(\operatorname{err}^1_n(\Gamma,P_p))\le E^\infty(\operatorname{err}^1_n(\Gamma,P_{\delta/2}|p_n=\delta/2))+P_p(p_n\le\delta/2), \tag{22}$$

where as usual $p_n := \frac{1}{n}\#\{i\le n : y_i=1\}$. Obviously, the last term (quickly) tends to zero. Moreover, it is easy to see that

$$\begin{aligned} E^\infty(\operatorname{err}^1_n(\Gamma,P_{\delta/2})|p_n=n(\delta/2)) &\le E^\infty\big(\operatorname{err}^1_n(\Gamma,P_{\delta/2})\big||n(\delta/2)-p_n|\le\varkappa_n/2\big)+E^\infty(\bar{\Delta}^1(P_{\delta/2},n)) \\ &\le \frac{1}{1-1/\sqrt{n}}E^\infty(\operatorname{err}^1_n(\Gamma,P_{\delta/2}))+E^\infty(\bar{\Delta}^1(P_{\delta/2},n)). \end{aligned} \tag{23}$$

The first term tends to zero, as it is known from the results for i.i.d. processes; thus, to establish (20) we have to show that

$$E(\bar{\Delta}^1(P_p,n,Z_1,\dots,Z_n))\to 0 \tag{24}$$

for any $p\in(0,1)$.

We will also show that (24) is sufficient to prove (21). Indeed,

$$\bar{\Delta}^1(P_p,n,z_1,\dots,z_n) \le \mathrm{err}_n^1(\Gamma,P_p,z_1,\dots,z_n)+ \sup_{j\le \varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\},z'_{n-j},\dots,z'_n} \mathrm{err}_n^1(\Gamma,P_p,z_{\pi(1)},\dots,z_{\pi(n-j)},z'_{n-j},\dots,z'_n)$$

Denote the last summand by D. Again, we observe that D will not decrease if an arbitrary (possibly large) portion of training examples labelled with ones is replaced with an arbitrary (but consistent with η) portion of the same size of examples labelled with zeros. Introduce $\widetilde{\Delta}^1(P_p,n,z_1,\dots,z_n)$ as $\bar{\Delta}^1(P_p,n,z_1,\dots,z_n)$ with $\varkappa_n$ in the definition replaced by $\frac{2}{\delta}\varkappa_n$. Using the same argument as in (22) and (23) we have

$$E^\infty(D) \quad \le \quad \frac{1}{1-1/\sqrt{n}}\big(E^\infty(\widetilde{\Delta}^1(P_{\delta/2},n)) \quad + \quad E^\infty(\mathrm{err}_n(\Gamma,P_{\delta/2})\big) \quad + \quad P_p(p_n \quad \le \quad \delta/2).$$

Thus, (21) holds true if (24) and

$$E^\infty(\widetilde{\Delta}^1(P_p,n,Z_1,\dots,Z_n)) \to 0. \tag{25}$$

Finally, we will prove (24); it will be seen that the proof of (25) is analogous (i.e. replacing $\varkappa_n$ by $\frac{2}{\delta}\varkappa_n$ does not affect the proof). Note that

$$E^\infty(\bar{\Delta}(P_p,n,Z_1,\dots,Z_n)) \le P_p\Big(\sup_{j\le \varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\},z'_{n-j},\dots,z'_n} \big|\,\mathrm{err}_n(\Gamma,P_p,Z_1,\dots,Z_n) \ne \mathrm{err}_n(\Gamma,P_p,Z_{\pi(1)},\dots,Z_{\pi(n-j)},z'_{n-j},\dots,z')\big|\Big),$$

where the maximum is taken over all z'_i consistent with η, $n-j\le i\le n$. The last expression should be shown to tend to zero. This we will prove for each of the predictors separately.

Nearest Neighbour predictor. Fix some distribution P_p, $0<p<1$ and some $\varepsilon>0$. Fix also some $n\in\mathbb{N}$ and define (leaving $x_1,\dots,x_n$ implicit)

$$B_n(x) := P_p^{n+1}\{t\in\mathbf{X} : t \text{ and } x \text{ have the same nearest neighbour among } x_1,\dots,x_n\}$$

and $B_n := E(B_n(X))$ Note that $E^\infty(B_n)=1/n$, where the expectation is taken over $X_1,\dots,X_n$. Define $\mathcal{B} := \{(x_1,\dots,x_n)\in\mathbf{X}^n : B_n\le 1/n\varepsilon\}$ and $\mathcal{A}(x_1,\dots,x_n) := \{x : B_n(x)\le 1/n\varepsilon^2\}$. Applying Markov's inequality twice, we obtain

$$\begin{aligned} E^\infty(\bar{\Delta}(P_p,n)) &\le E^\infty(\bar{\Delta}(P_p,n)|(X_1,\dots,X_n)\in\mathcal{B})+\varepsilon \\ &\le E^\infty\Big(\sup_{j\le\varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\},z'_{n-j},\dots,z'_n} \\ &P_p\big\{x : \mathrm{err}_n(\Gamma,P_p,Z_1,\dots,Z_n)\ne\mathrm{err}_n(\Gamma,P_p,Z_{\pi(1)},\dots,Z_{\pi(n-j)},z'_{n-j},\dots,z'_n) \\ &\big|x\in\mathcal{A}(X_1,\dots,X_n)\big\}\big|(X_1,\dots,X_n)\in\mathcal{B}\Big)+2\varepsilon. \end{aligned} \tag{26}$$

Removing one point x_i from a sample $x_1,\dots,x_n$ we can only change the value of Γ in the area

$$\{x\in\mathbf{X} : x_i \text{ is the nearest neighbour of x}\} = B_n(x_i),$$

while adding one point x_0 to the sample we can change the value of Γ in the area

$$D_n(x_0) := \{x \in \mathbf{X} : x_0 \text{ is the nearest neighbour of x}\}.$$

It can be shown that the number of examples (among $x_1,\dots,x_n$) for which a point x_0 is the nearest neighbour is not greater than a constant γ which depends only the space $\mathbf{X}$ (see Devroye, Györfi, Lugosi, 1996, Corollary 11.1). Thus,

$$D_n(x_0) \subset \cup_{i=j_1,\dots,j_\gamma} B_n(x_i)$$

for some $j_1,\dots,j_\gamma$, and so

$$E^\infty(\bar{\Delta}(P_p,n)) \le 2\varepsilon + 2(\gamma+1)\varkappa_n E^\infty(\max_{x\in\mathcal{A}(X_1,\dots,X_n)} B_n(x) | (X_1,\dots,X_n)\in\mathcal{B})$$
$$\le 2\varkappa_n \frac{\gamma+1}{n\varepsilon^2} + 2\varepsilon,$$

which, increasing n, can be made less than 3ε. □

Partitioning predictor. For any measurable sets $\mathcal{B}\subset\mathbf{X}^n$ and $\mathcal{A}\subset\mathbf{X}$ define

$$D(\mathcal{B},\mathcal{A}) := E^\infty\Big(\sup_{j\le\varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\},z'_{n-j},\dots,z'_n} P_p\big\{x : \mathrm{err}_n(\Gamma,P_p,Z_1,\dots,Z_n) \ne \mathrm{err}_n(\Gamma,P_p,Z_{\pi(1)},\dots,Z_{\pi(n-j)},z'_{n-j},\dots,z'_n)$$
$$\big| x\in\mathcal{A}\big\}\big|(X_1,\dots,X_n)\in\mathcal{B}\Big) + 2\varepsilon.$$

and $D := D(\mathbf{X}^n,\mathbf{X})$.

Fix some distribution P_p, $0<p<1$ and some $\varepsilon>0$. Introduce

$$\hat{\eta}(x,X_1,\dots,X_n) := \frac{1}{N(x)}\sum_{i=1}^n I_{Y_i=1} I_{X_i\in A(x)}$$

($X_1,\dots X_n$ will usually be omitted). From the consistency results for i.i.d. model (see, e.g. Devroye, Györfi, Lugosi, 1996, Theorem 6.1) we know that $E^{n+1}|\hat{\eta}_n(X)-\eta(X)|\to 0$ (the upper index in E^{n+1} indicating the number of examples it is taken over).

Thus, $E|\hat{\eta}_n(X)-\eta(X)|\le\varepsilon^4$ from some n on. Fix any such n and let $\mathcal{B} := \{(x_1,\dots,x_n) : E|\hat{\eta}_n(X)-\eta(X)|\le\varepsilon^2\}$. By Markov inequality we obtain $P_p(\mathcal{B})\ge 1-\varepsilon^2$. For any $(x_1,\dots,x_n)\in\mathcal{B}$ let $\mathcal{A}(x_1,\dots,x_n)$ be the union of all cells A_i^n for which $E(|\hat{\eta}_n(X)-\eta(X)||X\in A_i^n)\le\varepsilon$. Clearly, with $x_1,\dots,x_n$ fixed, $P_p(X\in\mathcal{A}(x_1,\dots,x_n))\ge 1-\varepsilon$. Moreover, $D\le D(\mathcal{B},\mathcal{A})+\varepsilon+\varepsilon^2$.

Fix $\mathcal{A} := (x_1,\dots,x_n)$ for some $(x_1,\dots,x_n)\in\mathcal{B}$. Since $\eta(x)$ is always either 0 or 1, to change a decision in any cell $A\subset\mathcal{A}$ we need to add or remove at least $(1-\varepsilon)N(A)$ examples, where $N(A) := N(x)$ for any $x\in A$. Let $N(n) := E(N(X))$ and $A(n) := E(P_p(A(X))$. Clearly, $\frac{N(n)}{nA(n)} = 1$ for any n, as $E\frac{N(X)}{n} = A(n)$.

As before, using Markov inequality and shrinking $\mathcal{A}$ if necessary we can have

$$P_p\Big(\frac{\varepsilon^2 nA(X)}{N(n)}\le\varepsilon | X\in\mathcal{A}\Big) = 1,\ \ P_p\Big(\frac{\varepsilon^2 nA(n)}{N(X)}\le\varepsilon|X\in\mathcal{A}\Big) = 1,$$

and $D\le D(\mathcal{B},\mathcal{A})+3\varepsilon+\varepsilon^2$. Thus, for all cells $A\subset\mathcal{A}$ we have $N(A)\ge\varepsilon nA(n)$, so that the probability of error can be changed in at most $2\frac{\varkappa_n}{(1-\varepsilon)\varepsilon nA(n)}$ cells; but the probability of each cell is not greater than $\frac{N(n)}{\varepsilon n}$. Hence $E^\infty(\bar{\Delta}(P_p,n))\le 2\frac{\varkappa_n}{n(1-\varepsilon)\varepsilon^2}+3\varepsilon+\varepsilon^2$. □

Appendix C: Proofs for Section 4

Proof of Theorem 5. Fix some probability distribution P_p and some $n \in \mathbb{N}$. Let $\varphi^\times$ be any decision rule $\varphi \in \mathcal{C}$ picked by $\Gamma_{n-\varkappa_n}$ on which (along with the corresponding permutation) the maximum

$$\max_{j \le \varkappa_n;\ \pi:\{1,\dots,n\}\to\{1,\dots,n\}} |\operatorname{err}_n(\Gamma, z_1, \dots, z_n) - \operatorname{err}_{n-j}(\Gamma, z_{\pi(1)}, \dots, z_{\pi(n-j)})|$$

is reached. We need to estimate $P^n(|\operatorname{err}(\varphi^*) - \operatorname{err}(\varphi^\times)| > \varepsilon)$.

Clearly, $|\overline{\operatorname{err}}_n(\varphi^\times) - \overline{\operatorname{err}}_n(\varphi^*)| \le \varkappa_n$, as $\varkappa_n$ is the maximal number of errors which can be made on the difference of the two samples.

Moreover,

$$\begin{aligned} &P^n\big(|\operatorname{err}(\varphi_n^*) - \operatorname{err}(\varphi^\times)| > \varepsilon\big) \\ &\quad \le P^n\big(|\operatorname{err}(\varphi_n^*) - \frac{1}{n}\overline{\operatorname{err}}_n(\varphi^*)| > \varepsilon/2\big) + P^n\big(|\frac{1}{n}\overline{\operatorname{err}}_n(\varphi^\times) - \operatorname{err}(\varphi^\times)| > \varepsilon/2 - \varkappa_n/n\big) \end{aligned}$$

Observe that

$$P^n(\sup_{\varphi\in\mathcal{C}} |\frac{1}{n}\overline{\operatorname{err}}_n(\varphi) - \operatorname{err}(\varphi)| > \varepsilon) \le 8\mathcal{S}(\mathcal{C}, n)e^{-n\varepsilon^2/32}, \tag{27}$$

see (Devroye, Györfi, Lugosi, 1996, Theorem 12.6). Thus,

$$\Delta(P_p, n, \varepsilon) \le 16\mathcal{S}(\mathcal{C}, n)e^{-n(\varepsilon/2-\varkappa_n/n)^2/32} \le 16\mathcal{S}(\mathcal{C}, n)e^{-n\varepsilon^2/512}$$

for $n > 4/\varepsilon^2$. So,

$$\mathbf{P}(\operatorname{err}_n(\Gamma, \mathbf{P}) > \varepsilon) \le I_{\sup_{p\in[\delta,1-\delta]} \operatorname{err}(\varphi_{P_p}, P_p) > \varepsilon/2} + 16\alpha C_n^{-1}\mathcal{S}(\mathcal{C}, n)e^{-n\delta^2\varepsilon^2/2048} + (1 - C_n).$$

It remains to notice that

$$\begin{aligned} &\operatorname{err}(\varphi_{P_p}, P_p) = \inf_{\varphi\in\mathcal{C}} (p\operatorname{err}^1(\varphi, P_p) + (1-p)\operatorname{err}^0(\varphi, P_p)) \\ &\le \inf_{\varphi\in\mathcal{C}} (\operatorname{err}^1(\varphi, P_{1/2}) + \operatorname{err}^0(\varphi, P_{1/2})) = 2\operatorname{err}(\varphi_{P_{1/2}}, P_{1/2}) \end{aligned}$$

for any $p \in [0, 1]$.

So far we have proven (12) and (13); (14) and (15) can be proven analogously, only for the case $\eta \in \mathcal{C}$ we have

$$P^n(\sup_{\varphi\in\mathcal{C}} |\frac{1}{n}\overline{\operatorname{err}}_n(\varphi) - \operatorname{err}(\varphi)| > \varepsilon) \le \mathcal{S}(\mathcal{C}, n)e^{-n\varepsilon}$$

instead of (27), and $\operatorname{err}(\varphi_{P_p}, P_p) = 0$. □

References

D. Aldous and U. Vazirani, *A Markovian Extension of Valiant's Learning Model.* In Proceedings of the 31st Symposium on Foundations of Computer Science, pp. 392–396, 1990.

P. Algoet, *Universal Schemes for Learning the Best Nonlinear Predictor Given the Infinite Past and Side Information.* IEEE Transactions on Information Theory, Vol. 45, No. 4, 1999.

P. Bartlett, S. Ben-David, S. Kulkarni, *Learning Changing Concepts by Exploiting the Structure of Change.* In Proceedings of the Workshop on Computational Learning Theory, pp. 131–139, Morgan Kaufmann Publishers, 1996.

E. Baum and D. Haussler, *What Size Net Gives Valid Generalisation?* Neural Computation, 1:151-160, 1989.

A. Blumer, A. Ehrenfeucht, D. Haussler M and Warmuth *Learnability and the Vapnik-Chervonenkis Dimension.* Journal of the ACM, 36, pp. 929–965, 1989.

O. Bousquet, A. Elisseeff. *Stability and Generalization.* Journal of Machine Learning Research, 2: 499-526, 2002.

A.P. Dawid *Conditional Independence in Statistical Theory.* Journal of the Royal Statistical Society, Series B (Methodological), Vol. 41 No 1, pp. 1–31, 1979.

L. Devroye, *On Asymptotic Probability of Error in Nonparametric Discrimination.* Annals of Statistics, Vol. 9, No. 6, pp. 1320–1327, 1981.

L. Devroye, L. Györfi, A. Krzyżak, G. Lugosi, *On the Strong Universal Consistency of Nearest Neighbor Regression Function Estimates.* Annals of Statistics, Vol. 22, pp. 1371–1385, 1994.

L. Devroye, L. Györfi, G. Lugosi, A Probabilistic Theory of Pattern Recognition. New York: Springer, 1996.

R. Duda, P. Hart, D. Stork. Pattern Classification, Second edition, Wiley-Interscience, 2001.

L. Györfi, G. Lugosi, G. Morvai, *A Simple Randomized Algorithm for Sequential Prediction of Ergodic Time Series.* IEEE Transactions on Information Theory, Vol. 45, pp. 2642–2650, 1999.

D. Helmbold and P. Long, *Tracking Drifting Concepts by Minimizing Disagreements.* Proceedings of the Fourth Annual Workshop on Computational Learning Theory, Santa Cruz, USA, pp. 13–23, 1991.

D. Gamarnik, *Extension of the PAC Framework to Finite and Countable Markov Chains.* IEEE Transactions on Information Theory, 49(1):338-345, 2003.

M. Kearns and D. Ron, *Algorithmic Stability and Sanity-Check Bounds on Leave-One-Out Cross-Validation*. Neural Computation, Vol. 11, No. 6, pp. 1427–1453, 1999.

M. Kearns M. and U. Vazirani, An Introduction to Computational Learning Theory. The MIT Press, Cambridge, Massachusetts, 1994.

S. Kulkarni, S. Posner. *Rates of Convergence of Nearest Neighbour Estimation Under Arbitrary Sampling*. IEEE Transactions on Information Theory, Vol. 41, No. 10, pp. 1028–1039, 1995.

S. Kulkarni, S. Posner, S. Sandilya. *Data-Dependent k_n-NN and Kernel Estimators Consistent for Arbitrary Processess*. IEEE Transactions on Information Theory, Vol. 48, No. 10, pp. 2785–2788, 2002.

G. Lugosi, A. Nobel, *Consistency of Data-Driven Histogram Methods for Density Estimation and Classification.* Annals of Statistics Vol. 24, No.2, pp. 687–706, 1996.

G. Lugosi and K. Zeger, *Nonparametric Estimation via Empirical Risk Minimization.* IEEE Transactions on Information Theory, Vol. 41 No. 3 pp. 677–687, 1995.

G. Morvai, S. Kulkarni, and A.B. Nobel, *Regression Estimation from an Individual Stable Sequence.* Statistics, vol. 33, pp.99–118, 1999.

G. Morvai, S. Yakowitz, P. Algoet, *Weakly Convergent Nonparametric Forecasting of Stationary Time Series.* IEEE Transactions on Information Theory, Vol. 43, No. 2, 1997.

A.B. Nobel, *Limits to Classification and Regression Estimation from Ergodic Process.* Annals of Statistics, vol. 27, pp. 262–273, 1999.

W. Rogers and T. Wagner. *A finite Sample Distribution-Free Performance Bound for Local Discrimination Rules.* Annals of Statistics, Vol. 6 No. 3 pp. 506–514, 1978.

B. Ryabko, *Prediction of random sequences and universal coding.* Problems of Information Transmission, Vol. 24, pp. 87–96, 1988.

D. Ryabko, *Online Learning of Conditionally I.I.D. Data.* In: C. E. Brodley (Ed.), Proceedings of the 21^{st} International Conference on Machine Learning, Banff, Canada, pp. 727–734, 2004.

D. Ryabko, *Application of Classical Nonparametric Predictors to Learning Conditionally I.I.D. Data*. In: S. Ben-David, J. Case, A. Maruoka (Eds.), Proceedings of 15th International Conference on Algorithmic Learning Theory, Padova, Italy, pp. 171–180, 2004.

L. Valiant, *A Theory of the Learnable.* Communications of the ACM, 27, pp. 1134–1142. 1984.

V. Vapnik, Statistical Learning Theory, New York etc.: John Wiley , Sons, Inc. 1998.

V. Vapnik, and A. Chervonenkis. Theory of Pattern Recognition. Nauka, Moscow, 1974 (in Russian).

Journal of Machine Learning Research 7 (2006) 665–704 Submitted 9/05; Published 4/06

Learning Minimum Volume Sets

Clayton D. Scott CSCOTT@RICE.EDU
Department of Statistics
Rice University
Houston, TX 77005, USA

Robert D. Nowak NOWAK@ECE.WISC.EDU
Department of Electrical and Computer Engineering
University of Wisconsin at Madison
Madison, WI 53706, USA

Editor: John Lafferty

Abstract

Given a probability measure P and a reference measure μ, one is often interested in the minimum μ-measure set with P-measure at least α. Minimum volume sets of this type summarize the regions of greatest probability mass of P, and are useful for detecting anomalies and constructing confidence regions. This paper addresses the problem of estimating minimum volume sets based on independent samples distributed according to P. Other than these samples, no other information is available regarding P, but the reference measure μ is assumed to be known. We introduce rules for estimating minimum volume sets that parallel the empirical risk minimization and structural risk minimization principles in classification. As in classification, we show that the performances of our estimators are controlled by the rate of uniform convergence of empirical to true probabilities over the class from which the estimator is drawn. Thus we obtain finite sample size performance bounds in terms of VC dimension and related quantities. We also demonstrate strong universal consistency, an oracle inequality, and rates of convergence. The proposed estimators are illustrated with histogram and decision tree set estimation rules.

Keywords: minimum volume sets, anomaly detection, statistical learning theory, uniform deviation bounds, sample complexity, universal consistency

1. Introduction

Given a probability measure P and a reference measure μ, the minimum volume set (MV-set) with mass at least $0 < \alpha < 1$ is

$$G_\alpha^* = \arg\min\{\mu(G) : P(G) \geq \alpha, G \text{ measurable}\}.$$

MV-sets summarize regions where the mass of P is most concentrated. For example, if P is a multivariate Gaussian distribution and μ is the Lebesgue measure, then the MV-sets are ellipsoids. An MV-set for a two-component Gaussian mixture is illustrated in Figure 1. Applications of minimum volume sets include outlier/anomaly detection, determining highest posterior density or multivariate confidence regions, tests for multimodality, and clustering. See Polonik (1997); Walther (1997); Schölkopf et al. (2001) and references therein for additional applications.

This paper considers the problem of MV-set estimation using a training sample drawn from P, which in most practical settings is the only information one has about P. The specifications to

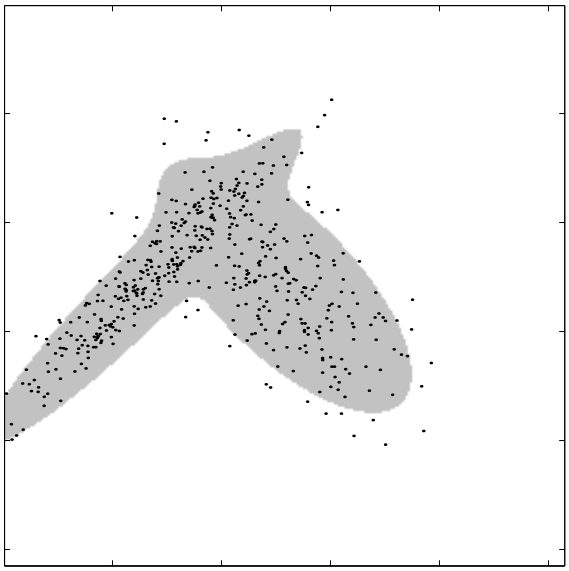


Figure 1: Minimum volume set (gray region) of a two-component Gaussian mixture. Also shown are 500 points drawn independently from this distribution.

the estimation process are the significance level α, the reference measure μ, and a collection of candidate sets $\mathcal{G}$.

A major theme of this work is the strong parallel between MV-set estimation and binary classification. In particular, we find that uniform convergence (of true probability to empirical probability over the class of sets $\mathcal{G}$) plays a central role in controlling the performance of MV-set estimators. Thus, we derive distribution free finite sample performance bounds in terms of familiar quantities such as VC dimension. In fact, as we will see, any uniform convergence bound can be directly converted to a rule for MV-set estimation.

In Section 2 we introduce a rule for MV-set estimation analogous to empirical risk minimization in classification, and shows that this rule obeys similar finite sample size performance guarantees. Section 3 extends the results of the previous section to allow $\mathcal{G}$ to grow in a controlled way with sample size, leading to MV-set estimators that are strongly universally consistent. Section 4 introduces an MV-set estimation rule similar in spirit to structural risk minimization in classification, and develops an oracle-type inequality for this estimator. The oracle inequality guarantees that the estimator automatically adapts its complexity to the problem at hand. Section 5 introduces a tuning parameter to the proposed rules that allows the user to affect the tradeoff between volume error and mass error without sacrificing theoretical properties. Section 6 provides a "case study" of tree-structured set estimators to illustrate the power of the oracle inequality for deriving rates of convergence. Section 7 includes a set of numerical experiments that explores the proposed theory (and algorithmic issues) using histogram and decision tree rules in two dimensions. Section 8 includes concluding remarks and avenues for potential future investigations. Detailed proofs of the main results of the paper are relegated to the appendices. Throughout the paper, the theoretical results are illustrated in detail through several examples, including VC classes, histograms, and decision trees.

1.1 Previous Work

All previous theoretical work on MV-set estimation has been asymptotic in nature, to our knowledge. Our work here is the first to provide explicit finite sample bounds. Most closely related to this paper is the pioneering work of Polonik (1997). Using empirical process theory, he establishes consistency results and rates of convergence for minimum volume sets which depend on the entropy of the class of candidate sets. This places restrictions on the MV-set G_α^* (e.g, $\mu(G_\alpha^*)$ is continuous in α), whereas our consistency result holds universally, i.e., for all distributions P. Also, the convergence rates obtained by Polonik apply under smoothness assumptions on the density. In contrast, our rate of convergence results in Section 6 depend on the smoothness of the boundary of G_α^*. Walther (1997) studies an approach based on "granulometric smoothing," which involves applying certain morphological smoothing operations to the α-mass level set of a kernel density estimate. His rates also apply under smoothness assumptions on the density, rather than more direct assumptions regarding the smoothness of the MV-set as in our approach.

Algorithms for MV-set estimation have been developed for convex sets (Sager, 1979) and ellipsoidal sets (Hartigan, 1987) in two dimensions. Unfortunately, for more complicated problems (dimension > 2 and non-convex sets), there has been a disparity between practical MV-set estimators and theoretical results. Polonik (1997) makes no comment on the practicality of his estimators. The smoothing estimators of Walther (1997) in practice must approximate the theoretical estimator via iterative level set estimation. On the other hand, computationally efficient procedures like those in Schölkopf et al. (2001) and Huo and Lu (2004) are motivated by the minimum volume set paradigm, but their performance relative to G_α^* is not known. Recently, however, Muñoz and Moguerza (2006) have proposed the so-called one-class neighbor machine and demonstrated its consistency under certain assumptions. Our proposed algorithms for histograms and decision trees are practical in low dimensional settings, but appear to be constrained by the same computational limitations as empirical risk minimization in binary classification.

More broadly, MV-set estimation theory has similarities (in terms of the nature of results and technical devices) to other set estimation problems, such as classification, discrimination analysis, density support estimation (which corresponds to the case $\alpha = 1$), and density level set estimation, to which we now turn.

1.2 Connection to Density Level Sets

The MV-set estimation problem is closely related to density level set estimation (Tsybakov, 1997; Ben-David and Lindenbaum, 1997; Cuevas and Rodriguez-Casal, 2003; Steinwart et al., 2005; Vert and Vert, 2005) and excess mass estimation problems (Nolan, 1991; Müller and Sawitzki, 1991; Polonik, 1995). Indeed, it is well known that density level sets are minimum volume sets (Nunez-Garcia et al., 2003). The main difference between density level sets and MV-sets is that the former require the specification of a density level of interest, rather than the specification of the mass α to be enclosed. Since the density is in general unknown, it seems that specifying α is much more reasonable and intuitive than setting a density level for problems like anomaly detection. Suppose for example that one is interested in a reference measure of the form $c\mu$, where μ is Lebesgue measure and $c > 0$. The choice of c does not change the minimum volume set, but it does affect the γ level set. Since there is no way a priori to choose the best c, the invariance of the minimum volume set seems highly desirable. To frame the same issue in a different way, suppose μ is uniform on some set containing the support of P. Then MV-sets are invariant to how the support of μ is

specified, while density level sets are not. Further advantages of MV-sets over level sets are given in the concluding section.

Algorithms for density level set estimation can be split into two categories, implicit plug-in methods and explicit set estimation methods. Plug-in strategies entail full density estimation and are the more popular practical approach. For example, Baillo et al. (2001) considers plug-in rules for density level set estimation problems and establishes upper bounds on the rate of convergence for such estimators in certain cases. The problem of estimating a density support set, the zero level set, is a special minimum volume set (i.e., the minimum volume set that contains the total probability mass). Cuevas and Fraiman (1997) study density support estimation and show that a certain (density estimator) plug-in scheme provides universally consistent support estimation.

While consistency and rate of convergence results for plug-in methods typically make global smoothness assumptions on the density, explicit methods make assumptions on the density at or near the level of interest. This fact, together with the intuitive appeal of not having to solve a problem harder than one is interested in, make explicit methods attractive. Steinwart et al. (2005) reduce level set estimation to a cost-sensitive classification problem by sampling from the reference measure. The idea of sampling from μ in the minimum volume context is discussed further in the concluding section. Vert and Vert (2005) study the one-class support vector machine (SVM) and show that it produces a consistent density level set estimator, based on the fact that consistent density estimators produce consistent plug-in level set estimators. Willett and Nowak (2005, 2006) propose a level set estimator based on decision trees, which is applicable to density level set estimation as well as regression level set estimation, and related dyadic partitioning schemes are developed by Klemelä (2004) to estimate the support set of a density.

The connections between MV-sets and density level sets will be important later in this paper. To make the connection precise the following assumption on the data-generating distribution and reference measure is needed. We emphasize that this assumption is not necessary for the results in Sections 2 and 3, where distribution free error bounds and universal consistency are established.

A1 P has a density f with respect to μ.

A key result relating density level and MV-sets is the following, stated without proof (see, e.g., Nunez-Garcia et al. (2003)).

Lemma 1 *Under assumption* **A1** *there exists* γ_α *such that for any MV-set* G^*_α,

$$\{x : f(x) > \gamma_\alpha\} \subset G^*_\alpha \subset \{x : f(x) \geq \gamma_\alpha\}.$$

Note that every density level set is an MV-set, but not conversely. If, however, $\mu(\{x : f(x) = \gamma_\alpha\}) = 0$, then the three sets in the Lemma coincide.

1.3 Notation

Let $(\mathcal{X}, \mathcal{B})$ be a measure space with $\mathcal{X} \subset \mathbb{R}^d$. Let X be a random variable taking values in $\mathcal{X}$ with distribution P. Let $S = (X_1, \ldots, X_n)$ be an independent and identically distributed (IID) sample drawn according to P. Let G denote a subset of $\mathcal{X}$, and let $\mathcal{G}$ be a collection of such subsets. Let $\widehat{P}$ denote the empirical measure based on S:

$$\widehat{P}(G) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(X_i \in G).$$

Here $\mathbb{I}(\cdot)$ is the indicator function. The notation μ will denote a measure[1] on $\mathcal{X}$. Denote by f the density of P with respect to μ (when it exists), $\gamma > 0$ a level of the density, and $\alpha \in (0,1)$ a user-specified mass constraint. Define

$$\mu^*_\alpha = \inf_G \{\mu(G) : P(G) \geq \alpha\}, \tag{1}$$

where the inf is over all measurable sets. A minimum volume set, G^*_α, is a minimizer of (1) when it exists.

2. Minimum Volume Sets and Empirical Risk Minimization

We introduce a procedure inspired by the empirical risk minimization (ERM) principle for classification. In classification, ERM selects a classifier from a fixed set of classifiers by minimizing the empirical error (risk) of a training sample. Vapnik and Chervonenkis established the basic theoretical properties of ERM (see Vapnik, 1998; Devroye et al., 1996), and we find similar properties in the minimum volume setting.

Let $\mathcal{G}$ be a class of sets. Given $\alpha \in (0,1)$, denote

$$\mathcal{G}_\alpha \quad = \quad \{G \in \mathcal{G} : P(G) \geq \alpha\},$$

the collection of all sets in $\mathcal{G}$ with mass at least α. Define

$$\mu_{\mathcal{G},\alpha} = \inf\{\mu(G) : G \in \mathcal{G}_\alpha\} \tag{2}$$

and

$$G_{\mathcal{G},\alpha} = \arg\min\{\mu(G) : G \in \mathcal{G}_\alpha\} \tag{3}$$

when it exists. Thus $G_{\mathcal{G},\alpha}$ is the best approximation to the minimum volume set G^*_α from $\mathcal{G}$.

Empirical versions of $\mathcal{G}_\alpha$ and $G_{\mathcal{G},\alpha}$ are defined as follows. Let $\phi(G,S,\delta)$ be a function of $G \in \mathcal{G}$, the training sample S, and a confidence parameter $\delta \in (0,1)$. Set

$$\widehat{\mathcal{G}}_\alpha = \{G \in \mathcal{G} : \widehat{P}(G) \geq \alpha - \phi(G,S,\delta)\}$$

and

$$\widehat{G}_{\mathcal{G},\alpha} = \arg\min\{\mu(G) : G \in \widehat{\mathcal{G}}_\alpha\}. \tag{4}$$

We refer to the rule in (4) as MV-ERM because of the analogy with empirical risk minimization in classification. A discussion of the existence and uniqueness of the above quantities is deferred to Section 2.5.

The quantity ϕ acts as a kind of "tolerance" by which the empirical mass may deviate from the targeted value α. Throughout this paper we assume that ϕ satisfies the following.

Definition 2 *We say ϕ is a (distribution free)* complexity penalty *for $\mathcal{G}$ if and only if for all distributions P and all $\delta \in (0,1)$,*

$$P^n\left(\left\{S : \sup_{G \in \mathcal{G}} \left(\left|P(G) - \widehat{P}(G)\right| - \phi(G,S,\delta)\right) > 0\right\}\right) \leq \delta.$$

1. Although we do not emphasize it, the results of Sections 2 and 3 only require μ to be a real-valued function on $\mathcal{B}$.

Thus, ϕ controls the rate of uniform convergence of $\widehat{P}(G)$ to $P(G)$ for $G \in \mathcal{G}$. It is well known that the performance of ERM (for binary classification) relative to the performance of the best classifier in the given class is controlled by the uniform convergence of true to empirical probabilities. A similar result holds for MV-ERM.

Theorem 3 *If ϕ is a complexity penalty for $\mathcal{G}$, then*

$$P^n\left(\left(P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha - 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta)\right) \text{ or } \left(\mu(\widehat{G}_{\mathcal{G},\alpha}) > \mu_{\mathcal{G},\alpha}\right)\right) \leq \delta.$$

Proof Consider the sets

$$\begin{aligned}
\Theta_P &= \{S : P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha - 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta)\}, \\
\Theta_\mu &= \{S : \mu(\widehat{G}_{\mathcal{G},\alpha}) > \mu(G_{\mathcal{G},\alpha})\}, \\
\Omega_P &= \left\{S : \sup_{G \in \mathcal{G}} \left(\left|P(G) - \widehat{P}(G)\right| - \phi(G, S, \delta)\right) > 0\right\}.
\end{aligned}$$

Lemma 4 *With Θ_P, Θ_μ, and Ω_P as defined above we have*

$$\Theta_P \cup \Theta_\mu \subset \Omega_P.$$

The proof is given in Appendix A, and follows closely the proof of Lemma 1 in Cannon et al. (2002). The theorem statement follows directly from this observation. ■

Lemma 4 may be understood by analogy with the result from classification that says $\mathcal{R}(\widehat{f}) - \inf_{f \in \mathcal{F}} \mathcal{R}(f) \leq 2\sup_{f \in \mathcal{F}} |\mathcal{R}(f) - \widehat{\mathcal{R}}(f)|$ (see Devroye et al. (1996), Ch. 8). Here $\mathcal{R}$ and $\widehat{\mathcal{R}}$ are the true and empirical risks, $\widehat{f}$ is the empirical risk minimizer, and $\mathcal{F}$ is a set of classifiers. Just as this result relates uniform convergence to empirical risk minimization in classification, so does Lemma 4 relate uniform convergence to the performance of MV-ERM.

The theorem above allows direct translation of uniform convergence results into performance guarantees on MV-ERM. Fortunately, many penalties (uniform convergence results) are known. In the next two subsections we take a closer look at penalties for VC classes and countable classes, and a Rademacher penalty.

2.1 Example: VC Classes

Let $\mathcal{G}$ be a class of sets with VC dimension V, and define

$$\phi(G, S, \delta) = \sqrt{32\frac{V\log n + \log(8/\delta)}{n}}. \tag{5}$$

By a version of the VC inequality (Devroye et al., 1996), we know that ϕ is a complexity penalty for $\mathcal{G}$, and therefore Theorem 3 applies.

To view this result in perhaps a more recognizable way, let $\varepsilon > 0$ and choose δ such that $\phi(G, S, \delta) = \varepsilon$ for all $G \in \mathcal{G}$ and all S. By inverting the relationship between δ and ε, we have the following.

Corollary 5 *With the notation defined above,*

$$P^n\left(\left(P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha - 2\varepsilon\right) or \left(\mu(\widehat{G}_{\mathcal{G},\alpha}) > \mu_{\mathcal{G},\alpha}\right)\right) \leq 8n^V e^{-n\varepsilon^2/128}.$$

Thus, for any fixed $\varepsilon > 0$, the probability of being within 2ε of the target mass α and being less than the target volume $\mu_{\mathcal{G},\alpha}$ approaches one exponentially fast as the sample size increases. This result may also be used to calculate a distribution free upper bound on the sample size needed to be within a given tolerance ε of α and with a given confidence $1-\delta$. In particular, the sample size will grow no faster than a polynomial in $1/\varepsilon$ and $1/\delta$, paralleling results for classification.

2.2 Example: Countable Classes

Suppose $\mathcal{G}$ is a countable class of sets. Assume that to every $G \in \mathcal{G}$ a number $[\![G]\!]$ is assigned such that

$$\sum_{G\in\mathcal{G}} 2^{-[\![G]\!]} \leq 1. \tag{6}$$

In light of the Kraft inequality for prefix[2] codes (Cover and Thomas, 1991), $[\![G]\!]$ may be defined as the codelength of a codeword for G in a prefix code for $\mathcal{G}$. Let $\delta > 0$ and define

$$\phi(G,S,\delta) = \sqrt{\frac{[\![G]\!]\log 2 + \log(2/\delta)}{2n}}. \tag{7}$$

By Chernoff's bound together with the union bound, ϕ is a penalty for $\mathcal{G}$. Therefore Theorem 3 applies and we have a result analogous to the Occam's Razor bound for classification (see Langford, 2005).

As a special case, suppose $\mathcal{G}$ is finite and take $[\![G]\!] = \log_2|\mathcal{G}|$. Setting $\varepsilon = \phi(G,S,\delta)$ and inverting the relationship between δ and ε, we have the following.

Corollary 6 *For the MV-ERM estimate $\widehat{G}_{\mathcal{G},\alpha}$ from a finite class $\mathcal{G}$*

$$P^n\left(\left(P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha - 2\varepsilon\right) or \left(\mu(\widehat{G}_{\mathcal{G},\alpha}) > \mu_{\mathcal{G},\alpha}\right)\right) \leq 2|\mathcal{G}|e^{-n\varepsilon^2/2}.$$

As with VC classes, these inequalities may be used for sample size calculations.

2.3 The Rademacher Penalty for Sets

The Rademacher penalty was originally studied in the context of classification by Koltchinskii (2001) and Bartlett et al. (2002). For a succinct exposition of its basic properties, see Bousquet et al. (2004). An analogous penalty exists for sets. Let $\sigma_1,\ldots,\sigma_n$ be Rademacher random variables, i.e., independent random variables taking on the values 1 and -1 with equal probability. Denote $\widehat{P}_{(\sigma_i)}(G) = \frac{1}{n}\sum_{i=1}^n \sigma_i \mathbb{I}(X_i \in G)$. We define the Rademacher average

$$\rho(\mathcal{G}) = \mathbf{E}\left[\sup_{G\in\mathcal{G}} \widehat{P}_{(\sigma_i)}(G)\right]$$

2. A prefix code is a collection of codewords (strings of 0s and 1s) such that no codeword is a prefix of another.

and the conditional Rademacher average

$$\widehat{\rho}(\mathcal{G},S) = \mathbf{E}_{(\sigma_i)}\left[\sup_{G\in\mathcal{G}} \widehat{P}_{(\sigma_i)}(G)\right],$$

where the second expectation is with respect the Rademacher random variables only, and conditioned on the sample S.

Proposition 7 *With probability at least* $1-\delta$ *over the draw of* S*,*

$$P(G) - \widehat{P}(G) \le 2\rho(\mathcal{G}) + \sqrt{\frac{\log(1/\delta)}{2n}}$$

for all $G \in \mathcal{G}$*. With probability at least* $1-\delta$ *over the draw of* S*,*

$$P(G) - \widehat{P}(G) \le 2\widehat{\rho}(\mathcal{G},S) + \sqrt{\frac{2\log(2/\delta)}{n}}$$

for all $G \in \mathcal{G}$*.*

The proof of this result follows exactly the same lines as the proof of Theorem 5 in Bousquet et al. (2004), and is omitted.

Assume $\mathcal{G}$ satisfies the property that $G \in \mathcal{G} \Rightarrow \overline{G} \in \mathcal{G}$, where $\overline{G}$ denotes the compliment of G. Then $\widehat{P}(G) - P(G) = P(\overline{G}) - \widehat{P}(\overline{G})$, and so the upper bounds of Proposition 7 also apply to $|P(G) - \widehat{P}(G)|$. Thus we are able to define the conditional Rademacher penalty

$$\phi(G,S,\delta) = 2\widehat{\rho}(\mathcal{G},S) + \sqrt{\frac{2\log(2/\delta)}{n}}.$$

By the above Proposition, this is a complexity penalty according to Definition 2. The conditional Rademacher penalty is studied further in Section 7 and in Appendix E, where it is shown that $\widehat{\rho}(\mathcal{G},S)$ can be computed efficiently for sets based on a fixed partition of $\mathcal{X}$ (such as histograms and trees).

2.4 Comparison to Generalized Quantile Processes

Polonik (1997) studies the *empirical quantile function*

$$\widehat{V}_\alpha = \inf\{\mu(G) : \widehat{P}(G) \ge \alpha\},$$

and the MV-set estimate that achieves the minimum (when it exists). The only difference compared with MV-ERM is the absence of the term $\phi(G,S,\delta)$ in the constraint. Thus, MV-ERM will tend to produce estimates with smaller volume and smaller mass. While Polonik proves only asymptotic properties of his estimator, we have demonstrated finite sample bounds for MV-ERM. Moreover, in Section 5, we show that the results of this section extend to a generalization of MV-ERM where ϕ is replaced by $\nu\phi$, where ν is any number $-1 \le \nu \le 1$. Thus finite sample bounds also exist for Polonik's estimator ($\nu = 0$).

2.5 Existence and Uniqueness

In this section we discuss the existence and uniqueness of the sets $G_{\mathcal{G},\alpha}$ and $\widehat{G}_{\mathcal{G},\alpha}$. Regarding the former, it is really not necessary that a minimizer exist. All of our results are stated in terms of $\mu_{\mathcal{G},\alpha}$, which certainly exists. When a minimizer exists, its uniqueness is not an issue for the same reason. Our results above involve only $\mu_{\mathcal{G},\alpha}$, which is the same regardless of which minimizer is chosen. Yet one may wonder whether convergence of the volume and mass to their optimal values implies convergence to the MV-set (when it is unique) in any sense. A result in this direction is presented in Theorem 10 below.

For the MV-ERM estimate $\widehat{G}_{\mathcal{G},\alpha}$, uniqueness is again not an issue because all results hold even if the minimizer is chosen arbitrarily. As for existence, we must be more careful. We cannot make the same argument as for $G_{\mathcal{G},\alpha}$ because we are ultimately interested in a concrete set estimate, not just its volume and mass. Clearly, if $\mathcal{G}$ is finite, $\widehat{G}_{\mathcal{G},\alpha}$ exists. For more general sets, existence must be examined on a case-by-case basis. For example, if $\mathcal{X} \subset \mathbb{R}^d$, μ is the Lebesgue measure, and $\mathcal{G}$ is the VC class of spherical or ellipsoidal sets, then $\widehat{G}_{\mathcal{G},\alpha}$ can be seen to exist.

In the event that $\widehat{G}_{\mathcal{G},\alpha}$ does not exist, it suffices to let $\widehat{G}_{\mathcal{G},\alpha}$ be a set whose volume comes within ε of the infimum, where ε is arbitrarily small. Then our results still hold with $\mu(\widehat{G}_{\mathcal{G},\alpha})$ replaced by $\mu(\widehat{G}_{\mathcal{G},\alpha}) - \varepsilon$. The consistency and rate of convergence results below are unchanged, as we may take $\varepsilon \to 0$ arbitrarily fast as a function of n.

3. Consistency

A minimum volume set estimator is consistent if its volume and mass tend to the optimal values μ^*_α and α as $n \to \infty$. Formally, define the error quantity

$$\mathcal{E}(G) := (\mu(G) - \mu^*_\alpha)_+ + (\alpha - P(G))_+ ,$$

where $(x)_+ = \max(x, 0)$. We are interested in MV-set estimators such that $\mathcal{E}(\widehat{G}_{\mathcal{G},\alpha})$ tends to zero as $n \to \infty$.

Definition 8 *A learning rule $\widehat{G}_{\mathcal{G},\alpha}$ is* strongly consistent *if*

$$\lim_{n\to\infty} \mathcal{E}(\widehat{G}_{\mathcal{G},\alpha}) = 0 \quad \textit{with probability 1.}$$

If $\widehat{G}_{\mathcal{G},\alpha}$ is strongly consistent for every possible distribution of X, then $\widehat{G}_{\mathcal{G},\alpha}$ is strongly universally *consistent.*

In this section we show that if the approximating power of $\mathcal{G}$ increases in a certain way as a function of n, then MV-ERM leads to a universally consistent learning rule.

To see how consistency might result from MV-ERM, it helps to rewrite Theorem 3 as follows. Let $\mathcal{G}$ be fixed and let $\phi(G, S, \delta)$ be a penalty for $\mathcal{G}$. Then with probability at least $1 - \delta$, both

$$\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha \le \mu(G_{\mathcal{G},\alpha}) - \mu^*_\alpha \tag{8}$$

and

$$\alpha - P(\widehat{G}_{\mathcal{G},\alpha}) \le 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta) \tag{9}$$

hold. We refer to the left-hand side of (8) as the *excess volume* of the class $\mathcal{G}$ and the left-hand side of (9) as the *missing mass* of $\widehat{G}_{\mathcal{G},\alpha}$. The upper bounds on the right-hand sides are an approximation error and a stochastic error, respectively.

The idea is to let $\mathcal{G}$ grow with n so that both errors tend to zero as $n \to \infty$. If $\mathcal{G}$ does not change with n, universal consistency is impossible. Either the approximation error will be nonzero for most distributions (when $\mathcal{G}$ is too small) or the bound on the stochastic error will be too large (otherwise). For example, if a class has universal approximation capabilities, its VC dimension is necessarily infinite (Devroye et al., 1996, Ch. 18).

To have both stochastic and approximation errors tend to zero, we apply MV-ERM to a class $\mathcal{G}^k$ from a sequence of classes $\mathcal{G}^1, \mathcal{G}^2, \ldots$, where $k = k(n)$ grows with the sample size. Given such a sequence, define

$$\widehat{G}_{\mathcal{G}^k,\alpha} \quad = \quad \arg\min\{\mu(G) : G \in \widehat{\mathcal{G}}_\alpha^k\}, \tag{10}$$

where

$$\widehat{\mathcal{G}}_\alpha^k = \{G \in \mathcal{G}^k : \widehat{P}(G) \geq \alpha - \phi_k(G,S,\delta)\}$$

and ϕ_k is a penalty for $\mathcal{G}^k$.

Theorem 9 *Choose $k = k(n)$ and $\delta = \delta(n)$ such that*

1. *$k(n) \to \infty$ as $n \to \infty$*
2. *$\sum_{n=1}^{\infty} \delta(n) < \infty$*

Assume the sequence of sets $\mathcal{G}^k$ and penalties ϕ_k satisfy

$$\lim_{k\to\infty} \inf_{G \in \mathcal{G}_\alpha^k} \mu(G) = \mu_\alpha^* \tag{11}$$

and

$$\lim_{n\to\infty} \sup_{G \in \mathcal{G}^k} \phi_k(G,S,\delta(n)) = 0. \tag{12}$$

Then $\widehat{G}_{\mathcal{G}^k,\alpha}$ is strongly universally consistent.

The proof is given in Appendix B. We now give some examples that satisfy these conditions.

3.1 Example: Hierarchy of VC Classes

Assume $\mathcal{G}^1, \mathcal{G}^2, \ldots,$ is a family of VC classes with VC dimensions $V_1 < V_2 < \ldots$. For $G \in \mathcal{G}^k$ define

$$\phi_k(G,S,\delta) = \sqrt{32\frac{V_k \log n + \log(8/\delta)}{n}}. \tag{13}$$

By taking $\delta(n) \asymp n^{-\beta}$ for some $\beta > 1$ and k such that $V_k = o(n/\log n)$ the assumption in (12) is satisfied. Examples of families of VC classes satisfying (11) include generalized linear discriminant rules with appropriately chosen basis functions and neural networks (Lugosi and Zeger, 1995).

3.2 Example: Histograms

Assume $\mathcal{X} = [0,1]^d$, and let $\mathcal{G}^k$ be the class of all sets formed by taking unions of cells in a regular partition of $\mathcal{X}$ into hypercubes of sidelength $1/k$. Each $\mathcal{G}^k$ has 2^{k^d} members and we may therefore apply the penalty for finite sets discussed in Section 2.2. To satisfy the Kraft inequality (6) it suffices to take $[\![G]\!] = k^d$. The penalty for $G \in \mathcal{G}^k$ is then

$$\phi_k(G,S,\delta) = \sqrt{\frac{k^d \log 2 + \log(2/\delta)}{2n}}. \tag{14}$$

By taking $\delta(n) \asymp n^{-\beta}$ for some $\beta > 1$ and k such that $k^d = o(n)$ the assumption in (12) is satisfied. The assumption in (11) is satisfied by the well-known universal approximation capabilities of histograms. Thus the conditions for consistency of histograms for minimum volume set estimation are exactly parallel to the conditions for consistency of histogram rules for classification (Devroye et al., 1996, Ch. 9). Dyadic decision trees, discussed below in Section 6, are another countable family for which consistency results are possible.

3.3 The Symmetric Difference Performance Metric

An alternative measure of performance for an MV-set estimator is the μ-measure of the symmetric difference, $\mu(\widehat{G}_{\mathcal{G},\alpha} \Delta G^*_\alpha)$, where $A \Delta B = (A \backslash B) \cup (B \backslash A)$. Although this performance metric has been commonly adopted in the study of density level sets, it is less desirable for our purposes. First, unlike with density level sets, there may not be a unique MV-set (imagine the case where the density of P has a plateau). Second, as pointed out by Steinwart et al. (2005), there is no known way to estimate the accuracy of this measure using only samples from P. Nonetheless, the symmetric difference metric coincides asymptotically with our error metric $\mathcal{E}$ in the sense of the following result. The theorem uses the notation γ_α to denote the density level corresponding to the MV-set, as discussed in Section 1.2.

Theorem 10 *Assume μ is a probability measure and P has a density f with respect to μ. Let G_n denote a sequence of sets. If G^*_α is a minimum volume set and $\mu(G_n \Delta G^*_\alpha) \to 0$ with n, then $\mathcal{E}(G_n) \to 0$. Conversely, assume $\mu(\{x : f(x) = \gamma_\alpha\}) = 0$. If $\mathcal{E}(G_n) \to 0$, then $\mu(G_n \Delta G^*_\alpha) \to 0$.*

The proof is given in Appendix C. The assumption of the second part of the theorem ensures that G^*_α is unique, otherwise the converse statement need not be true. The proof of the converse reveals yet another connection between MV-set estimation and classification. In particular, we show that $\mathcal{E}(G_n)$ bounds the excess classification risk for a certain classification problem. The converse statement then follows from a result of Steinwart et al. (2005) who show that this excess classification risk and the μ-measure of the symmetric difference tend to zero simultaneously.

4. Structural Risk Minimization and an Oracle Inequality

In the previous section on consistency the rate of convergence of the two errors to zero is determined by the choice of $k = k(n)$, which must be chosen a priori. Hence it is possible that the excess volume decays much more quickly than the missing mass, or vice versa. In this section we introduce a new rule called MV-SRM, inspired by the principle of structural risk minimization (SRM) from the theory of classification (Vapnik, 1982; Lugosi and Zeger, 1996), that automatically balances the two errors.

The results of this and subsequent sections are no longer distribution free. In particular, we assume

A1 P has a density f with respect to μ.

A2 for all $\alpha' \in (0,1)$, $G^*_{\alpha'}$ exists and $P(G^*_{\alpha'}) = \alpha'$.

Note that **A2** holds if f has no plateaus, i.e., $\mu(\{x : f(x) = \gamma\}) = 0$ for all $\gamma > 0$. This is a commonly made assumption in the study of density level sets. However, **A2** is somewhat more general. It still holds, for example, if μ is absolutely continuous with respect to Lebesgue measure, even if f has plateaus.

Recall from Section 1.2 that under assumption **A1**, there exists $\gamma_\alpha > 0$ such for any MV-set G^*_α,

$$\{x : f(x) > \gamma_\alpha\} \subset G^*_\alpha \subset \{x : f(x) \geq \gamma_\alpha\}.$$

Let $\mathcal{G}$ be a class of sets. Intuitively, view $\mathcal{G}$ as a collection of sets of varying capacities, such as a union of VC classes or a union of finite classes (examples are given below). Let $\phi(G,S,\delta)$ be a penalty for $\mathcal{G}$. The MV-SRM principle selects the set

$$\widehat{G}_{\mathcal{G},\alpha} \quad = \quad \arg\min_{G \in \mathcal{G}} \left\{ \mu(G) + 2\phi(G,S,\delta) : \widehat{P}(G) \geq \alpha - \phi(G,S,\delta) \right\}. \tag{15}$$

Note that MV-SRM is different from MV-ERM because it minimizes a complexity penalized volume instead of simply the volume. We have the following oracle inequality for MV-SRM. Recall $\mathcal{E}(G) := (\mu(G) - \mu^*_\alpha)_+ + (\alpha - P(G))_+$.

Theorem 11 *Let $\widehat{G}_{\mathcal{G},\alpha}$ be the MV-set estimator in (15) and assume* **A1** *and* **A2** *hold. With probability at least $1-\delta$ over the training sample S,*

$$\mathcal{E}(\widehat{G}_{\mathcal{G},\alpha}) \leq \left(1 + \frac{1}{\gamma_\alpha}\right) \inf_{G \in \mathcal{G}_\alpha} \left\{ \ \mu(G) - \mu^*_\alpha + 2\phi(G,S,\delta) \ \right\}. \tag{16}$$

Although the value of $1/\gamma_\alpha$ is in practice unknown, it can be bounded by

$$\frac{1}{\gamma_\alpha} \leq \frac{\mu(X) - \mu^*_\alpha}{1-\alpha} \leq \frac{\mu(X)}{1-\alpha}.$$

This follows from the bound $1 - \alpha \leq \gamma_\alpha \cdot (\mu(X) - \mu^*_\alpha)$ on the mass outside the minimum volume set. If μ is a probability measure, then $1/\gamma_\alpha \leq 1/(1-\alpha)$.

The oracle inequality says that MV-SRM performs about as well as the set chosen by an oracle to optimize the tradeoff between excess volume and missing mass.

4.1 Example: Union of VC Classes

Consider $\mathcal{G} = \cup_{k=1}^K \mathcal{G}^k$, where $\mathcal{G}^k$ has VC dimension V_k, $V_1 < V_2 < \cdots$, and K is possibly infinite. A penalty for $\mathcal{G}$ can be obtained by defining, for $G \in \mathcal{G}^k$,

$$\phi(G,S,\delta) = \phi_k(G,S,\delta 2^{-k}),$$

where ϕ_k is the penalty from Equation (13). Then ϕ is a penalty for $\mathcal{G}$ because ϕ_k is a penalty for $\mathcal{G}^k$, and by applying the union bound and the fact $\sum_{k \geq 1} 2^{-k} \leq 1$. In this case, MV-SRM adaptively

selects an MV-set estimate from a VC class that balances approximation and stochastic errors. Note that instead of setting $\delta_k = \delta 2^{-k}$ one could also choose $\delta_k \propto k^{-\beta}, \beta > 1$.

To be more concrete, suppose $\mathcal{G}^k$ is the collection of sets whose boundaries are defined by polynomials of degree k. It may happen that for certain distributions, the MV-set is well-approximated by a quadratic region (such as an ellipse), while for other distributions a higher degree polynomial is required. If the appropriate polynomial degree for the MV-set is not known in advance, as would be the case in practice, then MV-SRM adaptively chooses an estimator of a certain degree that does about as well as if the best degree was known in advance.

4.2 Example: Union of Histograms

Let $\mathcal{G} = \cup_{k=1}^{K} \mathcal{G}^k$, where $\mathcal{G}^k$ is as in Section 3.2. As with VC classes, we obtain a penalty for $\mathcal{G}$ by defining, for $G \in \mathcal{G}^k$,

$$\phi(G,S,\delta) = \phi_k(G,S,\delta 2^{-k}),$$

where ϕ_k is the penalty from Equation (14). Then MV-SRM adaptively chooses a partition resolution k that approximates the MV-set about as well as possible without overfitting the training data. This example is studied experimentally in Section 7.

5. Damping the Penalty

In Theorem 3, the reader may have noticed that MV-ERM does not equitably balance the excess volume ($\mu(\widehat{G}_{\mathcal{G},\alpha})$ relative to its optimal value) with the missing mass ($P(\widehat{G}_{\mathcal{G},\alpha})$ relative to α). Indeed, with high probability, $\mu(\widehat{G}_{\mathcal{G},\alpha})$ is *less than* $\mu(G_{\mathcal{G},\alpha})$, while $P(\widehat{G}_{\mathcal{G},\alpha})$ is only guaranteed to be within $2\phi(\widehat{G}_{\mathcal{G},\alpha})$ of α. The net effect is that MV-ERM (and MV-SRM) underestimates the MV-set. Our experiments in Section 7 demonstrate this to be the case.

In this section we introduce variants of MV-ERM and MV-SRM that allow the total error to be shared between the volume and mass, instead of all of the error residing in the mass term. Our approach is to introduce a damping factor $-1 \leq \nu \leq 1$ that scales the penalty. We will see that the resulting MV-set estimators obey performance guarantees like those we have already seen, but with the total error redistributed between the volume and mass. The reason for not introducing this more general framework initially is that the results are slightly less general, more involved to state, and to some extent follow as corollaries to the original ($\nu = 1$) framework.

The extensions of this section encompass the generalized quantile estimate of Polonik (1997), which corresponds to $\nu = 0$. Thus we have finite sample size guarantees for that estimator to match Polonik's asymptotic analysis. The case $\nu = -1$ is also of interest. If it is crucial that the estimate satisfies the mass constraint $P(\widehat{G}_{\mathcal{G},\alpha}) \geq \alpha$ (note that this involves the *true* probability measure P), setting $\nu = -1$ ensures this to be the case with probability at least $1-\delta$.

First we consider damping the penalty in MV-ERM. Assume that the penalty is independent of $G \in \mathcal{G}$ and of the sample S, although it can depend on n and δ. That is, $\phi(G,S,\delta) = \phi(n,\delta)$. For example, ϕ may be the penalty in (5) for VC classes or (7) for finite classes. Let $\nu \leq 1$ and define

$$\widehat{G}^{\nu}_{\mathcal{G},\alpha} = \arg\min_{G \in \mathcal{G}} \left\{ \mu(G) : \widehat{P}(G) \geq \alpha - \nu\phi(n,\delta) \right\}.$$

Since ϕ is independent of $G \in \mathcal{G}$, $\widehat{G}^{\nu}_{\mathcal{G},\alpha}$ coincides with the MV-ERM estimate (as originally formulated) $\widehat{G}_{\mathcal{G},\alpha'}$ but at the adjusted mass constraint $\alpha' = \alpha + (1-\nu)\phi(n,\delta)$. Therefore, we may apply Theorem 3 to obtain the following.

Corollary 12 *Let* $\alpha' = \alpha + (1-\nu)\phi(n,\delta)$. *Then*

$$P^n\left(\left(P(\widehat{G}^{\nu}_{\mathcal{G},\alpha}) < \alpha - (1+\nu)\phi(n,\delta)\right) \text{ or } \left(\mu(\widehat{G}_{\mathcal{G},\alpha}) > \mu_{\mathcal{G},\alpha'}\right)\right) \leq \delta.$$

Relative to the original formulation of MV-ERM, the bound on the missing mass is decreased by a factor $(1+\nu)$. On the other hand, the volume is now bounded by $\mu_{\mathcal{G},\alpha'} = \mu_{\mathcal{G},\alpha} + (\mu_{\mathcal{G},\alpha'} - \mu_{\mathcal{G},\alpha})$. Thus the bound on the excess volume is increased from 0 to $\mu_{\mathcal{G},\alpha'} - \mu_{\mathcal{G},\alpha}$. This may be interpreted as a stochastic component of the excess volume. Relative to the MV-set, $\mu(\widehat{G}_{\mathcal{G},\alpha})$ has only an approximation error, whereas $\mu(\widehat{G}^{\nu}_{\mathcal{G},\alpha})$ has both approximation and stochastic errors. The advantage is that now the stochastic error of the mass is decreased.

A similar construction applies to MV-SRM. Now assume $\mathcal{G} = \cup_{k=1}^{K} \mathcal{G}^k$. Given a scale parameter ν, define

$$\widehat{G}^{\nu}_{\mathcal{G},\alpha} = \arg\min_{G \in \mathcal{G}} \left\{ \mu(G) + (1+\nu)\phi(G,S,\delta) : \widehat{P}(G) \geq \alpha - \nu\phi(G,S,\delta) \right\}.$$

As above, assume ϕ is independent of the sample and constant on each $\mathcal{G}^k$. Denote $\varepsilon_k(n,\delta) = \phi(G,S,\delta)$ for $G \in \mathcal{G}^k$. Observe that computing $\widehat{G}^{\nu}_{\mathcal{G},\alpha}$ is equivalent to computing the MV-ERM estimate on each $\mathcal{G}^k$ at the level $\alpha(k,\nu) = \alpha + (1-\nu)\varepsilon_k(n,\delta)$, and then minimizing the penalized volume over these MV-ERM estimates.

Like the original MV-SRM, this modified procedure also obeys an oracle inequality. Recall the notation $\mathcal{G}^k_{\alpha(k,\nu)} = \{G \in \mathcal{G}^k : P(G) \geq \alpha(k,\nu)\} = \{G \in \mathcal{G}^k : P(G) \geq \alpha + (1-\nu)\varepsilon_k(n,\delta)\}$.

Theorem 13 *Let* $-1 \leq \nu \leq 1$. *Set* $\alpha(k,\nu) = \alpha + (1-\nu)\varepsilon_k(n,\delta)$. *Assume* **A1** *and* **A2** *hold. With probability at least* $1-\delta$,

$$\mathcal{E}\left(\widehat{G}^{\nu}_{\mathcal{G},\alpha}\right) \leq \left(1 + \frac{1}{\gamma_\alpha}\right) \min_{1 \leq k \leq K} \left[\inf_{G \in \mathcal{G}^k_{\alpha(k,\nu)}} \left\{ \mu(G) - \mu^*_{\alpha(k,\nu)} \right\} + C_k \varepsilon_k(n,\delta) \right], \tag{17}$$

where $C_k = \left((1+\nu) + \frac{1}{\gamma_{\alpha(k,\nu)}}(1-\nu) \right)$.

Here $\gamma_{\alpha(k,\nu)}$ is the density level corresponding to the MV-set with mass $\alpha(k,\nu)$. It may be bounded above in terms of known quantities, as discussed in the previous section. The proof of the theorem is very similar to the proof of the earlier oracle inequality and is omitted, although it may be found in Scott and Nowak (2005a). Notice that in the case $\nu = 1$ we recover Theorem 11 (under the stated assumptions on $\mathcal{G}$ and ϕ). Also note that $\mathcal{G}^k_{\alpha(k,\nu)}$ will be empty if $\alpha(k,\nu) > 1$, in which case those k should be excluded from the min.

To understand the result, assume that the rate at which $\mathcal{G}^k_\alpha$ approximates G^*_α is independent of α. In other words, the rate at which $\inf_{G \in \mathcal{G}^k_\alpha} \mu(G) - \mu^*_\alpha$ tends to zero as k increases is the same for all α. Then in the theorem we may replace the expression $\inf_{G \in \mathcal{G}^k_{\alpha(k,\nu)}} \mu(G) - \mu^*_{\alpha(k,\nu)}$ with $\inf_{G \in \mathcal{G}^k_\alpha} \mu(G) - \mu^*_\alpha$. Thus, the ν-damped MV-SRM error decays at the same rate is the original MV-SRM, and adaptively selects the appropriate model class $\mathcal{G}^k$ from which to draw the estimate. Furthermore, damping the

penalty by ν has the effect of decreasing the stochastic mass error and adding a stochastic error to the volume. This follows from the above discussion of MV-ERM and the observation that the MV-SRM coincides with an MV-SRM estimate over $\mathcal{G}^k$ for some k. The improved balancing of volume and mass error is confirmed by our experiments in Section 7.

6. Rates of Convergence for Tree-Structured Set Estimators

In this section we illustrate the application of MV-SRM, when combined with an appropriate analysis of the approximation error, to the study of rates of convergence. To preview the main result of this section (Theorem 16), we will consider the class of distributions such that the decision boundary has Lipschitz smoothness (loosely speaking) and d' of the d features are relevant. The best rate of convergence for this class is $n^{-1/d'}$. We will show that MV-SRM can achieve this rate (within a log factor) without knowing d' or which features are relevant. This demonstrates the strength of the oracle inequality, from which the result is derived.

To obtain these rates we apply MV-SRM to sets based on a special family of decision trees called dyadic decision trees (DDTs) (Scott and Nowak, 2006). Before introducing DDTs, however, we first introduce the class of distributions $\mathcal{D}$ with which our study is concerned. Throughout this section we assume $\mathcal{X} = [0,1]^d$ and μ is the Lebesgue (equivalently, uniform) measure.

Somewhat related to the approach considered here is the work of Klemelä (2004) who considers the problem of estimating the support of a uniform density. The estimators proposed therein are based on dyadic partitioning schemes similar in spirit to the DDTs studied here. However, it is important to point out that in the support set estimation problem studied by Klemelä (2004) the boundary of the set corresponds to discontinuity of the density, and therefore more standard complexity-regularization and tree pruning methods commonly employed in regression settings suffice to achieve near minimax rates. In contrast, DDT methods are capable of attaining near minimax rates for all density level sets whose boundaries belong to certain Hölder smoothness classes, regardless of whether or not there is a discontinuity at the given level. Significantly different risk bounding and pruning techniques are required for this additional capability (Scott and Nowak, 2006).

6.1 The Box-Counting Class

Before introducing $\mathcal{D}$ we need some additional notation. Let m denote a positive integer, and define $\mathcal{P}_m$ to be the collection of m^d cells formed by the regular partition of $[0,1]^d$ into hypercubes of sidelength $1/m$. Let $c_1, c_2 > 0$ be positive real numbers. Let G^*_α be a minimum volume set, assumed to exist, and let ∂G^*_α be the topological boundary of G^*_α. Finally, let $N_m(\partial G^*_\alpha)$ denote the number of cells in $\mathcal{P}_m$ that intersect ∂G^*_α.

We define the *box-counting* class to be the set $\mathcal{D}_{\text{BOX}} = \mathcal{D}_{\text{BOX}}(c_1, c_2)$ of all distributions satisfying

A1' : X has a density f with respect to μ and f is essentially bounded by c_1.

A3 : $\exists G^*_\alpha$ such that $N_m(\partial G^*_\alpha) \leq c_2 m^{d-1}$ for all m.

Note that since μ is the Lebesgue measure, assumption **A2** from above follows from **A1**, so we do not need to assume it explicitly here. Assumption **A1'** is a slight strengthening of **A1** and implies $P(A) \leq c_1 \mu(A)$ for all measurable sets A. Assumption **A3** essentially requires the boundary of the minimum volume set G^*_α to have Lipschitz smoothness, and thus one would expect the optimal rate

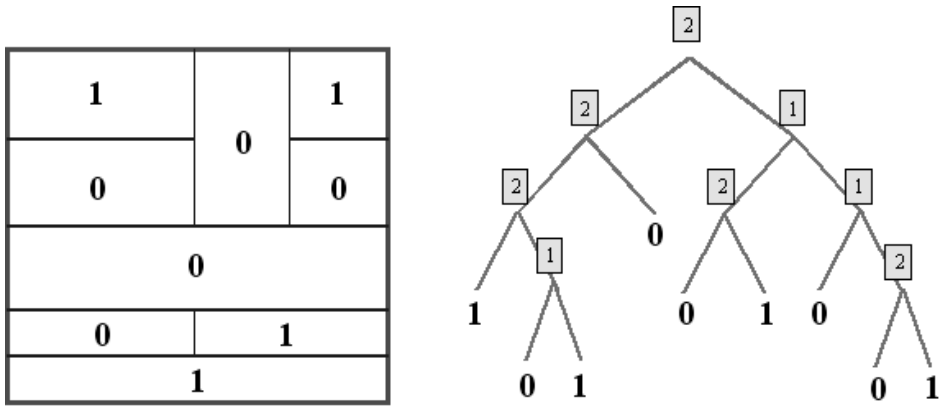


Figure 2: A dyadic decision tree (right) with the associated recursive dyadic partition (left) in $d = 2$ dimensions. Each internal node of the tree is labeled with an integer from 1 to d indicating the coordinate being split at that node. The leaf nodes are decorated with class labels.

of convergence to be $n^{-1/d}$ (the typical rate for set estimation problems characterized by Lipschitz smoothness). See Scott and Nowak (2006) for further discussion of the box-counting assumption.

6.2 Dyadic Decision Trees

Let T denote a tree structured classifier $T : [0,1]^d \to \{0,1\}$. Each such T gives rise to a set $G_T = \{x \in [0,1]^d : T(x) = 1\}$. In this subsection we introduce a certain class of trees, and later consider MV-SRM over the induced class of sets.

Scott and Nowak (2006) demonstrate that *dyadic decision trees* (DDTs) offer a computationally feasible classifier that also achieves optimal rates of convergence (for standard classification) under a wide range of conditions. DDTs are especially well suited for rate of convergence studies. Indeed, bounding the approximation error is handled by the restriction to dyadic splits, which allows us to take advantage of recent insights from multiresolution analysis and nonlinear approximations (DeVore, 1998; Cohen et al., 2001; Donoho, 1999). An analysis similar to that of Scott and Nowak (2006) applies to MV-SRM for DDTs, leading to similar results: optimal rates of convergence for a computationally efficient learning algorithm.

A dyadic decision tree is a decision tree that divides the input space by means of axis-orthogonal dyadic splits. More precisely, a DDT T is a binary tree (with a distinguished root node) specified by assigning (1) an integer $c(v) \in \{1,\ldots,d\}$ to each internal node v of T (corresponding to the coordinate that gets split at that node); (2) a binary label 0 or 1 to each leaf node of T. The nodes of DDTs correspond to hyperrectangles (cells) in $[0,1]^d$. Given a hyperrectangle $A = \prod_{c=1}^{d}[a_c, b_c]$, let $A^{c,1}$ and $A^{c,2}$ denote the hyperrectangles formed by splitting A at its midpoint along coordinate c. Specifically, define $A^{c,1} = \{x \in A \mid x_c \leq (a_c + b_c)/2\}$ and $A^{c,2} = A \backslash A^{c,1}$.

Each node of T is associated with a cell according to the following rules: (1) The root node is associated with $[0,1]^d$; (2) If v is an internal node associated with the cell A, then the children of v are associated with $A^{c(v),1}$ and $A^{c(v),2}$. See Figure 2. Note that every T corresponds to a set $G_T \in [0,1]^d$ (the regions labeled 1), and we think of DDTs as both classifiers and sets interchangeably.

Let $L = L(n)$ be a natural number and define $\mathcal{T}^L$ to be the collection of all DDTs such that (1) no leaf cell has a sidelength smaller than 2^{-L}, and (2) any two leaf nodes that are siblings have different labels. Condition (1) says that when traversing a path from the root to a leaf no coordinate is split more than L times. Condition (2) means that it is impossible to "prune" at any internal node and still have the same set/classifier. Also define $\mathcal{A}^L$ to be the collection of all cells A that correspond to nodes of DDTs in $\mathcal{T}^L$. Define $\pi(T)$ to be the collection of "leaf" cells of T. For a cell $A \in \mathcal{A}^L$, let $j(A)$ denote the depth of A when viewed as a node in some DDT. Observe that when μ is the Lebesgue measure, $\mu(A) = 2^{-j(A)}$.

6.3 MV-SRM with Dyadic Decision Trees

We study MV-SRM over the family $\mathcal{G}^L = \{G_T : T \in \mathcal{T}^L\}$, where L is set by the user. To simplify the notation, at times we will suppress the dependence of ϕ on the training sample S and confidence parameter δ. Thus our MV set estimator has the form

$$\widehat{G}_\alpha = \arg\min_{G \in \mathcal{G}^L} \left\{ \mu(G) + 2\phi(G) \,|\, \widehat{P}(G) + \phi(G) \geq \alpha \right\}. \tag{18}$$

It remains to specify the penalty ϕ. There are a number of ways to produce ϕ satisfying

$$P^n \left(\left\{ S : \sup_{G \in \mathcal{G}^L} \left(\left| P(G) - \widehat{P}(G) \right| - \phi(G,S,\delta) \right) > 0 \right\} \right) \leq \delta.$$

Since $\mathcal{G}^L$ is countable (in fact, finite), one approach is to devise a prefix code for $\mathcal{G}^L$ and apply the penalty in Section 2.2. Instead, we employ a different penalty which has the advantage that it leads to minimax optimal rates of convergence. Introduce the notation $[\![A]\!] = (3 + \log_2 d)\, j(A)$, which may be thought of as the codelength of A in a prefix code for $\mathcal{A}^L$, and define the *minimax* penalty

$$\phi(G_T) := \sum_{A \in \pi(T)} \sqrt{8 \max\left(\widehat{P}(A), \frac{[\![A]\!] \log 2 + \log(2/\delta)}{n} \right) \frac{[\![A]\!] \log 2 + \log(2/\delta)}{n}}. \tag{19}$$

For each $A \in \pi(T)$, set $\ell(A) = 1$ if $A \subset G_T$ and 0 otherwise. The bound originates from writing

$$P(G_T) - \widehat{P}(G_T) = \sum_{A \in \pi(T) : \ell(A) = 1} P(A) - \widehat{P}(A)$$

and

$$\begin{aligned} \widehat{P}(G_T) - P(G_T) &= P(\overline{G_T}) - \widehat{P}(\overline{G_T}) \\ &= \sum_{A \in \pi(T) : \ell(A) = 0} P(A) - \widehat{P}(A) \end{aligned}$$

from which it follows that

$$|P(G_T) - \widehat{P}(G_T)| \leq \sum_{A \in \pi(T)} P(A) - \widehat{P}(A). \tag{20}$$

The event $X \in A$ is a Bernoulli trial with probability of success $P(A)$, and so bounding the right hand side of (20) simply involves applying a concentration inequality for binomials to each $A \in \mathcal{A}^L$.

There are many ways to do this (additive Chernoff, relative Chernoff, exact tail inversion, etc.), but the one we have chosen is particularly convenient for rate of convergence analysis. For further discussion, see Scott and Nowak (2006). Proof of the following result is nearly identical to a similar result in Scott and Nowak (2006), and is omitted.

Proposition 14 *Let ϕ be as in (19) and let $\delta \in (0,1)$. With probability at least $1-\delta$ over the draw of S,*

$$|P(G) - \widehat{P}(G)| \leq \phi(G)$$

for all $G \in \mathcal{G}^L$. Thus ϕ is a complexity penalty for $\mathcal{G}^L$.

The MV-SRM procedure over $\mathcal{G}^L$ with the above penalty leads to an optimal rate of convergence for the box-counting class.

Theorem 15 *Choose $L = L(n)$ and $\delta = \delta(n)$ such that*

1. $2^{L(n)} \succcurlyeq (n/\log n)^{1/d}$
2. $\delta(n) = O(\sqrt{\log n/n})$ *and* $\log(1/\delta(n)) = O(\log n)$

Define $\widehat{G}_\alpha$ as in (18) with ϕ as in (19). For $d \geq 2$ we have

$$\sup_{\mathcal{D}_{\text{BOX}}} \mathbf{E}^n \mathcal{E}(\widehat{G}_\alpha) \preccurlyeq \left(\frac{\log n}{n}\right)^{\frac{1}{d}}. \tag{21}$$

We omit the proof, since this theorem is a special case of Theorem 16 below. Note that the condition on δ is satisfied if $\delta(n) \asymp n^{-\beta}$ for some $\beta > 1/2$.

6.4 Adapting to Relevant Features

The previous result could have been obtained without using MV-SRM. Instead, we could have applied MV-ERM to a fixed hierarchy $\mathcal{G}^{L(1)}, \mathcal{G}^{L(2)}, \ldots$ where $L(n) \asymp (n/\log n)^{1/d}$. The strength of MV-SRM and the associated oracle inequality is in its ability to adapt to favorable conditions on the data generating distribution which may not be known in advance. Here we illustrate this idea when the number of relevant features is not known in advance.

We define the *relevant data dimension* to be the number $d' \leq d$ of relevant features. A feature X^i, $i = 1, \ldots, d$, is said to be relevant provided $f(X)$ is not constant when X^i is varied from 0 to 1. For example, if $d = 2$ and $d' = 1$, then ∂G^*_α is a horizontal or vertical line segment (or union of such line segments). If $d = 3$ and $d' = 1$, then ∂G^*_α is a plane (or union of planes) orthogonal to one of the axes. If $d = 3$ and the third coordinate is irrelevant ($d' = 2$), then ∂G^*_α is a "vertical sheet" over a curve in the (X^1, X^2) plane (see Figure 3).

Let $\mathcal{D}'_{\text{BOX}} = \mathcal{D}'_{\text{BOX}}(c_1, c_2, d')$ be the set of all product measures P^n such that **A1'** and **A3** hold for the underlying distribution P, and X has relevant data dimension $d' \geq 2$. An argument of Scott and Nowak (2006) implies that the expected minimax rate for d' relevant features is $n^{-1/d'}$. By the following result, MV-SRM can achieve this rate to within a log factor.

Theorem 16 *Choose $L = L(n)$ and $\delta = \delta(n)$ such that*

1. $2^{L(n)} \succcurlyeq n/\log n$

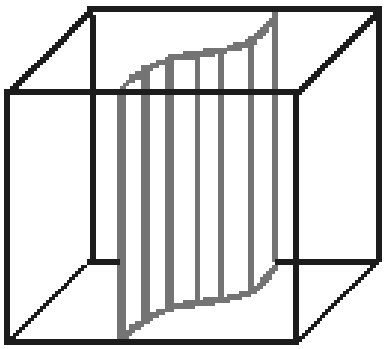


Figure 3: Cartoon illustrating relevant data dimension. If the X^3 axis is irrelevant, then the boundary of the MV-set is a "vertical sheet" over a curve in the (X^1, X^2) plane.

2. $\delta(n) = O(\sqrt{\log n/n})$ *and* $\log(1/\delta(n)) = O(\log n)$

Define $\widehat{G}_\alpha$ *as in (18) with* ϕ *as in (19). If* $d' \geq 2$ *then*

$$\sup_{\mathcal{D}'_{\mathrm{BOX}}} \mathbf{E}^n \mathcal{E}(\widehat{G}_\alpha) \preccurlyeq \left(\frac{\log n}{n}\right)^{\frac{1}{d'}}. \tag{22}$$

The proof hinges on the oracle inequality. The details of the proof are very similar to the proof of a result in Scott and Nowak (2006) and are therefore omitted. Here we just give a sketch of how the oracle inequality comes into play.

Let $K \leq L$ and let $G_K^* \in \mathcal{G}_\alpha^K$ be such that (i) $\mu(G_K^*) = \arg\min_{G \in \mathcal{G}_\alpha^K} \mu(G) - \mu_\alpha^*$; and (ii) G_K^* is based on the smallest possible partition among all sets satisfying (i). Set $m = 2^K$. It can be shown that

$$\mu(G_K^*) - \mu_\alpha^* + \phi(G_K^*, S, \delta) \preccurlyeq m^{-1} + m^{d'/2-1}\sqrt{\frac{\log n}{n}}$$

in expectation. This upper bound is minimized when $m \asymp (n/\log n)^{1/d'}$, in which case we obtain the stated rate. Here the oracle inequality is crucial because m depends on d', which is not known in advance. The oracle inequality tells us that MV-SRM performs as if it knew the optimal K.

Note that the set estimation rule does not require knowledge of the constants c_1 and c_2, nor d', nor which features are relevant. Thus the rule is completely automatic and adaptive.

7. Experiments

In this section we conduct some simple numerical experiments to illustrate the rules for MV-set estimation proposed in this work. Our objective is not an extensive comparison with competing methods, but rather to demonstrate that our estimators behave in a way that agrees with the theory, to gain insight into the behavior of various penalties, and to examine basic algorithmic issues. Throughout this section we take $\mathcal{X} = [0,1]^d$ and μ to be the Lebesgue (equivalently, uniform) measure.

7.1 Histograms

We devised a simple numerical experiment to illustrate MV-SRM in the case of histograms (see Sections 3.2 and 4.2). In this case, MV-SRM can be implemented exactly with a simple procedure. First, compute the MV-ERM estimate for each $\mathcal{G}^k$, $k = 1, \ldots, K$, where $1/k$ is the bin-width. To do this, for each k, sort the cells of the partition according to the number of samples in the cell. Then, begin incorporating cells into the estimate one cell at a time, starting with the most populated, until the empirical mass constraint is satisfied. Finally, once all MV-ERM estimates have been computed, choose the one that minimizes the penalized volume.

We consider two penalties. Both penalties are defined via $\phi(G,S,\delta) = \phi_k(G,S,\delta 2^{-k})$ for $G \in \mathcal{G}^k$, where ϕ_k is a penalty for $\mathcal{G}^k$. The first is based on the simple Occam-style bound of Section 3.2. For $G \in \mathcal{G}^k$, set

$$\phi_k^{Occ}(G,S,\delta) = \sqrt{\frac{k^d \log 2 + \log(2/\delta)}{2n}}.$$

The second is the (conditional) Rademacher penalty. For $G \in \mathcal{G}^k$, set

$$\phi_k^{Rad}(G,S,\delta) = \frac{2}{n}\mathbf{E}_{(\sigma_i)}\left[\sup_{G' \in \mathcal{G}^k} \sum_{i=1}^{n} \sigma_i \mathbb{I}\left(X_i \in G'\right)\right] + \sqrt{\frac{2\log(2/\delta)}{n}}.$$

Here $\sigma_1, \ldots, \sigma_n$ are Rademacher random variables, i.e., independent random variables taking on the values 1 and -1 with equal probability. Fortunately, the conditional expectation with respect to these variables can be evaluated exactly in the case of partition-based rules such as the histogram. See Appendix E for details.

As a data set we consider $\mathcal{X} = [0,1]^d$, the unit square, and data generated by a two-dimensional truncated Gaussian distribution, centered at the point $(1/2, 1/2)$ and having spherical variance with parameter $\sigma = 0.15$. Other parameter settings are $\alpha = 0.8$, $K = 40$, and $\delta = 0.05$. All experiments were conducted at nine different sample sizes, logarithmically spaced from 100 to 1000000, and repeated 100 times. Figure 4 shows a representative training sample and MV-ERM estimates with $\nu = 1, 0$, and -1. These examples clearly demonstrate that the larger ν, the smaller the estimate.

Figure 5 depicts the error $\mathcal{E}(\widehat{G})$ of the MV-SRM estimate with $\nu = 1$. The Occam's Razor penalty consistently outperforms the Rademacher penalty. For comparison, a damped version ($\nu = 0$) was also evaluated. It is clear from the graphs that $\nu = 0$ outperforms $\nu = 1$. This happens because the damped version distributes the error more evenly between mass and volume, as discussed in Section 5.

Figure 6 depicts the penalized volume of the MV-ERM estimates ($\nu = 1$) as a function of the resolution k, where $1/k$ is the sidelength of the histogram cell. MV-SRM selects the resolution where this curve is minimized. Clearly the Occam's Razor bound is tighter than the Rademacher bound (look at the right side of the graph), which explains why Occam outperforms Rademacher. Figure 7 depicts the average resolution of the estimate (top) and the average symmetric difference with respect to the true MV-set, for various sample sizes. These graphs are for $\nu = 1$. The graphs for $\nu = 0$ do not change considerably. Thus, while damping seems to have a noticeable effect on the error quantity $\mathcal{E}$, the effect on the symmetric difference is much less pronounced.

7.2 Dyadic Decision Trees

Implementing MV-SRM for dyadic decision trees is much more challenging than for histograms. Although an exact algorithm is possible (see Scott and Nowak, 2005a), we suggest an approximate

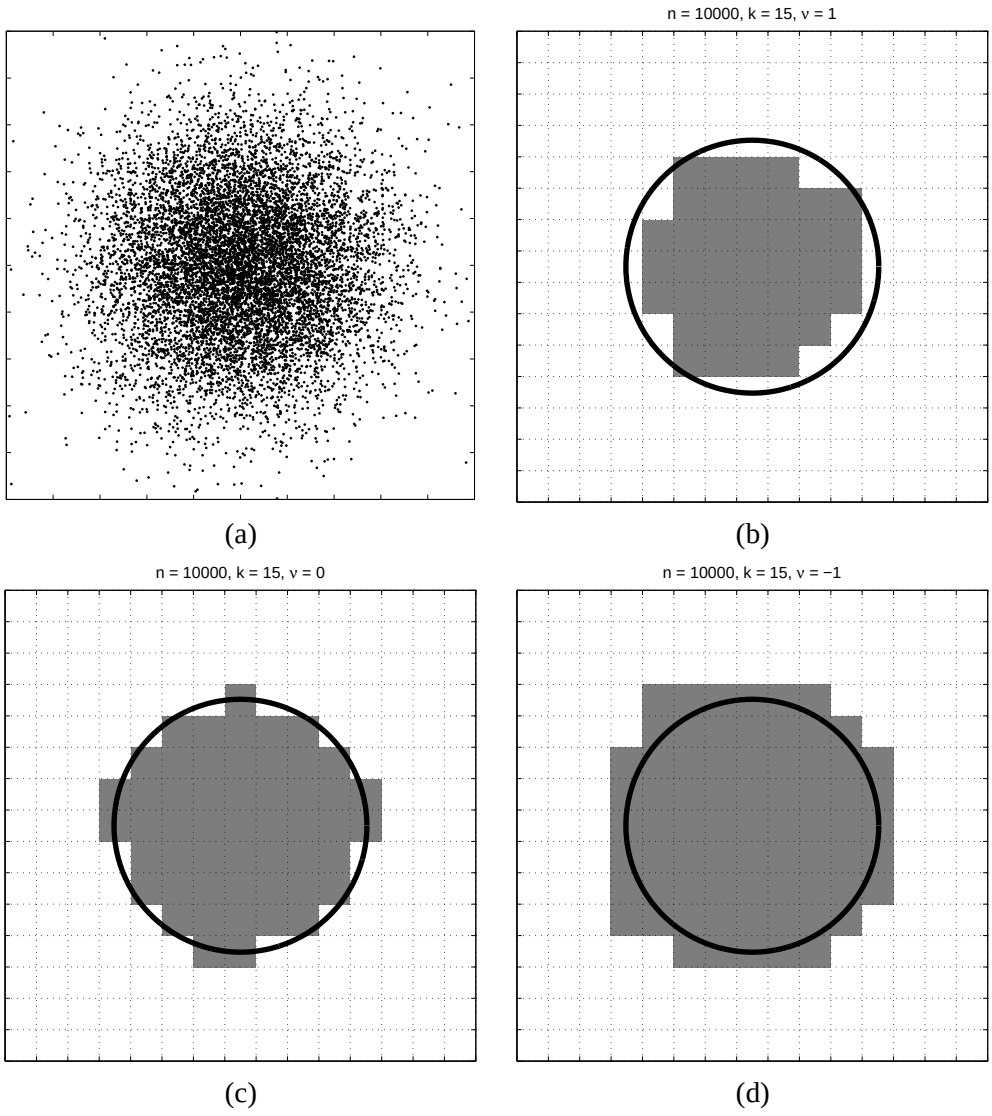


Figure 4: Data and three representative MV-ERM histogram estimates for the data in Section 7.1. The shaded region is the MV-set estimate, and the solid circle indicates the true MV-set. All estimates are based on the Occam bound. (a) 10000 realizations used for training. (b) MV-ERM estimate with a bin-width of 1/15 and $\nu = 1$. (c) $\nu = 0$. (d) $\nu = -1$. Clearly, the larger ν, the smaller the estimate.

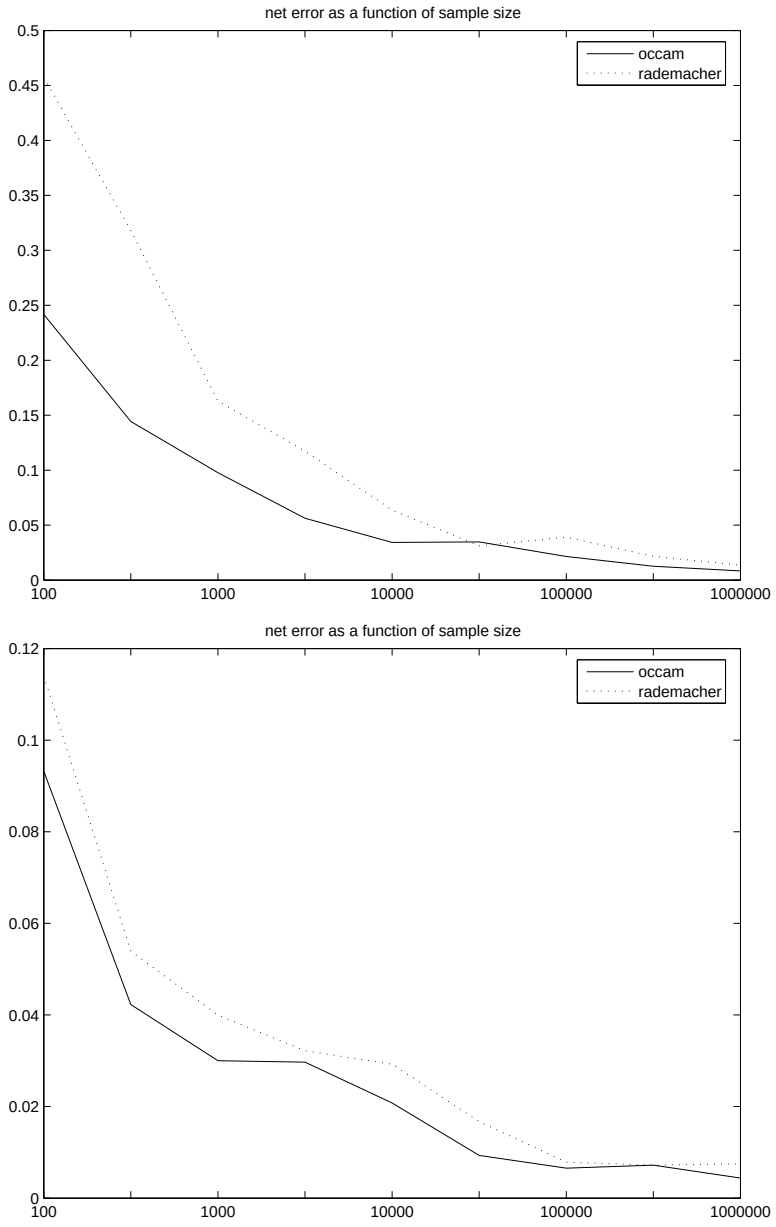


Figure 5: The error $\mathcal{E}\left(\widehat{G}_{\mathcal{G},\alpha}\right)$ as a function of sample size for the histogram experiments in Section 7.1. All results are averaged over 100 repetitions for each training sample size. (Top) Results for the original MV-SRM algorithm ($\nu = 1$). (Bottom) Results for $\nu = 0$. In this case the error is more evenly distributed between mass and volume, whereas in the former case all the error is in the mass term.

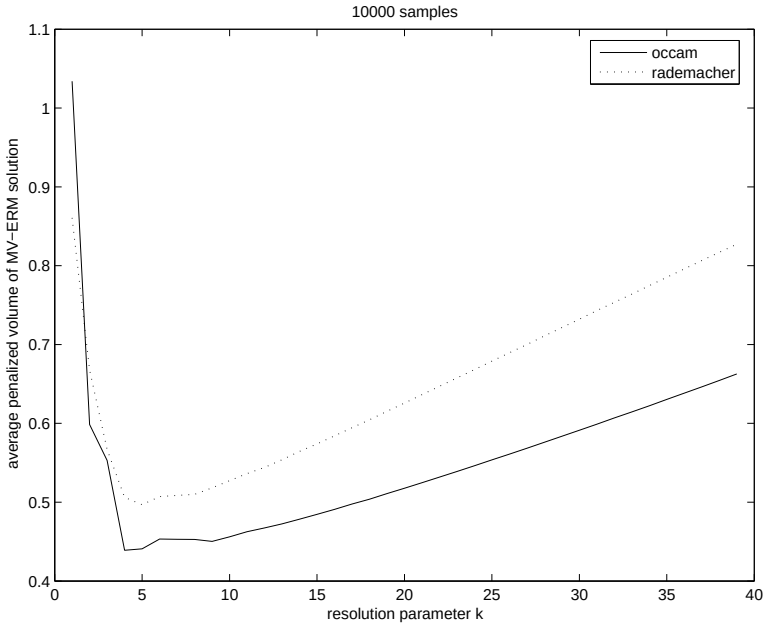


Figure 6: The penalized volume of the MV-ERM estimates $G^k_{\mathcal{G},\alpha}$, as a function of k, where $1/k$ is the sidelength of the histogram cell. The results are for a sample size of 10000. Results represent an average over 100 repetitions. Clearly, the Occam's razor bound is smaller than the Rademacher penalty (look at the right side of the plot), to which we may attribute its improved performance (see Figure 5).

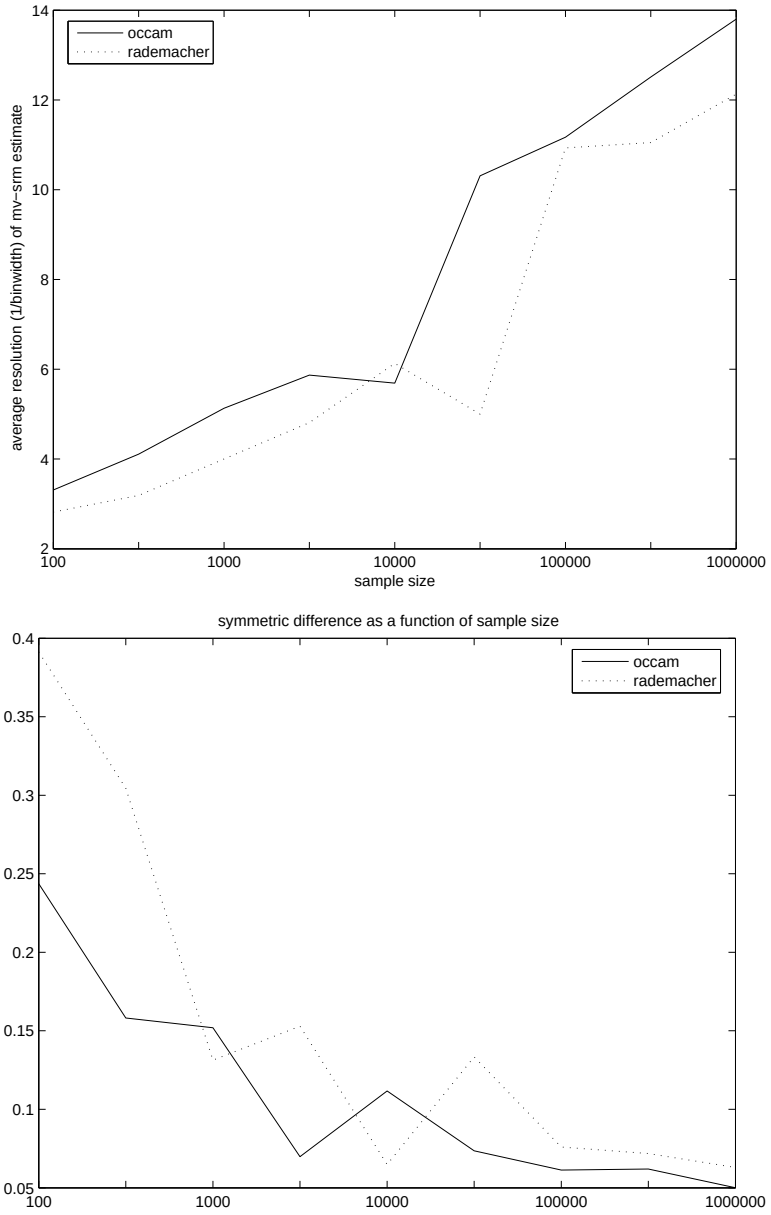


Figure 7: Results from the histogram experiments in Section 7.1. All results are averaged over 100 repetitions for each training sample size, and are for the non-damped version of MV-SRM ($\nu = 1$). (Top) Average value of the resolution parameter k ($1/k =$ sidelength of histogram cells) as a function of sample size. (Bottom) Average value of the symmetric difference between the estimated and true MV-sets. Neither graph changes significantly if ν is varied.

algorithm based on a reformulation of the constrained optimization problem defining MV-SRM in terms of its Lagrangian, coupled with a bisection search to find the appropriate Lagrange multiplier. If the penalty is additive, then the unconstrained Lagrangian can be minimized efficiently using existing algorithmic approaches.

A penalty for a DDT is said to be *additive* if it can be written in the form

$$\phi(G_T) = \sum_{A \in \pi(T)} \psi(A)$$

for some ψ. If ϕ is additive the optimization in (18) can be re-written as

$$\min_{T \in \mathcal{T}^L} \sum_{A \in \pi(T)} [\mu(A)\ell(A) + (1+\nu)\psi(A)] \text{ subject to } \sum_{A \in \pi(T)} \left[\widehat{P}(A)\ell(A) + \nu\psi(A)\right] \geq \alpha$$

where $\ell(A)$ is the binary label of leaf A ($\ell(A) = 1$ if A is in the candidate set and 0 otherwise). Introducing the Lagrange multiplier $\lambda > 0$, the unconstrained Lagrangian formulation of the problem is

$$\min_T \sum_{A \in T} \left[\mu(A)\ell(A) + (1+\nu)\psi(A) - \lambda\left(\widehat{P}(A)\ell(A) + \nu\psi(A)\right)\right].$$

Inspection of the Lagrangian reveals that the optimal choice of $\ell(A)$ is

$$\ell(A) = \begin{cases} 1 & \text{if } \lambda\widehat{P}(A) \geq \mu(A), \\ 0 & \text{otherwise} \end{cases}$$

Thus, we have a "per-leaf" cost function

$$\text{cost}(A) := \min(\mu(A) - \lambda\widehat{P}(A), 0) + (1 + \nu(1-\lambda))\psi(A)$$

For a given value of λ, the optimal tree can be efficiently obtained using the algorithm of Blanchard et al. (2004).

We also note that the above strategy works for tree structures besides the one studied in Section 6. For example, suppose an overfitted tree (with arbitrary, non-dyadic splits) has been constructed by some greedy heuristic (perhaps using an independent data set). Or, suppose that instead of binary dyadic splits with arbitrary orientation, one only considers "quadsplits" whereby every parent node has 2^d children (in fact, this is the tree structure used for our experiments below). In such cases, optimizing the Lagrangian reduces to a classical pruning problem, and the optimal tree can be found by a simple $O(n)$ dynamic program that has been used since at least the days of CART (Breiman et al., 1984).

Let $\widehat{T}_\lambda$ denote the tree resulting from the Lagrangian optimization above. From standard optimization theory, we know that for each value of λ, $\widehat{T}_\lambda$ will coincide with $\widehat{G}_\alpha$, for a certain value of α. For each value of λ there is a corresponding α, but the converse is not necessarily true. Therefore, the Lagrangian solutions correspond to many, but not all possible solutions of the original MV-SRM optimization with different values of α. Despite this potential limitation, the simplicity of the Lagrangian optimization makes this a very attractive approach to MV-SRM in this case. We can determine the best value of λ for a given target α by repeatedly solving the Lagrangian optimization and finding the setting for λ that meets or comes closest to the original constraint. The search over λ can be conducted efficiently using a bisection search.

In our experiments we do not consider the "free-split" tree structure described in Section 6, in which each parent has two children defined by one of $d = 2$ possible splits. Instead, we assume a quadsplit tree structure, whereby every cell is a square, and every parent has four square children. The total optimization time is $O(mn)$, where m is the number of steps in the bisection search. In our experiments presented below we found that ten steps (i.e., ten Lagrangian tree pruning optimizations) were sufficient to meet the constraint almost exactly (whenever possible).

We consider three complexity penalties. We refer to the first penalty as the *minimax* penalty, since it is inspired by the minimax optimal penalty in (19):

$$\psi^{mm}(A) := (0.01)\sqrt{8\max\left(\widehat{P}(A), \frac{[\![A]\!]\log 2 + \log(2/\delta)}{n}\right)\frac{[\![A]\!]\log 2 + \log(2/\delta)}{n}}. \tag{23}$$

Note that the penalty is down-weighted by a constant factor of 0.01, since otherwise it is too large to yield meaningful results:[3]

The second penalty is based on the Rademacher penalty (see Section 2.3). Let Π^L denote the set of all partitions π of trees in $\mathcal{T}^L$. Given $\pi_0 \in \Pi^L$, set $\mathcal{G}_{\pi_0} = \{G_T \in \mathcal{G}^L : \pi(T) = \pi_0\}$. Recall $\pi(T)$ denotes the partition associated with the tree T. Combining Proposition 7 with the results of Appendix E, we know that for any fixed π,

$$\sum_{A\in\pi}\sqrt{\frac{\widehat{P}(A)}{n}} + \sqrt{\frac{2\log(2/\delta)}{n}}$$

is a complexity penalty for $\mathcal{G}_\pi$. To obtain a penalty for all $\mathcal{G}^L = \cup_{\pi\in\Pi^L}\mathcal{G}_\pi$, we apply the union bound over all $\pi \in \Pi^L$ and replace δ by $\delta|\Pi^L|^{-1}$. Although distributing the "delta" uniformly across all partitions is perhaps not intuitive (one might expect smaller partitions to be more likely and hence they should receive a larger chunk of the delta), it has the important property that the delta term is the same for all trees, and thus can be dropped for the purposes of minimization. Hence, the effective penalty is additive. In summary, our second penalty, referred to as the Rademacher penalty,[4] is given by

$$\psi^{Rad}(A) \;=\; \sqrt{\frac{\widehat{P}(A)}{n}}. \tag{24}$$

The third penalty is referred to as the modified Rademacher penalty and is given by

$$\psi^{mRad}(A) \;=\; \sqrt{\frac{\widehat{P}(A) + \mu(A)}{n}}. \tag{25}$$

The modified Rademacher penalty is still a valid penalty, since it strictly dominates the basic Rademacher penalty. The basic Rademacher is proportional to the square-root of the empirical P mass and the modified Rademacher is proportional to the square-root of the *total* mass (empirical

3. Note that here down-weighting is distinct from damping by ν as discussed earlier. With down-weighting, both occurrences of the penalty, in the constraint and in the objective function, are scaled by the same factor. The oracle inequality (and hence minimax optimality) still holds for the downweighted penalty, albeit with larger constants.
4. Technically, this is an upper bound on the Rademacher penalty, but as discussed in Appendix E, this bound is tight to within a factor of $\sqrt{2}$. Using the exact Rademacher yields essentially the same results. Thus, we refer to this upper bound simply as the Rademacher penalty.

P mass plus μ mass). In our experiments we have found that the modified Rademacher penalty typically performs better than the basic Rademacher penalty, since it discourages the inclusion of very small isolated leafs containing a single data point (as seen in the experimental results below). Note that, unlike the minimax penalty, the two Rademacher-based penalties are not down-weighted; the true penalties are used.

We illustrate the performance of the dyadic quadtree approach with a two-dimensional Gaussian mixture distribution, taking $\nu = 0$. Figure 1 depicts 500 samples from the Gaussian mixture distribution, along with the true minimum volume set for $\alpha = 0.90$. Figures 8, 9, and 10 depict the minimum volume set estimates based on each of the three penalties, and for sample sizes of 100, 1000, and 10000. Here we use MM, Rad, and mRad to designate the three penalties.

In addition to the minimum volume set estimates based on a single tree, we also show the estimates based on voting over shifted partitions. This amounts to constructing $2^L \times 2^L$ different trees, each based on a partition offset by an integer multiple of the base sidelength 2^{-L}, and taking a majority vote over all the resulting set estimates to form the final estimate. These estimates are indicated by MM', Rad', and mRad', respectively. Similar methods based on averaging or voting over shifted partitions have been tremendously successful in image processing, and they tend to mitigate the "blockiness" associated with estimates based on a single tree, as is clearly seen in the results depicted. Moreover, because of the significant amount of redundancy in the shifted partitions, the MM', Rad', and mRad' estimates can be computed in just $O(mn \log n)$ operations.

Visual inspection of the resulting minimum volume set estimates (which were "typical" results selected at random) reveals some of the characteristics of the different penalties and their behaviors as a function of the sample size. Notably, the basic Rademacher penalty tends to allow very small and isolated leafs into the final set estimate, which is somewhat unappealing. The modified Rademacher penalty clearly eliminates this problem and provides very reasonable estimates. The (down-weighted) minimax penalty results in set estimates quite similar to those resulting from the modified Rademacher. However, the somewhat arbitrary choice of scaling factor (0.01 in this case) is undesirable. Finally, let us remark on the significant improvement provided by voting over multiple shifted trees. The voting procedure quite dramatically reduces the "blocky" partition associated with estimates based on single trees. Overall, the modified Rademacher penalty coupled with voting over multiple shifted trees appears to perform best in our experiments. In fact, in the case $n = 10000$, this set estimate is almost identical to the true minimum volume set depicted in Figure 1.

8. Conclusions

In this paper we propose two rules, MV-ERM and MV-SRM, for estimation of minimum volume sets. Our theoretical analysis is made possible by relating the performance of these rules to the uniform convergence properties of the class of sets from which the estimate is taken. This in turn lets us apply distribution free uniform convergence results such as the VC inequality to obtain distribution free, finite sample performance guarantees. It also leads to strong universal consistency when the class of candidate sets is allowed to grow in a controlled way. MV-SRM obeys an oracle inequality and thereby automatically selects the appropriate complexity of the set estimator. These theoretical results are illustrated with histograms and dyadic decision trees.

Our estimators, results, and proof techniques for minimum volume sets bear a strong resemblance to existing estimators, results, and proof techniques for supervised classification. This is no coincidence. Minimum volume set estimation is closely linked with hypothesis testing. Assume

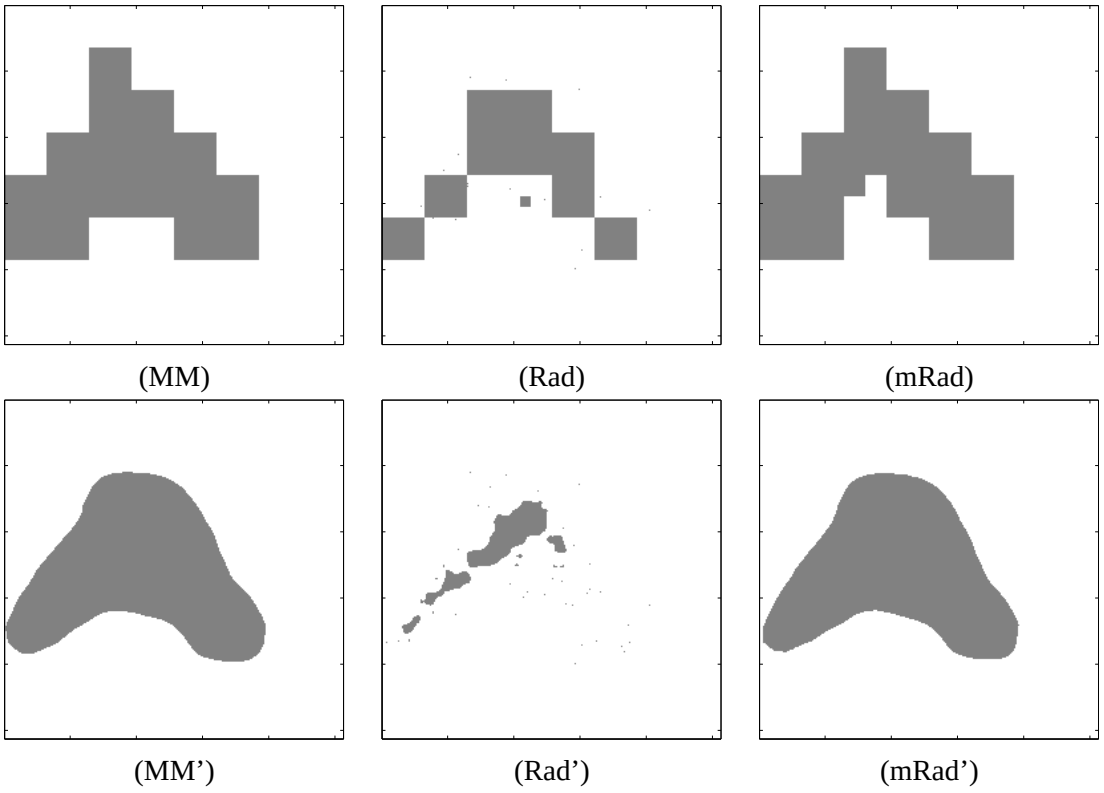


Figure 8: Minimum volume set estimates based on dyadic quadtrees for $\alpha = 0.90$ with $n = 100$ samples. Reconstructions based on MM = minimax penalty (23), Rad = Rademacher penalty (24), and mRad = modified Rademacher penalty (25), and MM', Rad', and mRad' denote the analogous estimates based on voting over multiple trees at different shifts.

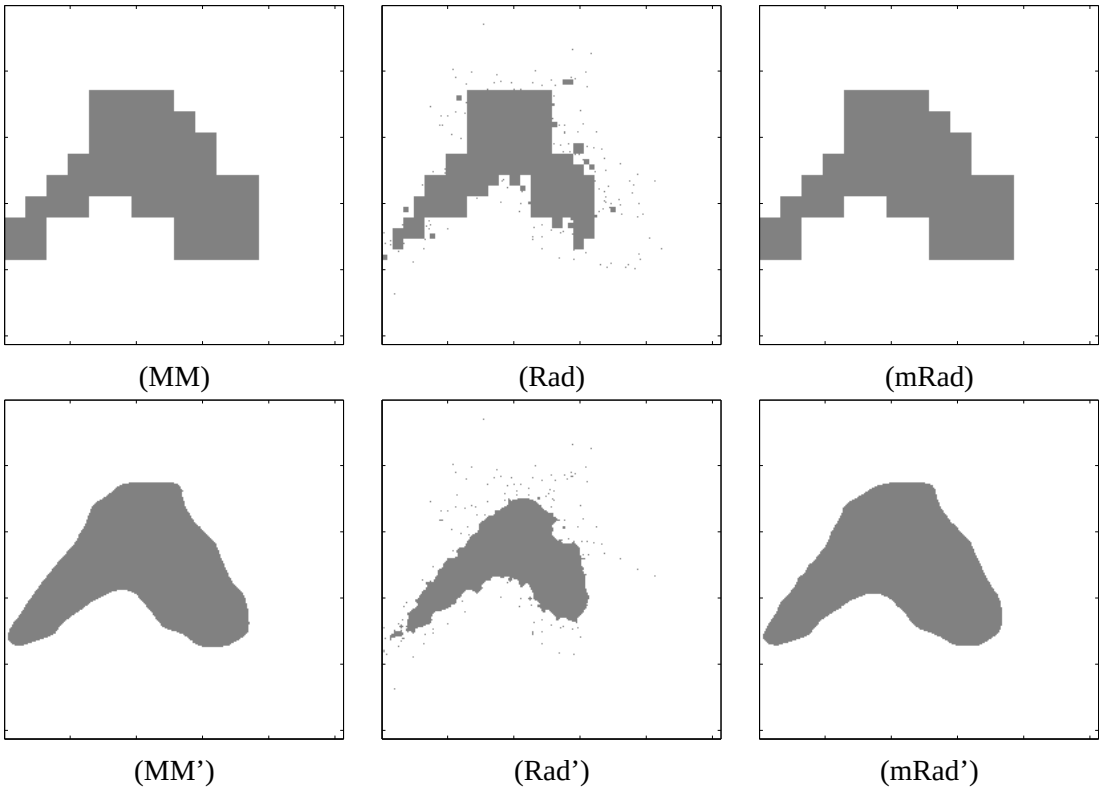


Figure 9: Minimum volume set estimates based on dyadic quadtrees for $\alpha = 0.90$ with $n = 1000$ samples. Reconstructions based on MM = minimax penalty (23), Rad = Rademacher penalty (24), and mRad = modified Rademacher penalty (25), and MM', Rad', and mRad' denote the analogous estimates based on voting over multiple trees at different shifts.

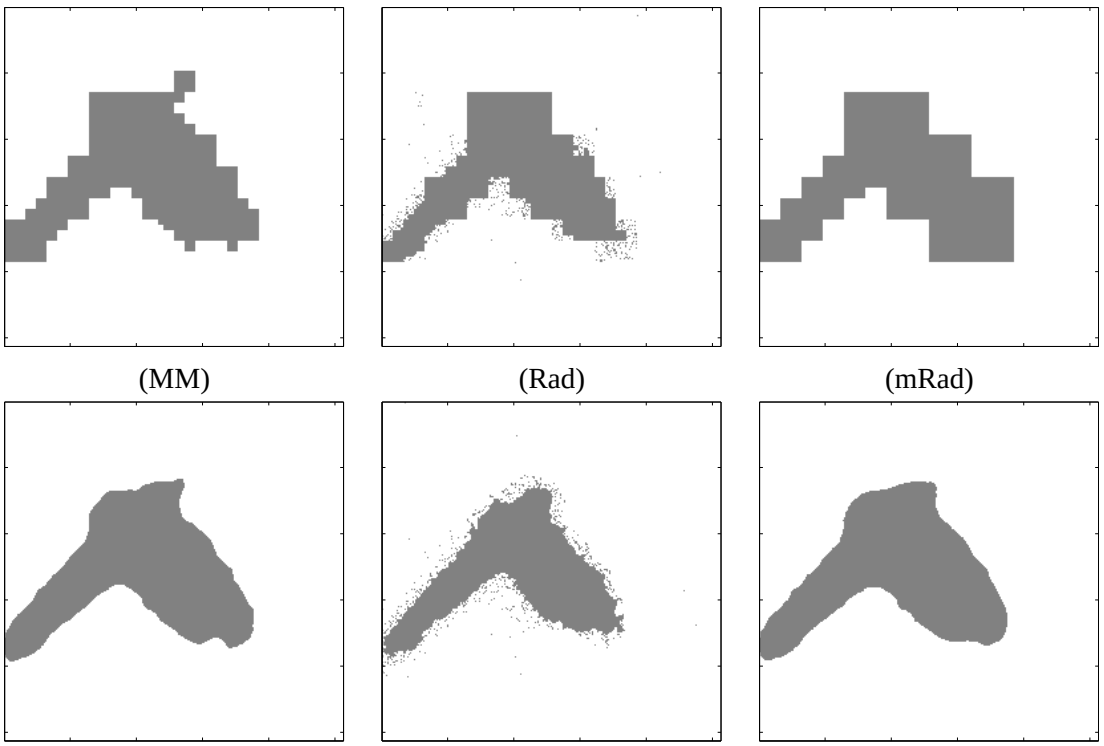


Figure 10: Minimum volume set estimates based on dyadic quadtrees for $\alpha = 0.90$ with $n = 10000$ samples. Reconstructions based on MM = minimax penalty (23), Rad = Rademacher penalty (24), and mRad = modified Rademacher penalty (25), and MM', Rad', and mRad' denote the analogous estimates based on voting over multiple trees at different shifts.

P has a density with respect to μ, and that μ is a probability measure. Then the minimum volume set with mass α is the acceptance region of the most powerful test of size $1-\alpha$ for testing $H_0 : X \sim P$ versus $H_1 : X \sim \mu$. But classification and hypothesis testing have the same goals; the difference lies in what knowledge is used to design a classifier/test (training data versus knowledge of the true densities). The problem of learning minimum volume sets stands halfway between these two: For one class the true distribution is known (the reference measure), but for the other only training samples are available.

This observation provides not only intuition for the similarity between MV-set estimation and classification, but it also suggests an alternative approach to MV-set estimation. In particular, suppose it is possible to sample at will from the reference measure. Consider these samples, together with the original training data, to be a labeled training set. Then the MV-set may be estimated by learning a classifier with respect to the Neyman-Pearson criterion (Cannon et al., 2002; Scott and Nowak, 2005b). Briefly, the Neyman-Pearson classification paradigm involves learning a classifier from training data that minimizes the "miss" generalization error while constraining the "false alarm" generalization error to be less than or equal to a specified size, in our case $1-\alpha$.

Minimum volume set estimation based on Neyman-Pearson classification offers a distinct advantage over the rules studied in this paper. Indeed, our algorithms for histograms and dyadic decision trees take advantage of the fact that the reference measure μ is easily evaluated for these special types of sets. For more general sets or non-uniform reference measures, direct evaluation of the reference measure may be impractical. Neyman-Pearson classification, in contrast, involves computing the empirical volume based on the training sample, a much easier task. Moreover, in principle one may take an arbitrarily large sample from μ to mitigate finite sample effects. A similar idea has been employed by Steinwart et al. (2005), who sample from μ so as to reduce density level set estimation to cost-sensitive classification. In this setting the advantage of MV-sets over density level sets is further magnified. For example, to sample from a uniform distribution, one must specify its support, which is a priori unknown. Fortunately, MV-sets are invariant to the choice of support, whereas the γ-level set changes with the support of μ.

Acknowledgments

The authors thank Ercan Yildiz and Rebecca Willett for their assistance with the experiments involving dyadic trees, Gilles Blanchard for his insights into the Rademacher penalty for partition-based estimators, and an anonymous referee for suggesting a simplification in the proof of Theorem 10.

The first author was supported by an NSF VIGRE postdoctoral training grant. The second author was supported by NSF Grants CCR-0310889 and CCF-0353079.

Appendix A. Proof of Lemma 4

The proof follows closely the proof of Lemma 1 in Cannon et al. (2002). Define $\Xi = \{S : \widehat{P}(G_{\mathcal{G},\alpha}) < \alpha - \phi(G_{\mathcal{G},\alpha}, S, \delta)\}$. It is true that $\Theta_\mu \subset \Xi$. To see this, if $S \notin \Xi$ then $G_{\mathcal{G},\alpha} \in \widehat{\mathcal{G}}_\alpha$, and hence $\mu(\widehat{G}_{\mathcal{G},\alpha}) \leq \mu(G_{\mathcal{G},\alpha})$ by definition of $\widehat{G}_{\mathcal{G},\alpha}$. Thus $S \notin \Theta_\mu$. It follows that

$$\Theta_P \cup \Theta_\mu \subset \Theta_P \cup \Xi$$

and hence it suffices to show $\Theta_P \subset \Omega_P$ and $\Xi \subset \Omega_P$.

First, we show that $\Theta_P \subset \Omega_P$. If $S \in \Theta_P$ then

$$P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha - 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta).$$

This implies

$$\begin{aligned} P(\widehat{G}_{\mathcal{G},\alpha}) - \widehat{P}(\widehat{G}_{\mathcal{G},\alpha}) &< \alpha - 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta) - \widehat{P}(\widehat{G}_{\mathcal{G},\alpha}) \\ &\leq -\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta), \end{aligned}$$

where the last inequality is true because $\widehat{P}(\widehat{G}_{\mathcal{G},\alpha}) \geq \alpha - \phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta)$. Therefore $S \in \Omega_P$.

Second, we show that $\Xi \subset \Omega_P$. If $S \in \Xi$, then

$$\begin{aligned} \widehat{P}(G_{\mathcal{G},\alpha}) - P(G_{\mathcal{G},\alpha}) &< \alpha - \phi(G_{\mathcal{G},\alpha}, S, \delta) - P(G_{\mathcal{G},\alpha}) \\ &\leq -\phi(G_{\mathcal{G},\alpha}, S, \delta), \end{aligned}$$

where the last inequality holds because $P(G_{\mathcal{G},\alpha}) \geq \alpha$. Thus, $S \in \Omega_P$, and the proof is complete.

Appendix B. Proof of Theorem 9

By the Borel-Cantelli Lemma (Durrett, 1991), it suffices to show that for any $\varepsilon > 0$,

$$\sum_{n=1}^{\infty} P^n(\mathcal{E}(\widehat{G}_{\mathcal{G},\alpha}) > \varepsilon) < \infty.$$

We will show this by establishing

$$\sum_{n=1}^{\infty} P^n\left(\left(\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha\right)_+ > \frac{\varepsilon}{2}\right) < \infty \tag{26}$$

and

$$\sum_{n=1}^{\infty} P^n\left(\left(\alpha - P(\widehat{G}_{\mathcal{G},\alpha})\right)_+ > \frac{\varepsilon}{2}.\right) < \infty \tag{27}$$

First consider (26). By assumption (11), there exists K such that $\mu(G^k_{\mathcal{G},\alpha}) - \mu^*_\alpha \leq \varepsilon/2$ for all $k \geq K$. Let N be such that $k(n) \geq K$ for $n \geq N$. For any fixed $n \geq N$, consider a sample S of size n. By Theorem 3, it follows that with probability at least $1 - \delta(n)$, $\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha \leq \mu(G^k_{\mathcal{G},\alpha}) - \mu^*_\alpha \leq \varepsilon/2$. Therefore

$$\begin{aligned} &\sum_{n=1}^{\infty} P^n\left(\left(\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha\right)_+ > \frac{\varepsilon}{2}\right) \\ &= \sum_{n=1}^{N-1} P^n\left(\left(\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha\right)_+ > \frac{\varepsilon}{2}\right) + \sum_{n=N}^{\infty} P^n\left(\left(\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha\right)_+ > \frac{\varepsilon}{2}\right) \\ &\leq \sum_{n=1}^{N-1} P^n\left(\left(\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha\right)_+ > \frac{\varepsilon}{2}\right) + \sum_{n=N}^{\infty} \delta(n) \\ &< \infty. \end{aligned}$$

The second inequality follows from the assumed summability of $\delta(n)$.

To establish (27), let N be large enough so that

$$\sup_{G\in\mathcal{G}^{k(n)}} \phi_k(G,S,\delta(n)) \leq \frac{\varepsilon}{4}$$

for all $n \geq N$. For any fixed $n \geq N$, consider a sample S of size n. By Theorem 3, it follows that with probability at least $1-\delta(n)$, $\alpha - P(\widehat{G}_{\mathcal{G},\alpha}) \leq 2\phi_k(\widehat{G}_{\mathcal{G},\alpha},S,\delta(n)) \leq \varepsilon/2$. Therefore

$$\begin{aligned}
&\sum_{n=1}^{\infty} P^n\left(\left(\alpha - P(\widehat{G}_{\mathcal{G},\alpha})\right)_+ > \frac{\varepsilon}{2}\right) \\
&= \sum_{n=1}^{N-1} P^n\left(\left(\alpha - P(\widehat{G}_{\mathcal{G},\alpha})\right)_+ > \frac{\varepsilon}{2}\right) + \sum_{n=N}^{\infty} P^n\left(\left(\alpha - P(\widehat{G}_{\mathcal{G},\alpha})\right)_+ > \frac{\varepsilon}{2}\right) \\
&\leq \sum_{n=1}^{N-1} P^n\left(\left(\alpha - P(\widehat{G}_{\mathcal{G},\alpha})\right)_+ > \frac{\varepsilon}{2}\right) + \sum_{n=N}^{\infty} \delta(n) \\
&< \infty.
\end{aligned}$$

This completes the proof.

Appendix C. Proof of Theorem 10

The first part of the theorem is straightforward. First, we claim that $(\mu(G_n) - \mu^*_\alpha)_+ \leq \mu(G_n \backslash G^*_\alpha)$. To see this, assume $\mu(G_n) - \mu^*_\alpha \geq 0$, otherwise the statement is trivial. Then

$$\begin{aligned}
(\mu(G_n) - \mu^*_\alpha)_+ &= \mu(G_n) - \mu^*_\alpha \\
&= \mu(G_n) - \mu(G^*_\alpha) \\
&\leq \mu(G_n) - \mu(G^*_\alpha \cap G_n) \\
&= \mu(G_n \backslash G^*_\alpha).
\end{aligned}$$

Similarly, one can show $(\alpha - P(G_n))_+ \leq P(G^*_\alpha \backslash G_n)$. Let $D_\gamma = \{x : f(x) \geq \gamma\}$ and $E_n = G^*_\alpha \backslash G_n$. Then for any $\gamma > 0$,

$$P(E_n) = P(E_n \cap D_\gamma) + P(E_n \cap \overline{D_\gamma}) \leq P(D_\gamma) + \gamma\mu(E_n).$$

By the dominated convergence theorem, $P(D_\gamma) \to 0$ as $\gamma \to \infty$. Thus, for any $\varepsilon > 0$, we can choose γ such that $P(D_\gamma) \leq \varepsilon$ and then n large enough so that $\gamma\mu(E_n) \leq \varepsilon$. The result follows.

Now the second part of the theorem. From Section 1.2, we know $G^*_\alpha = \{x : f(x) = \gamma_\alpha\}$ where γ_α is the unique number such that $\int_{f(x)\geq\gamma_\alpha} f(x)d\mu(x) = \alpha$.

Consider the distribution Q of $(X,Y) \in \mathcal{X} \times \{0,1\}$ given by the class-conditional distributions $X|Y=0 \sim P$ and $X|Y=1 \sim \mu$, and a priori class probabilities $Q(Y=0) = p = 1-Q(Y=1)$, where p will be specified below. Then Q defines a classification problem. Let h^* denote a Bayes classifier with respect to Q (i.e., a classifier with minimum probability of error), and let $h : \mathcal{X} \to \{0,1\}$ be an arbitrary classifier. The classification risk of h is defined as $\mathcal{R}(h) = Q(h(X) \neq Y)$, and the excess classification risk is $\mathcal{R}(h) - \mathcal{R}(h^*)$. From Bayes decision theory we know that h^* is the rule that compares the likelihood ratio to $p/(1-p)$. But, as discussed in Section 1.2, the likelihood ratio is $1/f$. Therefore, if p is such that $p/(1-p) = 1/\gamma_\alpha$, then $h^*(x) = 1 - \mathbb{I}(x \in G^*_\alpha)$ μ almost everywhere.

Setting $h_n(x) = 1 - \mathbb{I}(x \in G_n)$, we have

$$\begin{aligned}
&\mathcal{R}(h_n) - \mathcal{R}(h^*) \\
&\quad = Q(h_n(X) \neq Y) - Q(h^*(X) \neq Y) \\
&\quad = (1-p)(\mu(h_n(X)=0) - \mu(h^*(X)=0))) + p(P(h_n(X)=1) - P(h^*(X)=0)) \\
&\quad = (1-p)(\mu(G_n) - \mu(G_\alpha^*)) + p(1 - P(G_n) - (1 - P(G_\alpha^*))) \\
&\quad = (1-p)(\mu(G_n) - \mu_\alpha^*) + p(\alpha - P(G_n)) \\
&\quad \leq (\mu(G_n) - \mu_\alpha^*) + (\alpha - P(G_n)) \\
&\quad \leq \mathcal{E}(G_n).
\end{aligned}$$

Therefore $\mathcal{R}(h_n) \to \mathcal{R}(h^*)$. We now invoke a result of Steinwart et al. (2005) that says, in our notation, that $\mathcal{R}(h_n) \to \mathcal{R}(h^*)$ if and only if $\mu(G_n \Delta G_\alpha^*) \to 0$, and the proof is complete.

Appendix D. Proof of Theorem 11

Let Ω_P be as in the proof of Theorem 3, and assume $S \in \overline{\Omega_P}$. This holds with probability at least $1-\delta$. We consider three separate cases: (1) $\mu(\widehat{G}_{\mathcal{G},\alpha}) \geq \mu_\alpha^*$ and $P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha$, (2) $\mu(\widehat{G}_{\mathcal{G},\alpha}) \geq \mu_\alpha^*$ and $P(\widehat{G}_{\mathcal{G},\alpha}) \geq \alpha$, and (3) $\mu(\widehat{G}_{\mathcal{G},\alpha}) < \mu_\alpha^*$ and $P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha$. Note that the case in which both $\alpha \leq P(\widehat{G}_{\mathcal{G},\alpha})$ and $\mu(\widehat{G}_{\mathcal{G},\alpha}) < \mu_\alpha^*$ is impossible by definition of minimum volume sets. We will use the following fact:

Lemma 17 *If $S \in \overline{\Omega_P}$, then $\alpha - P(\widehat{G}_{\mathcal{G},\alpha}) \leq 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta)$.*

The proof is a repetition of the proof that $\Theta_P \subset \Omega_P$ in Lemma 4.

For the first case we have

$$\begin{aligned}
\mathcal{E}(\widehat{G}_{\mathcal{G},\alpha}) &= \mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu_\alpha^* + \alpha - P(\widehat{G}_{\mathcal{G},\alpha}) \\
&\leq \mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu_\alpha^* + 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta) \\
&= \inf_{G \in \widehat{\mathcal{G}}_\alpha} \left\{ \mu(G) - \mu_\alpha^* + 2\phi(G, S, \delta) \right\} \\
&\leq \inf_{G \in \mathcal{G}_\alpha} \left\{ \mu(G) - \mu_\alpha^* + 2\phi(G, S, \delta) \right\} \\
&\leq \left(1 + \frac{1}{\gamma_\alpha}\right) \inf_{G \in \mathcal{G}_\alpha} \left\{ \mu(G) - \mu_\alpha^* + 2\phi(G, S, \delta) \right\}.
\end{aligned}$$

The first inequality follows from $S \in \overline{\Theta_P}$. The next line comes from the definition of $\widehat{G}_{\mathcal{G},\alpha}$. The second inequality follows from $S \in \overline{\Omega_P}$, from which it follows that $\mathcal{G}_\alpha \subset \widehat{\mathcal{G}}_\alpha$. The final step is trivial (this constant is needed for case 3).

For the second case, $\mu(\widehat{G}_{\mathcal{G},\alpha}) \geq \mu_\alpha^*$ and $P(\widehat{G}_{\mathcal{G},\alpha}) \geq \alpha$, note

$$\begin{aligned}
\mathcal{E}(\widehat{G}_{\mathcal{G},\alpha}) &= \mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu_\alpha^* \\
&\leq \mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu_\alpha^* + 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta)
\end{aligned}$$

and proceed as in the first case.

For the third case, $\mu(\widehat{G}_{\mathcal{G},\alpha}) < \mu^*_\alpha$ and $P(\widehat{G}_{\mathcal{G},\alpha}) < \alpha$, we rely on the following lemmas.

Lemma 18 *Let* $\varepsilon > 0$. *Then*

$$\mu^*_\alpha - \mu^*_{\alpha-\varepsilon} \leq \frac{\varepsilon}{\gamma_\alpha}.$$

Proof By assumptions **A1** and **A2**, there exist MV-sets $G^*_{\alpha-\varepsilon}$ and G^*_α such that

$$\int_{G^*_\alpha} f(x)d\mu(x) = \alpha$$

and

$$\int_{G^*_{\alpha-\varepsilon}} f(x)d\mu(x) = \alpha - \varepsilon.$$

Furthermore, we may choose $G^*_{\alpha-\varepsilon}$ and G^*_α such that $G^*_{\alpha-\varepsilon} \subset G^*_\alpha$. Thus

$$\begin{aligned}
\varepsilon &= \int_{G^*_\alpha} f(x)d\mu(x) - \int_{G^*_{\alpha-\varepsilon}} f(x)d\mu(x) \\
&= \int_{G^*_\alpha \setminus G^*_{\alpha-\varepsilon}} f(x)d\mu(x) \\
&\geq \gamma_\alpha \mu(G^*_\alpha \setminus G^*_{\alpha-\varepsilon}) \\
&= \gamma_\alpha(\mu^*_\alpha - \mu^*_{\alpha-\varepsilon})
\end{aligned}$$

and the result follows. ■

Lemma 19 *If* $S \in \overline{\Omega_P}$ *and* $G \in \widehat{\mathcal{G}}_\alpha$, *then*

$$\mu^*_\alpha - \mu(G) \leq \frac{2}{\gamma_\alpha} \cdot \phi(G,S,\delta).$$

Proof Denote $\varepsilon = 2\phi(G,S,\delta)$. Since $S \in \overline{\Omega_P}$ and $G \in \widehat{\mathcal{G}}_\alpha$, we know

$$P(G) \geq \widehat{P}(G) - \frac{1}{2}\varepsilon \geq \alpha - \varepsilon.$$

In other words, $G \in \mathcal{G}_{\alpha-\varepsilon}$. Therefore, $\mu(G) \geq \mu^*_{\alpha-\varepsilon}$ and it suffices to bound $\mu^*_\alpha - \mu^*_{\alpha-\varepsilon}$. Now apply the preceding lemma. ■

It now follows that

$$\begin{aligned}
\mathcal{E}(\widehat{G}_{\mathcal{G},\alpha}) &= \alpha - P(\widehat{G}_{\mathcal{G},\alpha}) \\
&\leq 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta) \\
&= \mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha + \mu^*_\alpha - \mu(\widehat{G}_{\mathcal{G},\alpha}) + 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta) \\
&\leq \mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha + \left(1 + \frac{1}{\gamma_\alpha}\right) 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta) \\
&\leq \left(1 + \frac{1}{\gamma_\alpha}\right) \left(\mu(\widehat{G}_{\mathcal{G},\alpha}) - \mu^*_\alpha + 2\phi(\widehat{G}_{\mathcal{G},\alpha}, S, \delta)\right) \\
&= \left(1 + \frac{1}{\gamma_\alpha}\right) \inf_{G \in \widehat{\mathcal{G}}_\alpha} \left\{ \mu(G) - \mu^*_\alpha + 2\phi(G, S, \delta) \right\} \\
&\leq \left(1 + \frac{1}{\gamma_\alpha}\right) \inf_{G \in \mathcal{G}_\alpha} \left\{ \mu(G) - \mu^*_\alpha + 2\phi(G, S, \delta) \right\}
\end{aligned}$$

The first inequality follows from Lemma 17. The second inequality is by Lemma 19. The next to last line follows from the definition of $\widehat{G}_{\mathcal{G},\alpha}$, and the final step is implied by $S \in \overline{\Omega_P}$ as in case 1. This completes the proof.

Appendix E. The Rademacher Penalty for Partition-Based Sets

In this appendix we show how the conditional Rademacher penalty introduced in Section 2.3 can be evaluated for a class $\mathcal{G}$ based on a fixed partition. The authors thank Gilles Blanchard for pointing out the properties that follow. Let $\pi = \{A_1, \ldots, A_k\}$ be a fixed, finite partition of $\mathcal{X}$, and let $\mathcal{G}$ be the set of all sets formed by taking the union of cells in π. Thus $|\mathcal{G}| = 2^k$ and every $G \in \mathcal{G}$ is specified by a k-length string of binary digits $\ell(A_1), \ldots, \ell(A_k)$, with $\ell(A) = 1$ if and only if $A \subset G$.

The conditional Rademacher penalty may be rewritten as follows:

$$\begin{aligned}
\frac{2}{n}\mathbf{E}_{(\sigma_i)}\left[\sup_{G \in \mathcal{G}} \sum_{i=1}^n \sigma_i \mathbb{I}(X_i \in G)\right] &= \frac{2}{n}\mathbf{E}_{(\sigma_i)}\left[\sup_{\ell(A): A \in \pi} \sum_{i=1}^n \sigma_i \ell(A)\right] \\
&= \frac{2}{n}\sum_{A \in \pi} \mathbf{E}_{(\sigma_i)}\left[\sup_{\ell(A)} \sum_{i: X_i \in A} \sigma_i \ell(A)\right] \\
&=: \sum_{A \in \pi} \psi(A).
\end{aligned}$$

Thus the penalty is additive (modulo the delta term). Now consider a fixed cell A:

$$\begin{aligned}
\psi(A) &= \frac{2}{n}\mathbf{E}_{(\sigma_i)}\left[\sup_{\ell(A)}\sum_{i:X_i\in A}\sigma_i\ell(A)\right] \\
&= \frac{1}{n}\mathbf{E}_{(\sigma_i)}\left[\sup_{\ell(A)}\sum_{i:X_i\in A}\sigma_i(2\ell(A)-1)\right] \\
&= \frac{1}{n}\mathbf{E}_{(\sigma_i)}\left[\sup_{\ell(A)}(2\ell(A)-1)\sum_{i:X_i\in A}\sigma_i\right] \\
&= \frac{1}{n}\mathbf{E}_{(\sigma_i)}\left[\left|\sum_{i:X_i\in A}\sigma_i\right|\right].
\end{aligned}$$

Now let $\mathrm{bin}(M,p,m)=\binom{M}{m}p^m(1-p)^{M-m}$ be the probability of observing m successes in a sequence of M Bernoulli trials having success probability p. Then this last expression can be computed explicitly as

$$\psi(A)=\frac{1}{n}\sum_{i=0}^{n_A}\mathrm{bin}(n_A,1/2,i)|n_A-2i|,$$

where $n_A=|\{i:X_i\in A\}|$. This is the penalty used in the histogram experiments (after the delta term is included).

A more convenient and intuitive penalty may be obtained by bounding

$$\begin{aligned}
\psi(A) &= \frac{1}{n}\mathbf{E}_{(\sigma_i)}\left[\left|\sum_{i:X_i\in A}\sigma_i\right|\right] \\
&\le \frac{1}{n}\mathbf{E}_{(\sigma_i)}\left[\left(\sum_{i:X_i\in A}\sigma_i\right)^2\right]^{\frac{1}{2}} \\
&= \frac{1}{n}\mathbf{E}_{(\sigma_i)}\left[\sum_{i:X_i\in A}\sigma_i^2\right]^{\frac{1}{2}} \\
&= \sqrt{\frac{\widehat{P}(A)}{n}},
\end{aligned}$$

where the inequality is Jensen's. Moreover, by the Khinchin-Kahane inequality (see, e.g., Ledoux and Talagrand, 1991, Lemma 4.1), the converse inequality holds with a factor $\sqrt{2}$, so the bound is tight up to this factor. This is the "Rademacher" penalty employed in the dyadic decision tree experiments.

References

A. Baillo, J. A. Cuesta-Albertos, and A. A. Cuevas. Convergence rates in nonparametric estimation of level sets. *Stat. Prob. Letters*, 53:27–35, 2001.

P. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2002.

S. Ben-David and M. Lindenbaum. Learning distributions by their density levels: a paradigm for learning without a teacher. *J. Comp. Sys. Sci.*, 55:171–182, 1997.

G. Blanchard, C. Schäfer, and Y. Rozenholc. Oracle bounds and exact algorithm for dyadic classification trees. In J. Shawe-Taylor and Y. Singer, editors, *Learning Theory: 17th Annual Conference on Learning Theory, COLT 2004*, pages 378–392. Springer-Verlag, Heidelberg, 2004.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In O. Bousquet, U.v. Luxburg, and G. Rtsch, editors, *Advanced Lectures in Machine Learning*, pages 169–207. Springer, 2004.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

A. Cannon, J. Howse, D. Hush, and C. Scovel. Learning with the Neyman-Pearson and min-max criteria. Technical Report LA-UR 02-2951, Los Alamos National Laboratory, 2002. URL `http://www.c3.lanl.gov/~kelly/ml/pubs/2002_minmax/paper.pdf`.

A. Cohen, W. Dahmen, I. Daubechies, and R. A. DeVore. Tree approximation and optimal encoding. *Applied and Computational Harmonic Analysis*, 11(2):192–226, 2001.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, 1991.

A. Cuevas and R. Fraiman. A plug-in approach to support estimation. *Ann. Stat.*, 25:2300–2312, 1997.

A. Cuevas and A. Rodriguez-Casal. Set estimation: An overview and some recent developments. *Recent advances and trends in nonparametric statistics*, pages 251–264, 2003.

R. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

D. Donoho. Wedgelets: Nearly minimax estimation of edges. *Ann. Stat.*, 27:859–897, 1999.

R. Durrett. *Probability: Theory and Examples*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1991.

J. Hartigan. Estimation of a convex density contour in two dimensions. *J. Amer. Statist. Assoc.*, 82 (397):267–270, 1987.

X. Huo and J. Lu. A network flow approach in finding maximum likelihood estimate of high concentration regions. *Computational Statistics and Data Analysis*, 46(1):33–56, 2004.

J. Klemelä. Complexity penalized support estimation. *J. Multivariate Anal.*, 88:274–297, 2004.

V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Trans. Inform. Theory*, 47:1902–1914, 2001.

J. Langford. Tutorial on practical prediction theory for classification. *J. Machine Learning Research*, 6:273–306, 2005.

M. Ledoux and M. Talagrand. *Probability in Banach spaces*. Springer-Verlag, Berlin, 1991.

G. Lugosi and K. Zeger. Concept learning using complexity regularization. *IEEE Trans. Inform. Theory*, 42(1):48–54, 1996.

G. Lugosi and K. Zeger. Nonparametric estimation using empirical risk minimization. *IEEE Trans. Inform. Theory*, 41(3):677–687, 1995.

D. Müller and G Sawitzki. Excess mass estimates and tests for multimodality. *J. Amer. Statist. Assoc.*, 86(415):738–746, 1991.

A. Muñoz and J. M. Moguerza. Estimation of high-density regions using one-class neighbor machines. *IEEE Trans. Patt. Anal. Mach. Intell.*, 28:476–480, 2006.

D. Nolan. The excess mass ellipsoid. *J. Multivariate Analysis*, 39:348–371, 1991.

J. Nunez-Garcia, Z. Kutalik, K.-H.Cho, and O. Wolkenhauer. Level sets and minimum volume sets of probability density functions. *Approximate Reasoning*, 34:25–47, Sept. 2003.

W. Polonik. Measuring mass concentrations and estimating density contour cluster–an excess mass approach. *Ann. Stat.*, 23(3):855–881, 1995.

W. Polonik. Minimum volume sets and generalized quantile processes. *Stochastic Processes and their Applications*, 69:1–24, 1997.

T. W. Sager. An iterative method for estimating a multivariate mode and isopleth. *J. Am. Stat. Asso.*, 74:329–339, 1979.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1472, 2001.

C. Scott and R. Nowak. Learning minimum volume sets. Technical Report ECE-05-2, UW-Madison, 2005a. URL `http://www.stat.rice.edu/~cscott`.

C. Scott and R. Nowak. A Neyman-Pearson approach to statistical learning. *IEEE Trans. Inform. Theory*, 51(8):3806–3819, 2005b.

C. Scott and R. Nowak. Minimax-optimal classification with dyadic decision trees. *IEEE Trans. Inform. Theory*, pages 1335–1353, April 2006.

I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *J. Machine Learning Research*, 6:211–232, 2005.

A. B. Tsybakov. On nonparametric estimation of density level sets. *Ann. Stat.*, 25:948–969, 1997.

V. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, New York, 1982.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

R. Vert and J.-P. Vert. Consistency and convergence rates of one-class SVM and related algorithms. Technical Report 1414, Universit Paris-Sud, 2005.

G. Walther. Granulometric smoothing. *Ann. Stat.*, 25:2273–2299, 1997.

R. Willett and R. Nowak. Minimax optimal level set estimation. submitted to *IEEE Trans. Image Proc.*, 2006. URL `http://www.ee.duke.edu/~willett/`.

R. Willett and R. Nowak. Minimax optimal level set estimation. In *Proc. SPIE, Wavelets XI*, 31 July - 4 August, San Diego, CA, USA, 2005.

Journal of Machine Learning Research 7 (2006) 705–732 Submitted 7/05; Revised 12/05; Published 5/06

Some Theory for Generalized Boosting Algorithms

Peter J. Bickel BICKEL@STAT.BERKELEY.EDU
Department of Statistics
University of California at Berkeley
Berkeley, CA 94720, USA

Ya'acov Ritov (corresponding author) YAACOV.RITOV@HUJI.AC.IL
Department of Statistics and The Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem
91905 Jerusalem, Israel

Alon Zakai ALONZAKA@POB.HUJI.AC.IL
The Interdisciplinary Center for Neural Computation
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Editor: Bin Yu

Abstract

We give a review of various aspects of boosting, clarifying the issues through a few simple results, and relate our work and that of others to the minimax paradigm of statistics. We consider the population version of the boosting algorithm and prove its convergence to the Bayes classifier as a corollary of a general result about Gauss-Southwell optimization in Hilbert space. We then investigate the algorithmic convergence of the sample version, and give bounds to the time until perfect separation of the sample. We conclude by some results on the statistical optimality of the L_2 boosting.

Keywords: classification, Gauss-Southwell algorithm, AdaBoost, cross-validation, non-parametric convergence rate

1. Introduction

We consider a standard classification problem: Let $(X,Y),(X_1,Y_1),\ldots,(X_n,Y_n)$ be an i.i.d. sample, where $Y_i \in \{-1,1\}$ and $X_i \in \mathcal{X}$. The goal is to find a good classification rule, $\mathcal{X} \to \{-1,1\}$.

The AdaBoost algorithm was originally defined, Schapire (1990), Freund (1995), and Freund and Schapire (1996) as an algorithm to construct a good classifier by a "weighted majority vote" of simple classifiers. To be more exact, let $\mathcal{H}$ be a set of simple classifiers. The AdaBoost classifier is given by $\operatorname{sgn}\left(\sum_{m=1}^{M} \lambda_m h_m(x)\right)$, where $\lambda_m \in \mathbb{R}$, $h_m \in \mathcal{H}$, are found sequentially by the following algorithm:

0. Let $c_1 = c_2 = \cdots = c_n = 1$, and set $m = 1$.

1. Find $h_m = \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} c_i h(X_i) Y_i$. Set

$$\lambda_m = \frac{1}{2}\log\left(\frac{\sum_{i=1}^{n} c_i + \sum_{i=1}^{n} c_i h_m(X_i) Y_i}{\sum_{i=1}^{n} c_i - \sum_{i=1}^{n} c_i h_m(X_i) Y_i}\right) = \frac{1}{2}\log\left(\frac{\sum_{h_m(X_i)=Y_i} c_i}{\sum_{h_m(X_i)\neq Y_i} c_i}\right).$$

2. Set $c_i \leftarrow c_i \exp\bigl(-\lambda_m h_m(X_i)Y_i\bigr)$, and $m \leftarrow m+1$, If $m \le M$, return to step 1.

M is unspecified and can be arbitrarily large.

The success of these methods on many data sets and their "resistance to overfitting"—the test set error continues to decrease even after all the training set observations were classified correctly, has led to intensive investigation to which this paper contributes.

Let $\mathcal{F}_\infty$ be the linear span of $\mathcal{H}$. That is,

$$\mathcal{F}_\infty = \bigcup_{k=1}^{\infty} \mathcal{F}_k, \text{ where } \mathcal{F}_k = \Bigl\{ \sum_{j=1}^{k} \lambda_j h_j : \lambda_j \in \mathbb{R},\ h_j \in \mathcal{H},\ 1 \le j \le k \Bigr\}.$$

A number of workers have noted, Breiman (1998,1999), Friedman, Hastie and Tibshirani (2000), Mason, Bartlett, Baxter and Frean (2000), and Schapire and Singer (1999), that the AdaBoost classifier can be viewed as $\operatorname{sgn}\bigl(F(X)\bigr)$, where F is found by a greedy algorithm minimizing

$$n^{-1} \sum_{i=1}^{n} \exp\bigl(-Y_i F(X_i)\bigr)$$

over $\mathcal{F}_\infty$.

¿From this point of view, the algorithm appeared to be justifiable, since as was noted in Breiman (1999) and Friedman, Hastie, and Tibshirani (2000), the corresponding expression $E\exp\bigl(-YF(X)\bigr)$, obtained by replacing the sum by expectation, is minimized by

$$F(X) = \frac{1}{2}\log\Bigl(P(Y=1|X)/P(Y=-1|X)\Bigr),$$

provided the linear span $\mathcal{F}_\infty$ is dense in the space $\mathcal{F}$ of all functions in a suitable way. However, it was also noted that the empirical optimization problem necessarily led to rules which would classify every training set observation correctly and hence not approach the Bayes rule whatever be n, except in very special cases. Jiang (2003) established that, for observation centered stumps, the algorithm converged to nearest neighbor classification, a good but rarely optimal rule.

In another direction, the class of objective functions $W(\cdot)$ that can be considered was extended by Friedman, Hastie, and Tibshirani (2000) to other W, in particular, $W(t) = \log(1+e^{-2t})$, whose empirical version they identified with logistic regression in statistics, and $W(t) = -2t + t^2$, which they referred to as "L_2 Boosting" and has been studied, under the name "matching pursuit", in the signal processing community. For all these objective functions, the population optimization of $EW\bigl(YF(X)\bigr)$ over $\mathcal{F}$ leads to a solution such that $\operatorname{sgn} F(X)$ is the Bayes rule. Friedman et al. also introduced consideration of other algorithms for the empirical optimization problem. Lugosi and Vayatis (2004) added regularization, changing the function whose expectation (both empirically and in the population) is to be minimized from $W\bigl(YF(X)\bigr)$ to $W_n\bigl(YF(X)\bigr)$ where $W_n \to W$ as $n \to \infty$. Bühlmann and Yu (2003) considered L_2 boosting starting from very smooth functions. We shall elaborate on this later.

We consider the behavior of the algorithm as applied to the sample $(Y_1,X_1),\ldots,(Y_n,X_n)$, as well to the "population", that is when means are replaced by expectations and sums by probabilities. The structure of, and the differences between, the population and sample versions of the optimization problem has been explored in various ways by Jiang (2003), Zhang and Yu (2003), Bühlmann (2003), Bartlett, Jordan, and McAuliffe (2003), Bickel and Ritov (2003).

Our goal in this paper is

1. To clarify the issues through a few simple results.

2. To relate our work and that of Bühlmann (2003), Bühlmann and Yu (2003), Lugosi and Vayatis (2004), Zhang (2004), Zhang and Yu (2003) and Bartlett, Jordan, and McAuliffe (2003) to the minimax results of Mammen and Tsybakov (1999), Baraud (2001) and Tsybakov (2001).

In Section 2 we will discuss the population version of the basic boosting algorithms and show how their convergence and that of more general greedy algorithms can be derived from a generalization of Theorem 3 of Mallat and Zhang (1993) with a simple proof. The result can, we believe, also be derived from the even more general theorem of Zhang and Yu (2003), but our method is simpler and the results are transparent.

In Section 3 we show how Bayes consistency of various sample algorithms when suitably stopped or of sample algorithms based on minimization of a regularized W follow readily from population convergence of the algorithms and indicate how test bed validation can be used to do this in a way leading to optimal rates (in Section 4).

In Section 5 we address the issue of bounding the time to perfect separation of the different boosting algorithm (including the standard AdaBoost).

Finally in Section 6 we show how minimax rate results for estimating $E(Y|X)$ may be attained for a "sieve" version of the L_2 boosting algorithm, and relate these to results of Baraud (2001), Lugosi and Vayatis (2004), Bühlmann and Yu (2003), Barron, Birgé, Massart(1999) and Bartlett, Jordan and McAuliffe (2003). We also discuss the relation of these results to classification theory.

2. Boosting "Population" Theorem

We begin with a general theorem on Gauss-Southwell optimization in vector space. It is, in part, a generalization of Theorem 1 of Mallat and Zhang (1993) with a simpler proof. A second part relates to procedures in which the step size is regularized cf. Zhang and Yu (2003) and Bartlett et al. (2003). We make the boosting connection after its statement.

Let w be a real, bounded from below, convex function on a vector space $\mathbb{H}$. Let $\mathcal{H} = \mathcal{H}' \cup (-\mathcal{H}')$, where $\mathcal{H}'$ is a subset of $\mathbb{H}$ whose members are linearly independent, with linear span $\mathcal{F}_\infty = \{\sum_{m=1}^{k} \lambda_m h_m : \lambda_j \in \mathbb{R},\ h_j \in \mathcal{H},\ 1 \le j \le k, 1 \le k < \infty\}$. We assume that $\mathcal{F}_\infty$ is dense in $\mathbb{H}$, at least in the sense that $\{w(f) :\ f \in \mathcal{F}_\infty\}$ is dense in the image of w. We define two relaxed Gauss-Southwell "algorithms".

Algorithm I: For $\alpha \in (0,1]$, and given $f_1 \in \mathbb{H}$, find inductively $f_2, f_3, \ldots, \ldots$ by, $f_{m+1} = f_m + \lambda_m h_m$, $\lambda_m \in \mathbb{R}$, $h_m \in \mathcal{H}$ and

$$w(f_m + \lambda_m h_m) \le \alpha \min_{\lambda \in \mathbb{R},\, h \in \mathcal{H}} w(f_m + \lambda h) + (1-\alpha) w(f_m)\ . \tag{1}$$

Generalize Algorithm I to :

Algorithm II: Like Algorithm I, but replace (1) by

$$w(f_m + \lambda_m h) + \gamma \lambda_m^2 \le \alpha \min_{\lambda \in \mathbb{R},\, h \in \mathcal{H}} (w(f_m + \lambda h) + \gamma \lambda^2) + (1-\alpha) w(f_m)\ .$$

There are not algorithms in the usual sense since they do not specify a unique sequence of iterations but our theorems will apply to any sequence generated in this way. Technically, this scheme

is used in the proof of Theorem 3. The standard boosting algorithms theoretically correspond to $\alpha = 1$, although in practice, since numerical minimization is used, α may equal 1 only approximately. Our generalization makes for a simple proof and covers the possibility that the minimum of $w(f_m + \lambda h)$ over $\mathcal{H}$ and $\mathbb{R}$ is not assumed, or multiply assumed. Let $\omega_0 = \inf_{f \in \mathcal{F}_\infty} w(f) > -\infty$. Let $w'(f;h)$ the linear operator of the Gataux derivative at $f \in \mathcal{F}_\infty$ in the direction $h \in \mathcal{F}_\infty$: $w'(f;h) = \partial w(f+\lambda h)/\partial\lambda\big|_{\lambda=0}$, and let $w''(f;h)$ be the second derivative of w at f in the direction h: $w''(f,h) \equiv \partial^2 w(f+\lambda h)/\partial\lambda^2\big|_{\lambda=0}$ (both derivative are assumed to exist). We consider the following conditions.

GS1. For any c_1 and c_2 such that $\omega_0 < c_1 < c_2 < \infty$,

$$0 < \inf\{w''(f,h) : c_1 < w(f) < c_2,\ h \in \mathcal{H}\} \leq \sup\{w''(f,h) : w(f) < c_2,\ h \in \mathcal{H}\} < \infty.$$

GS2. For any $c_2 < \infty$,

$$\sup\{w''(f,h) :\ w(f) < c_2,\ h \in \mathcal{H}\} < \infty\,.$$

Theorem 1 *Under Assumption* GS1*, any sequence of functions generated according to Algorithm I satisfies:*

$$w(f_m) \leq \omega_0 + c_m$$

and if $c_m > 0$*:*

$$w(f_m) - w(f_{m+1}) \geq \xi(w(f_m)) > 0$$

where the sequence $c_m \to 0$ *and the function* $\xi(\cdot)$ *depend only on* α*, the initial points of the iterates, and* $\mathcal{H}$*. The same conclusion holds under Condition* GS2 *for any sequence* f_m *generated according to algorithm II.*

The proof can be found in Appendix A.

Remark:

1. Condition GS2 of Theorem 1 guarantees that $\sum_{m=1}^{\infty} \lambda_m^2 < \infty$. It can be replaced by any other condition that guarantees the same, for example, limiting the step size, replacing the penalty by other penalties, etc.

2. It will be clear from the proof in Appendix A that if w'' is bounded away from 0 and ∞ then c_m is of order $(\log m)^{-\frac{1}{2}}$ so that we, in fact, have an approximation rate – but it is so slow as to be essentially useless. On the other hand, with strong conditions such as orthonormality of the elements of $\mathcal{H}$, and $\mathcal{H}$ a classical approximation class such as trigonometric functions we expect, with L_2 boosting, to obtain rates such as $m^{-1/2}$ or better.

Let $(X,Y) \sim P$, $X \in \mathcal{X}$, $Y \in \{-1,1\}$. Let $\mathcal{H} \subset \{h : \mathcal{X} \to [-1,1]\}$ be a symmetric set of functions. In particular, $\mathcal{H}$ can, but need not, be a set of classifiers such as trees with

$$\mathcal{H} = -\mathcal{H}\,. \tag{2}$$

Given a loss function $W : \mathbb{R} \to \mathbb{R}^+$, we consider a greedy sequential procedure for finding a function F that minimizes $EW\big(YF(X)\big)$. That is, given $F_0 \in \mathcal{H}$ fixed, we define for $m \geq 0$:

$$\begin{aligned}
\lambda_m(h) &= \arg\min_{\lambda \in \mathbb{R}} EW\Big(Y\big(F_m(X) + \lambda h(X)\big)\Big) \\
h_m &= \arg\min_{h \in \mathcal{H}} EW\Big(Y\big(F_m(X) + \lambda_m(h) h(X)\big)\Big) \\
F_{m+1} &= F_m + \lambda_m(h_m) h_m.
\end{aligned}$$

Assume, wlog (without loss of generality), by shifting and rescaling, that $W(0) = -W'(0) = 1$. Note that by Bartlett et al. (2003), $W'(0) < 0$ is necessary and sufficient for population consistency defined below. We can suppose again wlog in view of (2), that $\lambda_m \geq 0$. Define $\mathcal{F}_k$ and $\mathcal{F}_\infty$ as in Section 1 and let $\mathcal{F} \equiv \bar{\mathcal{F}}_\infty$ be the closure of $\mathcal{F}_\infty$ in convergence in probability:

$$\begin{aligned}
\mathcal{F} &\equiv \big\{F : \exists F_m \in \mathcal{F}_m,\ F_m(X) \xrightarrow{p} F(X)\big\} \\
F_\infty &\equiv \arg\min_{F \in \mathcal{F}} EW\big(YF(X)\big)
\end{aligned}$$

If $\operatorname{sgn} F_\infty$ is the Bayes rule for 0-1 loss, we say that F_∞ is population consistent for classification, "calibrated" in the Bartlett et al. terminology. Let

$$\begin{aligned}
p(X) &\equiv P(Y = 1|X) \\
\widetilde{W}(x,d) &\equiv p(x)W\big(d\big) + \big(1 - p(x)\big)W\big(-d\big). \\
\widetilde{W}(F) &\equiv \widetilde{W}(X, F(X))
\end{aligned}$$

By the assumptions below F_∞ is the unique function such that $\widetilde{W}'(F_\infty) = 0$ with probability 1, where $\widetilde{W}'(F) = \widetilde{W}'(X, F(X))$ and $\widetilde{W}'(x,d) = \partial W(x,d)/\partial d$. Define $\widetilde{W}''$ similarly.

Here are some conditions.

P1. $P[p(X) = 0 \text{ or } 1] = 0$.

P2. W is twice differentiable and convex on $\mathbb{R}$.

P3. $\mathcal{H}$ is closed and compact in the weak topology. $\mathcal{F}$ is the set of all measurable functions on $\mathcal{X}$.

P4. $\widetilde{W}''(F)$ is bounded above and below on $\{F :\ c_1 < \widetilde{W}(F) < c_2\}$ for all c_1, c_2 such that

$$\inf_{F \in \mathcal{F}} E\widetilde{W}(F) < c_1 < c_2 < E\widetilde{W}(F_0).$$

P5. $F_\infty \in L_2(P)$.

Note that P1 and P2 imply that $\widetilde{W}(x,d) \to \infty$ as $|d| \to \infty$, which ensures that F_∞ is finite almost anywhere. Condition P1, which says that no point can be classified with absolute certainty, is only needed technically to ensure that $\widetilde{W}(x,d) \to \infty$ as $|d| \to \infty$, even if W itself is monotone. It is not needed for L_2 boosting.

Conditions P2 and P4 ensure that along the optimizing path W behaves locally like $W_0(t) = -2t + t^2$ corresponding to L_2 boosting. They are more stringent than we would like and, in particular,

rule out W such as the "hinge" appearing in SVM. More elaborate arguments such as those of Zhang and Yu (2003) and Bartlett et al. (2003) can give somewhat better results.

The functions commonly appearing in boosting such as, $W_1(t) = e^{-t}$, $W_2(t) = -2t + t^2$, $W_3(t) = -\log(1+e^{-2t})$ satisfy condition P4 if P1 also holds. This is obvious for W_2. For W_1 and W_3, it is clear that P4 holds, if P1 does, since otherwise $E\widetilde{W}\big(YF_m(X)\big) \to \infty$. The conclusions of Theorem 2 continue to hold if $h \in \mathcal{H} \implies |h| \geq \delta > 0$ since then below $w''(F;h) = Eh^2(X)\widetilde{W}\big(F(X)\big) \geq \delta^2 E\widetilde{W}(F(X))$ and P4 follows. Note that if $|h| \not\equiv 1$ the λ optimization step requires multiplying λ^2 by $Eh^2(x)$.

We have,

Theorem 2 *If $\mathcal{H}$ is a set of classifiers, $(h^2 \equiv 1)$ and Assumptions* P2 – P5 *hold, then*

$$F_m(X) \xrightarrow{P} F_\infty(X) \ ,$$

and the misclassification error, $P\big(YF_m(X) \leq 0\big) \to P[YF_\infty(X) \leq 0]$, the Bayes risk.

Proof Identify $w(F) = EW\big(YF(X)\big) = E\widetilde{W}\big(F(X)\big)$. Then,

$$w''(F,h) = Eh^2(X)\widetilde{W}''\big(F(X)\big) = E\widetilde{W}''\big(F(X)\big)$$

and (P4) can be identified with condition GS1 of Theorem 1. Thus,

$$E\widetilde{W}\big(F_m(X)\big) \to E\widetilde{W}(F_\infty(X)) \ .$$

Since,

$$E\widetilde{W}\big(F_m(X)\big) - E\widetilde{W}\big(F_\infty(X)\big) = E\left((F_\infty - F_m)^2 \int_0^1 \widetilde{W}''\big((1-\lambda)(X)F_\infty(X) + \lambda F_m(X)\big)\lambda d\lambda\right) \to 0 \ ,$$

the conclusion of Theorem 2 follows from (P4). The second assertion is immediate. ■

3. Consistency of the Boosting Algorithm

In this section we study the Bayes consistency properties of the sample versions of the boosting algorithms we considered in Section 2. In particular, we shall

(i) Show that under mild additional conditions, there will exist a random sequence $m_n \to \infty$ such that $\hat{F}_{m_n} \xrightarrow{P} F_\infty$, where $\hat{F}_m$ is defined below as the mth sample iterate, and moreover, that such a sequence can be determined using the data.

(ii) Comment on the relationship of this result to optimization for penalized versions of W. The difference is that the penalty forces $m < \infty$ to be optimal while with us, cross-validation (or a test bed sample) determines the stopping point. We shall see that the same dichotomy applies later, when we "boost" using the method of sieves for nonparametric regression studied by Barron, Birge and Massart (1999) and Baraud (2001).

3.1 The Golden Chain Argument

Here is a very general framework. This section is largely based on Bickel and Ritov (2003).

Let $\Theta_1 \subset \Theta_2 \subset \ldots$ be a sequence of sets contained in a separable metric space, $\Theta = \overline{\cup \Theta_m}$ where $\bar{}$ denotes closure. Let $\Pi_m : \Theta_m \to 2^{\Theta_{m+1}}$ be a sequence of point to set mappings. Let K be a target function, and $\vartheta_\infty = \arg\min_{\vartheta\in\Theta} K(\vartheta)$. Finally, let $\hat{K}_n$ be a sample based approximation of K. We assume:

G1. $K : \Theta \to \mathbb{R}$ is strictly convex, with a unique minimizer ϑ_∞.

Our result is applicable to loosely defined algorithms. In particular we want to be able to consider the result of the algorithm applied to the data as if it were generated by a random algorithm applied to the population. We need therefore, the following definitions. Let $\mathcal{S}(\vartheta_0, \alpha)$ be the set of all sequences $\bar{\vartheta}_m \in \Theta_m$, $m = 0, 1, \ldots$ with $\bar{\vartheta}_0 = \vartheta_0$ and satisfying:

$$\bar{\vartheta}_{m+1} \in \Pi_m(\bar{\vartheta}_m)$$
$$K(\bar{\vartheta}_{m+1}) \le \alpha \inf_{\vartheta\in\Pi_m(\bar{\vartheta}_m)} K(\vartheta) + (1-\alpha)K(\bar{\vartheta}_m).$$

The resemblance to Gauss-Southwell Algorithm I and the boosting procedures is not accidental. Suppose the following uniform convergence criterion is satisifed:

G2. If $\{\bar{\vartheta}_m\} \in \mathcal{S}(\vartheta_0, \alpha)$ with any initial ϑ_0, then $K(\bar{\vartheta}_m) - K(\bar{\vartheta}_{m+1}) \ge \xi\big(K(\bar{\vartheta}_m) - K(\vartheta_\infty)\big)$, for $\xi(\cdot) > 0$ strictly increasing, and $K(\bar{\vartheta}_m) - K(\vartheta_\infty) \le c_m$ where $c_m \to 0$ uniformly over $\mathcal{S}(\vartheta_0, \alpha)$.

In boosting, given P, $\Theta = \{F(X), F \in \tilde{\mathcal{F}}\}$ with a metric of convergence in probability, $\Theta_m = \{\sum_{j=1}^m \lambda_j h_j, h_j \in \mathcal{H}\}$, $\Pi_m(F) = \Pi(F) = \{F + \lambda h, \lambda \in \mathbb{R}, h \in \mathcal{H}\}$, and $K(F) = \mathrm{E}W\big(YF(X)\big)$. Condition G2, follows from the conclusion of Theorem 1.

Now suppose $\hat{K}_n(\cdot)$ is a sequence of random functions on Θ, empirical entities that resemble the population K. Let $\hat{\mathcal{S}}_n(\vartheta_0, \alpha')$ be the set of all sequences $\hat{\vartheta}_{0,n}, \hat{\vartheta}_{1,n} \ldots$, such that $\hat{\vartheta}_{0,n} = \vartheta_0$, and

$$\hat{\vartheta}_{m+1,n} \in \Pi_m(\hat{\vartheta}_{m,n})$$
$$\hat{K}_n(\hat{\vartheta}_{m+1,n}) \le \alpha' \min\{\hat{K}_n(\vartheta) : \ \vartheta \in \Pi_m(\hat{\vartheta}_{m,n})\} + (1-\alpha')\hat{K}_n(\hat{\vartheta}_{m,n}).$$

We assume

G3. $\hat{K}_n$ is convex, and for all integer m, $\sup\{|\hat{K}_n(\vartheta) - K(\vartheta)| : \vartheta \in A_m\} \xrightarrow{\text{a.s.}} 0$ as $n \to \infty$, for a sequence $A_m \subset \Theta_m$ such that $P(\hat{\vartheta}_{m,n} \in A_m) \to 1$.

In boosting, $\hat{K}_n(F) = n^{-1}\sum_{i=1}^n W\big(Y_i F(X_i)\big)$, $K(F) = E_p\big(YF(X)\big)$

The sequence $\{\bar{\vartheta}_m\}$ is the golden chain we try to follow using the obscure information in the sample.

We now state and prove,

Theorem 3 *If assumptions* G1– G3 *hold, and* $\alpha' \in (0, 1]$, *then for any sequence* $\{\hat{\vartheta}_{m,n}\} \in \hat{\mathcal{S}}(\vartheta_0, \alpha')$, *there exists a subsequence* $\{\hat{m}_n\}$ *such that* $K(\hat{\vartheta}_{\hat{m}_n,n}) \xrightarrow{\text{p}} K(\vartheta_\infty)$.

Proof

Fix ϑ_0 and α, $\alpha < \alpha'$. Let $M_n \to \infty$ be some sequence, and let $\hat{m}_n = \arg\min_{m \le M_n} K(\hat{\vartheta}_{m,n})$. We need to prove that $K(\hat{\vartheta}_{\hat{m}_n,n}) \xrightarrow{\text{p}} K(\vartheta_\infty)$. We will prove this by contradiction. Suppose otherwise:

$$\inf_{m \le M_n} K(\hat{\vartheta}_{m,n}) - K(\vartheta_\infty) \ge c_1 > 0, \quad n \in \mathcal{N} \tag{3}$$

where $\mathcal{N}$ is unbounded with positive probability. Let $\varepsilon_{m,n} \equiv \sup_{\vartheta \in A_m} |K(\vartheta) - \hat{K}_n(\vartheta)|$. For any fixed m, $\varepsilon_{m,n} \xrightarrow{\text{a.s.}} 0$ by G3. Let

$$m_n = \arg\max\Big\{ m' \le M_n : \forall m \le m', \varepsilon_{m-1,n} + 2\varepsilon_{m,n} < (\alpha' - \alpha)\xi(c_1) \ \ \& \ \ \hat{\vartheta}_{m,n} \in A_m \Big\}.$$

Clearly, $m_n \xrightarrow{\text{p}} \infty$, and for any $m \le m_n$, assuming (3):

$$\begin{aligned}
K(\hat{\vartheta}_{m,n}) &\le \hat{K}_n(\hat{\vartheta}_{m,n}) + \varepsilon_{m,n} \\
&\le \alpha' \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} \hat{K}_n(\vartheta) + (1-\alpha')\hat{K}_n(\hat{\vartheta}_{m-1,n}) + \varepsilon_{m,n} \\
&\le \alpha' \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} K(\vartheta) + (1-\alpha')K(\hat{\vartheta}_{m-1,n}) + \varepsilon_{m-1,n} + 2\varepsilon_{m,n} \\
&= \alpha \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} K(\vartheta) + (1-\alpha)K(\hat{\vartheta}_{m-1,n}) \\
&\quad - (\alpha' - \alpha)\big(K(\hat{\vartheta}_{m-1,n}) - \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} K(\vartheta)\big) + \varepsilon_{m-1,n} + 2\varepsilon_{m,n} \\
&\le \alpha \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} K(\vartheta) + (1-\alpha)K(\hat{\vartheta}_{m-1,n}) \\
&\quad - (\alpha' - \alpha)\xi\Big(K(\hat{\vartheta}_{m,n}) - K(\vartheta_\infty)\Big) + \varepsilon_{m-1,n} + 2\varepsilon_{m,n} \\
&\le \alpha \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} K(\vartheta) + (1-\alpha)K(\hat{\vartheta}_{m-1,n}) \\
&\quad - (\alpha' - \alpha)\xi\Big(c_1\Big) + \varepsilon_{m-1,n} + 2\varepsilon_{m,n} \\
&\le \alpha \inf_{\vartheta \in \Pi_{m-1}\hat{\vartheta}_{m-1}} K(\vartheta) + (1-\alpha)K(\hat{\vartheta}_{m-1,n}) \text{ for all } m \le m_n \ .
\end{aligned}$$

Thus, there is a sequence $\{\overline{\vartheta}_1^{(n)}, \overline{\vartheta}_2^{(n)}, \dots\} \in \mathcal{S}(\vartheta_0, \alpha)$, such that $\overline{\vartheta}_m^{(n)} = \hat{\vartheta}_{m,n}$, $m \le m_n$. Hence, by Assumption G2, $K(\vartheta_{m_n,n}) \le K(\vartheta_\infty) + c_{m_n}$, where $\{c_m\}$ is independent of n, and $c_m \to 0$. Therefore, since $m_n \to \infty$, $K(\hat{\vartheta}_{m_n,n}) \to K(\vartheta_\infty)$, contradicting (3). ■

In fact we have proved that sequences m_n can be chosen in the following way involving K.

Corollary 4 *Let M_n be* any *sequence tending to ∞. Let $\tilde{m}_n = \arg\min\{K(\hat{\vartheta}_{m,n}) : \ 1 \le m \le M_n\}$. Then, under* G1 – G3, $\hat{\vartheta}_{\tilde{m}_n} \xrightarrow{P} \vartheta_\infty$.

To find $\hat{\vartheta}_{\hat{m}_n,n}$ which are totally determined by the data determining $\hat{K}_n$, we need to add some information about the speed of convergence of $\hat{K}_n$ to K on the "sample" iterates. Specifically, suppose we can determine, in advance, $M_n^* \to \infty$, $\varepsilon_n \to 0$ such that,

$$P[\sup\{|\hat{K}_n(\hat{\vartheta}_{m,n}) - K(\hat{\vartheta}_{m,n})| : \ 1 \le m \le M_n^*\} \ge \varepsilon_n] \le \varepsilon_n \ .$$

Then $\hat{m}_n = \arg\min\{\hat{K}_n(\hat{\vartheta}_{m,n}) : 1 \le m \le M_n^*\}$ yields an appropriate $\hat{\vartheta}_{\hat{m}_n}$ sequence. We consider this in Section 4. Before that we return to the application of the result of this section to boosting.

3.2 Back to Boosting

We return to boosting, where we consider $\Theta_m = \{\sum_{j=1}^m \lambda_j h_j : \lambda_j \in \mathbb{R}, h_j \in \mathcal{H}\}$, and therefore $\Pi_m \equiv \Pi$, $\Pi(\vartheta) = \{\vartheta + \lambda h, \lambda \in \mathbb{R}, h \in \mathcal{H}\}$. To simplify notation, for any function $a(X,Y)$, let $P_n a(X,Y) = n^{-1}\sum_{i=1}^n a(X_i,Y_i)$ and $Pa(X,Y) = Ea(X,Y)$. Finally, we identify $\hat{\vartheta}_{m,n} = \sum_{j=1}^m \hat{\lambda}_j \hat{h}_j = \sum_{j=1}^m \hat{\lambda}_{j,n}\hat{h}_{j,n}$.

We assume further

GA1. $W(\cdot)$ is of bounded variation on finite intervals.

GA2. $\mathcal{H}$ has finite L_1 bracketing entropy.

GA3. There are finite $a_1, a_2, \ldots$ such that $\sup_n \sum_{j=1}^m |\hat{\lambda}_{j,n}| \le a_m$ with probability 1.

Theorem 5 *Suppose the conclusion of Theorem 1 and Conditions* GA1–GA3 *are satisfied, then conditions* G2, G3 *are satisfied.*

Proof Condition G2 follows from Theorem 1. It remains to prove the uniform convergence in Condition G3. However, GA2 and GA3 imply that $\mathcal{F} \equiv \{F : F = \sum_{j=1}^m \lambda_j h_j, h_j \in \mathcal{H}, |\lambda_j| \le M\}$ has finite L_1 bracketing entropy. Since W can be written as the difference of two monotone functions $\{W(YF) : F \in \mathcal{F}\}$ inherits this property. The result follows from Bickel and Millar (1991), Proposition 2.1. ■

4. Test Bed Stopping

Again we face the issue of data dependent and in some way optimal selection of $\hat{m}_n$. We claim that this can be achieved over a wide range of possible rates of convergence of $EW\big(\hat{F}_{\hat{m}_n}(YX)\big)$ to $EW\big(F_\infty(YX)\big)$ by using a test bed sample to pick the estimator. The following general result plays a key role.

Let $B = B_n \to \infty$, and let $(X,Y),(X_1,Y_1),\ldots,(X_{n+B},Y_{n+B})$ be i.i.d. P, $X \in \mathcal{X}$, $|Y| \le 1$. Let $\hat{\vartheta}_m : \mathcal{X} \to \mathbb{R}$, $1 \le m \le m_n$ be data dependent functions which depend only on $(X_1,Y_1),\ldots,(X_n,Y_n)$ which are predictors of Y. For $g, g_1, g_2 : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$, given P, define

$$\begin{aligned}
\langle g_1, g_2\rangle_* &\equiv \frac{1}{B_n}\sum_{b=1}^{B_n} g_1(X_{b+n},Y_{b+n})g_2(X_{b+n},Y_{b+n}) \\
\langle g_1, g_2\rangle_P &\equiv P\big(g_1(X,Y)g_2(X,Y)\big) = \int g_1(x,y)g_2(x,y)dP(x,y) \\
\|g\|_*^2 &\equiv \langle g_1, g_2\rangle_* \\
\|g\|_P^2 &\equiv \langle g_1, g_2\rangle_P
\end{aligned}$$

Let,

$$\tau = \arg\min\{\|Y - \hat{\vartheta}_m(X)\|_*^2 : 1 \le m \le M_n\}$$

and $\hat{\vartheta}_\tau$ be the selected predictor. Similarly, let

$$o = \arg\min\{\|Y - \hat{\vartheta}_m(X)\|_P^2 : \ 1 \le m \le M_n\}$$

and $\hat{\vartheta}_o$ be the corresponding predictor.

That is, $\hat{\vartheta}_o(X,Y)$ is the predictor an "oracle" knowing P and (X_i,Y_i), $1 \le i \le n$ would pick from $\hat{\vartheta}_1,\ldots,\hat{\vartheta}_{M_n}$ to minimize squared error loss. Let $\vartheta_o(X) \equiv E_P(Y|X)$, the Bayes predictor. Let $\mathcal{P}$ be a set of probabilities and $r_n \equiv \sup\{E_P\|\hat{\vartheta}_o - \vartheta_o\|_P^2 : \ P \in \mathcal{P}\}$.

The following result is due to Györfi et al. (2002) (Theorem 7.1), although there it is stated in the form of an oracle inequality. We need the following condition:

C. $B_n r_n / \log M_n \to \infty$.

Theorem 6 *(Györfi et al.) Suppose condition* **C** *is satisfied, and* $|Y| \le 1$, $\|\hat{\vartheta}_m\|_\infty \le 1$. *Then,*

$$\sup\{\left|E_P(Y - \hat{\vartheta}_\tau)^2 - E_P(Y - \hat{\vartheta}_o)^2\right| : P \in \mathcal{P}\} = o(r_n).$$

Condition **C** very simply asks that the test sample size B_n be large only: (i) In terms of r_n, the minimax rate of convergence; (ii) In terms of the logarithm of the number of procedures being studied. If $|Y| \le 1$, there is no loss in requiring $\|\hat{\vartheta}_m\|_\infty \le 1$, since we could also replace $\hat{\vartheta}_m$ by its truncation at ± 1, minimizing the L_2 cross validated test set risk. Along similar lines, using $\mathrm{sgn}(\hat{\vartheta}_m)$ is equivalent to cross validating the probability of misclassification for these rules, since if $\hat{\vartheta}_m, Y \in \{-1,1\}$, $E(Y - \hat{\vartheta}_m)^2 = 4P(\hat{\vartheta}_m \ne Y)$.

As we shall see in Section 6, typically $r_n = n^{-1+\delta}$, and M_n is at most polynomial in n. If n/B_n is slowly varying, we can check that the conditions hold. Essentially we can only not deal with r_n of order $n^{-1}\log n$.

5. Algorithmic Speed of Convergence

We consider now the time it takes the sample algorithm to convergence. The fact that the algorithm converges follows from Theorem 1. We show in this section that in fact the algorithm perfectly separates the data (*perfect separation* is achieved when $Y_i F_m(x_i) > 0$ for all $i = 1,\ldots,n$) after no more than $c_1 n^2$ steps. Perfect separation is equivalent to empirical misclassification error 0.

The randomness considered in this section comes only from the Y_i, while the design points are considered fixed. We denote them, therefore, by lower case $x_1,\ldots,x_n$. We consider the following assumptions:

O1. W has regular growth in the sense that $W' < \kappa(W+1)$ for some $\kappa < \infty$. Assume, wlog, that $W(0) = -W'(0) = 1$.

O2. Suppose $x_1,\ldots,x_n$ are all different Then the points can be finitely isolated by $\mathcal{H}$ in the sense that there is k and positive $\alpha_1,\ldots,\alpha_k$ such that for every i there are $h_1,\ldots,h_k \in \mathcal{H}$ such that $\sum_{j=1}^k \alpha_j h_j(x_s) = 1$ if $s = i$, and 0 otherwise. Assume further, as usual, that if $h \in \mathcal{H}$ then $h^2 \equiv 1$ and $-h \in \mathcal{H}$.

Condition O1 is satisfied by all the loss functions mentioned in the introduction. Condition O2 is satisfied, for example by stumps, trees, and any $\mathcal{H}$ whose span includes indicators of small sets with arbitrary location. In particular, if $x_i \in \mathbb{R}$, $x_1 < x_2 < \cdots < x_n$, and $\mathcal{H} = \{\mathrm{sgn}(\cdot - x), x \in \mathbb{R}\}$, we can then take $\alpha_1 = \alpha_2 = 1$, $h_1(\cdot) = \mathrm{sgn}\big(\cdot - (x_{i-1} + x_i)/2\big)$, and $h_2(\cdot) = -\mathrm{sgn}\big(\cdot - (x_i + x_{i+1})/2\big)$

Theorem 7 *Suppose assumptions* O1 *and* O2 *are satisfied and the algorithm starts with* $F_0(0) = 0$. *If* $Y_i F_m(x_i) < 0$ *for at least one* i, *then*

$$\frac{1}{n}\sum_{i=1}^{n} W\big(Y_i F_m(x_i)\big) - \frac{1}{n}\sum_{i=1}^{n} W\big(Y_i F_{m+1}(x_i)\big) \geq \frac{1}{2\kappa\big(n\sum_{j=1}^{k}\alpha_j\big)^2}.$$

Hence, the boosting algorithm perfectly separates the data after at most $2\kappa(n\sum_{j=1}^{k}|\alpha_j|)^2$ *steps.*

Proof Let, for i such that $Y_i F_m(x_i) < 0$,

$$f_m(\lambda; h) = n^{-1}\sum_{s=1}^{n} W\Big(Y_i\big(F_m(x_s) + \lambda h(x_s)\big)\Big),$$

and $f_m'(0;h) = d\, f_m(\lambda;h)/d\lambda\big|_{\lambda=0}$. Consider $h_1,\ldots,h_k$ as in assumption O2. Replace h_j by $-h_j$ if necessary to ensure that $Y_i\sum_{j=1}^{k}\alpha_j h_j(x_s) = \delta_{si}$. Then

$$\begin{aligned}\sum_{j=1}^{k}\alpha_j f_m'(0;h_j) &= n^{-1}\sum_{j=1}^{k}\alpha_j\sum_{s=1}^{n} W'\big(Y_i F_m(x_s)\big)Y_i h_j(x_s)\\ &= n^{-1}W'\big(Y_i F_m(x_i)\big).\end{aligned}$$

Hence

$$\inf_{h\in\mathcal{H}} f_m'(0;h) \leq \frac{1}{n\sum_{j=1}^{k}\alpha_j}\min_i W'(Y_i F_m(x_i)) \leq \frac{W'(0)}{n\sum_{j=1}^{k}\alpha_j} = \frac{-1}{n\sum_{j=1}^{k}\alpha_j}, \tag{4}$$

since $Y_i F_m(x_i) < 0$ for at least one i.

Let $\bar{h}$ be the minimizer of $f_m'(0,h)$. Note that in particular $f_m'(0;\bar{h}) < 0$. The function $f_m(\cdot;\bar{h})$ is convex, hence it is decreasing in some neighborhood of 0. Denote by $\bar{\lambda}$ its minimizer. Consider the Taylor expansion:

$$\begin{aligned}f_m(\bar{\lambda};\bar{h}) &= f_m(0;\bar{h}) + \bar{\lambda} f_m'(0;\bar{h}) + \frac{\bar{\lambda}^2}{2n}\sum_{s=1}^{n} W''\Big(Y_i\big(F_m(x_s) + \widetilde{\lambda}(\lambda)\bar{h}(x_s)\big)\Big)\\ &= f_m(0;\bar{h}) + \inf_{\lambda}\Big\{\lambda f_m'(0;\bar{h}) + \frac{\lambda^2}{2n}\sum_{s=1}^{n} W''\Big(Y_i\big(F_m(x_s) + \widetilde{\lambda}(\lambda)\bar{h}(x_s)\big)\Big)\Big\}\end{aligned}$$

where $\widetilde{\lambda}(\lambda)$ lies between 0 and $\bar{\lambda}$. By condition 01,

$$\begin{aligned}&\inf_{\lambda}\Big\{\lambda f_m'(0;\bar{h}) + \frac{\lambda^2}{2n}\sum_{s=1}^{n} W''\Big(Y_i\big(F_m(x_s) + \widetilde{\lambda}(\lambda)\bar{h}(x_s)\big)\Big)\Big\}\\ &\leq \inf_{\lambda}\{\lambda f_m'(0;\bar{h}) + \frac{\lambda^2\kappa}{4n}\sum_{s=1}^{n} W\Big(Y_i\big(F_m(x_s) + \widetilde{\lambda}(\lambda)\bar{h}(x_s)\big)\Big) + \frac{\lambda^2\kappa}{4}\Big\} \qquad (5)\\ &\leq \inf_{\lambda}\{\lambda f_m'(0;\bar{h}) + \frac{\lambda^2\kappa}{2}\}\end{aligned}$$

because $\frac{1}{n}\sum_{s=1}^n W(Y_i(F_m(x_s)+\widetilde{\lambda}(\lambda)\bar{h}(x_s)) \le \frac{1}{n}\sum_{s=1}^n W(Y_iF_m(x_s)) \le W(0)=1$ since $\bar{\lambda}$ minimizes $f_m(\lambda;\bar{h})$ on $[0,\bar{\lambda}]$, $\widetilde{\lambda}$ is an intermediate point, and $F_0 \equiv 0$. Combining (4) and (5) and the minimizing property of $\bar{h}$,

$$\begin{aligned} f_m(\bar{\lambda};\bar{h}) &\le f_m(0;\bar{h}) - \frac{\left(f_m'(0;\bar{h})\right)^2}{2\kappa} \\ &\le f_m(0;\bar{h}) - \frac{1}{2\kappa(n\sum_{j=1}^k \alpha_j)^2} \quad . \end{aligned}$$

The second statement of the theorem follows because the initial value of $n^{-1}\sum_{i=1}^n W\left(Y_iF_0(x_i)\right)$ is 1, and the value would fall below 0 after at most $m = 2\kappa(n\sum_{j=1}^k \alpha_j)^2$ steps in which at least one observation is not classified correctly. Since the value is necessarily positive, we conclude that all observations would be classified correctly before the mth step.

■

6. Achieving Rates with Sieve Boosting

We propose a regularization of L_2 boosting which we view as being in the spirit of the original proposal, but, unlike it, can be shown for, suitable $\mathcal{H}$, to achieve minimax rates for estimation of $E(Y|X)$ under quadratic loss for $\mathcal{P}$ for which $E(Y|X)$ is assumed to belong to a compact set of functions such as a ball in Besov space if $X \in \mathbb{R}$ or to appropriate such subsets of spaces of smooth functions in $X \in \mathbb{R}^d$—see, for example, the classes $\mathcal{F}$ of Györfi et al. (2003). In fact, they are adaptive in the sense of Donoho et al (1995) for scales of such spaces. We note that Bühlmann and Yu (2003) have introduced a version of L_2 boosting which achieves minimax rates for Sobolev classes on $\mathbb{R}$ adaptively already. However, their construction is in a different spirit than that of most boosting papers. They start out with $\mathcal{H}$ consisting of one extremely smooth and complex function and show that boosting reduces bias (roughness of the function) while necessarily increasing variance. Early stopping is still necessary and they show it can achieve minimax rates.

It follows, using a result of Yang (1999) that our rule is adaptive minimax for classification loss for some of the classes we have mentioned as well. Unfortunately, as pointed out by Tsybakov (2001), the sets $\{x: |F_B(x)| \le \varepsilon\}$ can behave very badly as $\varepsilon \downarrow 0$, no matter how smooth F_B, the misclassification Bayes rule, is, so that these results are not as indicative as we would like them to be. In a recent paper, Bartlett, Jordan, and McAuliffe (2003) considered minimization of the W empirical risk $n^{-1}\sum_{i=1}^n W(Y_iF(X_i))$, for fairly general convex W, over sets of the form $\mathcal{F} = \{F = \sum_{j=1}^m \alpha_j h_j,\ h_j \in \mathcal{H},\ \sum_{j=1}^m |\alpha_j| \le \alpha_n,\ \text{(for some representation of } F)\}$. They obtained oracle inequalities relating $EW(Y\hat{F}(X))$ for $\hat{F}_j$ the empirical minimizer over $\mathcal{F}_j$ to the empirical W risk minimum. They then proceeded to show using conditions related to Tsybakov's (A1) above how to relate the misclassification regret of $\hat{\mathcal{F}}_j$, given by $\langle P[Y\hat{F}_j(X)<0] - P[YF_B(X)<0]\rangle$ to $\langle E_pW(Y\hat{F}_j) - E_pW(YF_B^*)\rangle$, the W regret where F_B^* is the Bayes rule for W. Using these results (Theorems 3 and 10) they were able to establish oracle inequalities for $\hat{F}_j$ under misclassification loss. Manor, Meir, and Zhang (2004) considered the same problem, but focused their analysis mainly on L_2 boosting. They obtained an oracle inequality similar to that of Bartlett et al. regularizing by permitting step sizes which are only a fraction $\beta < 1$ of the step size declared optimal by Gauss-Southwell. They went further by obtaining near minimax results on suitable sets.

We also limit our results to L_2 boosting, although we believe this limitation is primarily due to the lack of minimax theorems for prediction when other losses than L_2 are considered. We use yet a different regularization method in what follows. We show in Theorem 8 our variant of L_2 boosting achieves minimax rates for estimating $E(Y|X)$ in a wide class of situations. Boosting up to a simple data-determined cutoff in each sieve level of a model, and then cross-validating to choose between sieve levels, we can obtain results equivalent to those in which full optimization using penalties are used, such as Theorem 2.1 of Baraud (2000) and results of Baron, Birgé, Massart (1999). Then, in Theorem 9, we show, using inequalities related to ones of Tsybakov (2001), Zhang (2004) and Bartlett et al. (2003), that the rules we propose are also minimax for 0–1 loss in suitable spaces.

6.1 The Rule

Our regularization requires that $\mathcal{H} \equiv \mathcal{H}^{(\infty)} = \overline{\cup_{m\geq 1}\mathcal{H}^{(m)}}$ where $\mathcal{H}^{(m)}$ are finite sets with certain properties. For instance, if $\mathcal{H}$ consists of the stumps in $[0,1]$, $\mathcal{H} = \{F_y(\cdot) : F_y(x) = \mathrm{sgn}(x-y),\ x,y \in [0,1]\}$ we can take $\mathcal{H}^{(m)} = \{F_y(\cdot) : y \text{ a dyadic number of order } k,\ y = \frac{j}{2^k},\ 0 \leq j \leq 2^k\}$. Essentially, we construct a sieve approximating $\mathcal{H}$. Let $\mathcal{F}^{(m)}$ be the linear span of $\mathcal{H}^{(m)}$. Evidently $\mathcal{F} = \overline{\cup_{m\geq 1}\mathcal{F}^{(m)}}$. Let $|\mathcal{H}^{(m)}| \equiv D_m$. Then, $\dim(\mathcal{F}^{(m)}) = D_m$. We now describe our proposed regularization of L_2 boosting.

We use the following notation of Section 4, and begin with a glossary and conditions. Let $(X_1,Y_1),\ldots,(X_n,Y_n),(X,Y)$ i.i.d. with

$$\begin{aligned}
(X,Y) &\sim P << \mu, \quad P \in \mathcal{P}, \quad \mathbf{X} \equiv (X_1,\ldots,X_n),\ \mathbf{Y} \equiv (Y_1,\ldots,Y_n)\ . \\
Y &\in \{-1,1\} \\
\|f\|_\mu^2 &\equiv \int f^2\, d\mu \\
\|f\|_n^2 &\equiv \frac{1}{n}\sum_{i=1}^n f^2(X_i,Y_i) \\
\|f\|_\infty &= \sup_{x,y} |f(x,y)| \\
F_P(X) &\equiv E_P(Y|X) \\
\hat{F}_m(X) &= \arg\min\{\|t(X)-Y\|_n^2 :\ t \in \mathcal{F}^{(m)}\} \\
F_m(X) &= \arg\min\{\|t(X)-Y\|_P^2 :\ t \in \mathcal{F}^{(m)}\} \\
E_{\mathbf{X}} &\equiv \text{ Conditional expectation given } X_1,\ldots,X_n
\end{aligned}$$

Note that we will often suppress $\mathbf{X},\mathbf{Y}$ in $v(\mathbf{X},\mathbf{Y},X,Y)$ and drop subscript to P.

Let $\hat{F}_{m,k}$, the kth iterate in $\mathcal{F}_m$, be defined as follows

$$\begin{aligned}
\hat{F}_{1,0} &\equiv F_0 \\
\hat{F}_{m+1,0} &= \hat{F}_{m,\hat{k}(m)} \\
\hat{F}_{m,k+1} &= \hat{F}_{m,k} + \hat{\lambda}_{m,k}\hat{h}_{m,km}
\end{aligned}$$

where

$$\begin{aligned}
(\hat{\lambda}_{m,k},\hat{h}_{m,k}) &\equiv \underset{\lambda\in\mathbb{R},h\in\mathcal{H}^{(m)}}{\arg\min}\ \{-2\lambda P_n(Y-\hat{F}_{m,k})h + \lambda^2 P_n(h^2)\} \\
\hat{k}(m) &= \text{ First } k \text{ such that } \hat{\lambda}_{m,k}^2 \leq \Delta_{m,n},
\end{aligned}$$

where $\Delta_{m,n}$ are constants. Let

$$\tilde{F}_m = H(\hat{F}_{m,\hat{k}(m)})$$

where

$$H(x) = \begin{cases} x & \text{if } |x| \le 1 \\ \operatorname{sgn}(x) & \text{if } |x| > 1 \end{cases} \tag{6}$$

Note that we have suppressed dependence on n here, indicating it only by the "hats". Let,

$$\hat{m} = \arg\min\{\|Y - \widetilde{F}_m(x)\|_* : \ m \le M_n\}$$

where

$$\|f\|_*^2 = \frac{1}{B}\sum_{i=n+1}^{n+B} f^2(X_i, Y_i), \text{ and we take } B = B_n = \frac{n}{\log n} \ .$$

The rule we propose is: $\hat{\delta} = \operatorname{sgn}(\hat{\hat{F}})$, where

$$\hat{\hat{F}} \equiv H(F_{\hat{m},\hat{k}(\hat{m})}) \ . \tag{7}$$

Note: We show at the end of the Appendix (Proof of Lemma 10) that for wavelet $\mathcal{H}$ we take at most $Cn\log n$ steps total in this algorithm.

6.2 Conditions and Results

We use C as a generic constant throughout, possibly changing from line to line but not depending on m, n, or P. Lemma 6.3 and the condition we give are essentially due to Baraud (2001). Let μ be a sigma finite measure on $\mathcal{H}$ and $\|f\|_\mu$ be the $L_2(\mu)$ norm.

R1. If $\mathcal{H}^{(m)} = \{h_{m,1}, \ldots, h_{m,D_m}\}$ and $f_{m,j} \equiv h_{m,j}/\|h_{m,j}\|_\mu$, then $\{f_{m,j}\}, j \ge 1$ is an orthonormal basis of $\mathcal{F}^{(m)}$ in $L_2(\mu)$ such that:

(i) $\|f_{m,j}\|_\infty \le C_\infty D_m^{\frac{1}{2}}$ for all j, where $\|f\|_\infty = \sup_x |f(x)|$.

(ii) There exists an L such that for all m, j, j', $f_{m,j} f_{m,j'} = 0$ if $|j - j'| \ge L$.

R2. There exists $\varepsilon = \varepsilon(P) > 0$ such that, $\varepsilon \le \frac{dP}{d\mu} \le \varepsilon^{-1}$ for all $P \in \mathcal{P}$.

R3. $\sup_{P\in\mathcal{P}} \|F_P - F_m\|_P^2 \le CD_m^{-\beta}$ for all m, $\beta > 1$.

R4. $M_n \le D_{M_n} \le \frac{n}{(\log n))^p}$ for some $p > 1$.

Condition R1 is needed to conclude that we can bound the behavior of the L_∞ norm on $\mathcal{F}^{(m)}$ by that of the L_2 norm for μ. Condition R2 simply ensures that we can do so for $P \in \mathcal{P}$ as well. The members $f_{m,j}$ of the basis of $\mathcal{F}^{(m)}$ must have compact support. It is well known that if $\mathcal{H}_m$ consists of scaled wavelets (in any dimension) then R1 holds. Clearly, if say μ is Lebesgue measure on an hypercube then to satisfy R2 $\mathcal{P}$ can consist only of densities bounded from above and away from 0. Condition R3 gives the minimum approximation error incurred by using an estimate F based

on $\mathcal{F}^{(m)}$, and thus limits our choice of $\mathcal{H}$. Finally, R4 links the oracle error for these sequences of procedures to the number of candidate procedures.

Let

$$r_n(P) = \inf\{E_P\|\hat{F}_m - F_P\|_P^2 : \ 1 \le m \le M_n\}, \qquad r_n \equiv \sup_{P\in\mathcal{P}} r_n(P).$$

Thus, r_n is the minimax regret for an oracle knowing P but restricted to $\hat{F}_m$. We use the notation $a_n \asymp b_n$ for a shortcut for $a_n = O(b_n)$ and $b_n = O(a_n)$, We have

Theorem 8 *Suppose that $\mathcal{P}$ and $\mathcal{F}$ satisfy* R1–R4 *and that $\mathcal{H}$ is a VC class. If $\Delta_{m,n} = O(D_m/n)$, then,*

$$\sup_{\mathcal{P}} E_P\|\hat{\hat{F}}(X) - F_P(X)\|_P^2 \asymp r_n \ . \tag{8}$$

Thus, $\hat{\hat{F}}$ given by (7) is rate minimax.

Theorem 9 *Suppose the assumptions of Theorem 8 hold and $\mathcal{P}_0 = \mathcal{P} \cap \{P : \ P\big(\ |F_P(X)| \le t\big) \le ct^\alpha\}$, $\alpha \ge 0$. Let $\Delta_n(F,P)$ be the Bayes classification regret for P,*

$$\Delta_n(F,P) \equiv P\big(YF(X) < 0\big) - P\big(YF_P(X) < 0\big) \ . \tag{9}$$

Then,

$$\sup_{\mathcal{P}_0} \Delta_n(\hat{\hat{F}},P) \asymp r_n^{\frac{\alpha+1}{\alpha+2}} \ . \tag{10}$$

The condition $P[|F_P(x)| \le t] \le ct^\alpha$, some $\alpha \ge 0$, t sufficiently small appears in Proposition 1 of Tsybakov (2001) as sufficient for his condition (A1) which is studied by both Bartlett et al. (2003) and Mammen and Tsybakov (1999).

The proof of Theorem 9 uses 2 lemmas of interest which we now state. Their proofs are in the Appendix.

We study the algorithm on $\mathcal{F}_m$. For any positive definite matrix Σ define the condition number $\gamma(\Sigma) \equiv \frac{\lambda_{\max}(\Sigma)}{\lambda_{\min}(\Sigma)}$, where $\lambda_{\max}$, $\lambda_{\min}$ are the largest and smallest eigenvalues of Σ. Let $G_m(P) = \|E_P f_{m,i}(X) f_{m,j}(X)\|$ be the $D_m \times D_m$ Gram matrix of the basis $\{f_{m,1}, \ldots, f_{m,D_m}\}$.

Lemma 10 *Under* R1 *and* R2*,*

a) $\gamma(G_m(P)) \le \varepsilon^{-2}$*, where* ε *is as in* R2*.*

b) Let $G_m(P_n)$ be the empirical Gram matrix $\hat{\gamma}_m \equiv \gamma(G_m(P_n))$. Then, if in addition to R1 *and* R2*, $\mathcal{H}$ is a VC class, $P[\gamma(\hat{G}_m) \ge C_1] \le C_2 \exp\{-C_3 n/L^2 D_m\}$ for all $m \le M_n$ for such that $D_m \le n/(\log n)^p$ for $p > 1$.*

c) If $\mathcal{H}$ is a VC class, $P[\|\hat{F}_{m,\hat{k}(m)} - \hat{F}_m\|_k \le C\frac{D_m}{n}] = 1 - O(\frac{1}{n})$ The C and 0 terms are determed solely by the constants appearing in the R conditions.

Lemma 11 *Suppose* R1*,* R2*, and* R4 *hold. Then,*

$$E_P(\widetilde{F}_m - F_P)^2 \le C\{E_P(F_m - F_P)^2 + \frac{D_m}{n} + E_P(\tilde{F}_m - \hat{F}_m)^2\}.$$

This "oracle inequality" is key for what follows.

Proof of Theorem 9

$$P\big(YF(x)<0\big)=\frac{1}{2}E_P\Big(1(F(X)>0)\big(1-F_P(X)\big)\Big)+\frac{1}{2}E_P\Big(1(F(X)<0)\big(1+F_P(X)\big)\Big).$$

Hence for all $\varepsilon>0$,

$$\begin{aligned}\Delta_n(F,P)&=E_P\Big(1\big(F(X)<0,F_P(X)>0\big)F_P(X)-1\big(F(X)>0,F_P(X)<0\big)F_P(X)\Big)\\&=E_P\Big(|F_P(X)|1(F_P(X)F(X)<0)\Big)\\&\le E_P\Big(\big|F(X)-F_P(X)\big|1(F_PF(X)<0,|F_P(X)|>\varepsilon)\Big)+\varepsilon P\big(|F_P(X)|\le\varepsilon\big)\\&\le\frac{1}{\varepsilon}E_P\big(F(X)-F_P(X)\big)^2+c\varepsilon^{\alpha+1}\end{aligned}$$

by assumption. The theorem follows. ■

6.3 Discussion

1) If $X\in\mathbb{R}$ and $\mathcal{H}^{(m)}$ consists of stumps with the discontinuity at a dyadic rational $j/2^m$, then $\mathcal{F}^{(m)}$ is the linear space of Haar wavelets of order m. This is also true if $\mathcal{H}_m$ is the space of differences of two such dyadic stumps. More generally, if $\mathcal{H}$ consists of suitably scaled wavelets, so that $|h|\le 1$, based on the dyadic rationals of order m, them $\mathcal{F}^{(m)}$ is the linear space spanned by the first 2^m elements of the wavelet series. A slight extension of results of Baraud (2001) yields that if we run the algorithm to the limit $k=\infty$ for each m rather than stopping as we indicate, the resulting $\hat{F}_m$ obey the oracle inequality of Lemma 11 with $\Delta_{m,n}=0$.

   Suppose that $X\in\mathbb{R}$ and F_∞ ranges over a ball in an approximation space such as Sobolev or, more generally, Besov. Then, if $\mathcal{F}^{(m)}$ has the appropriate approximation properties, e.g., wavelets as smooth as the functions in the specified space, it follows from Baraud (2001) that we can use penalties not dependent on the data to pick $\hat{F}_{\hat{m}}$ such that,

$$\begin{aligned}\max_{\hat{F}}E_P\Big(\hat{F}_{\hat{m}}(X)-E_P(Y|X)\Big)^2&\asymp\min_{\hat{F}}\max\Big\{E_P\big(\hat{F}(X)-E_P(Y|X)\big)^2:E_P(Y|X)\in\mathcal{F}\Big\}\\&\asymp n^{-1+\varepsilon}\Omega(n)\end{aligned}$$

   where $\Omega(n)$ is slowly varying and $0<\varepsilon<1$. Here $\hat{F}$ ranges over all estimators based only on the data and not on P. The same type of result has been established for more specialized models with $X\in\mathbb{R}^d$ by Baron, Birgé, Massart (1999), and others, see Györfi et al. (2003).

   The resulting minimax risk,

$$\min_{\hat{F}}\max\{E_P\big(\hat{F}(X)-E_P(Y|X)\big)^2:\ E_P(Y|X)\in\mathcal{F}\}$$

   is always of order $n^{-1+\varepsilon}\Omega(n)$ where $\Omega(n)$ is typically constant and $0<\varepsilon<1$.

   What we show in Theorem 8 is that if, rather than optimizing all the way for each m, we stop in a natural fashion and cross validate as we have indicated, then we can achieve the optimal order as well.

2) "Stumps" unfortunately do not satisfy condition R1 with μ Lebesgue measure. Their Gram matrices are too close to being singular. But differences of stumps work.

3) It follows from the results of Yang (1999) that the rate of Theorem 9 for $\alpha = 0$, that is, if $\mathcal{P}_0 = \mathcal{P}$, is best possible for Sobolev balls and the other spaces we have mentioned.

   Tsybakov implicitly defines a class of F_P for which he is able to specify classification minimax rates. Specifically let $X \in [0,1]^d$ and let $b(x_1,\ldots,x_{d-1})$ be a function having continuous partial derivatives up to order ℓ. Let $p_{b,x}(\cdot)$ be the Taylor polynomial or order ℓ obtained from expanding b at x. Then, he defines $\Sigma(l,L)$ to be the class of all such b for which, $|b(y) - p_{b,x}(y)| \le L|y-x|^\ell$ for all $x,y \in [0,1]^{d-1}$. Evidently if b has bounded partial derivatives of order $\ell+1$, $b \in \Sigma(\ell,L)$, for some L. Now let

$$\begin{aligned}\mathcal{P}_\ell = \quad & \{P : F_P(x) = x_d - b(x_1,\ldots,x_{d-1}),\\ & P[|F_P(x)| \le t] \le Ct, \text{ for all } 0 \le t \le 1, b \in \Sigma(\ell,L)\}\end{aligned}$$

   Tsybakov following Mammen and Tsybakov (1999) shows that the classification minimax regret for $\mathcal{P}$ (Theorem 2 of Tsybakov (2001) for $K = 2$) is $\frac{2\ell}{3\ell+(d-1)}$. On the other hand, if we assume that $Y = F_P(x) + \varepsilon$ where ε is independent of X, bounded and $E(\varepsilon) = 0$, then the L_2 minimax regret rate is $2\ell/(2\ell+(d-1))$ – see Birgé and Massart (1999) Sections 4.1.1 and Theorem 9. Our theorem 9 now yields a classification minimax regret rate of

$$\frac{2}{3} \cdot \frac{2\ell}{2\ell+(d-1)} = \frac{2\ell}{3\ell+\frac{3}{2}(d-1)}$$

   which is slightly worse than what can be achieved using Tsybakov's not as readily computable procedures. However, note that as $\ell \to \infty$ so that F_P and the boundary become arbitrarily smooth, L_2 boosting approaches the best possible rate for $\mathcal{P}_\ell$ of $\frac{2}{3}$. Similar remarks can be made about $0 < \alpha \le 1$.

7. Conclusions

In this paper we presented different mathematical aspects of boosting. We consider the observations as an i.i.d. sample from a population (i.e., a distribution). The boosting algorithm is a Gauss-Southwell minimization of a classification loss function (which typically dominates the 0-1 misclassification loss). We show that the output of the boosting algorithm follows the theoretical path as if it were applied to the true distribution of the population. Since early stopping is possible as argued, the algorithm, supplied with an appropriate stopping rule, is consistent.

However, there are no simple rate results other than those of Bühlmann and Yu (2003), which we discuss, for the convergence of the boosting classifier to the Bayes classifier. We showed that rate results can be obtained when the boosting algorithm is modified to a cautious version, in which at each step the boosting is done only over a small set of permitted directions.

Acknowledgments

We would like to acknowledge support for this project from the National Science Foundation (NSF grant DMS-0104075), and the Israel Science Foundation (ISF grant 793/03).

Appendix A. Proof of Theorem 1:

Let $w_0 = \inf_{f\in\mathcal{F}_\infty} w(f)$. Let $f_k^* = \sum_m \alpha_{km} h_{km}$, $h_{k,m} \in \mathcal{H}$, $\sum_m |\alpha_{km}| < \infty$, $k = 0,1,2,\ldots$ be any member of $\mathcal{F}_\infty$ such that (i) $f_0^* = f_0$; (ii) $w(f_k^*) \searrow w_0$ is strictly decreasing sequence; (iii) The following condition is satisfied:

$$w(f_k^*) \geq \alpha w_0 + (1-\alpha) w(f_{k-1}^*) + (1-\alpha)(\nu_{k-1} - \nu_k), \tag{11}$$

where $\nu_k \searrow 0$ is a strictly decreasing real sequence. The construction of the sequence $\{f_k^*\}$ is possible since, by assumption, $\mathcal{F}_\infty$ is dense in the image of $w(\cdot)$. That is, we can start with the sequence $\{w(f_k^*)\}$, and then look for suitable $\{f_k^*\}$. Here is a possible construction. Let c and η be suitable small number. Let $\gamma = (1-\alpha)(1+2\eta)/(1-\eta)$, $\nu_k = c\eta\gamma^k/(1-\gamma)$. Select now f_k^* such $w_0 + c(1-\eta)\gamma^k \leq w(f_k^*) \leq w_0 + c(1+\eta)\gamma^k$. ($\eta$ should be small enough such that $\gamma < 1$ and c should selected such that $w(f_1^*) < w(f_0)$.) Our argument rests on the following,

Lemma 12 *There is a sequence $m_k \to \infty$ such that $w(f_m) \leq w(f_k^*) + \nu_k$ for $m \geq m_k$, $k = 1,2,\ldots$, and $m_k \leq \zeta_k(m_{k-1}) < \infty$, where $\zeta_k(\cdot)$ is a monotone non-decreasing functions which depends only on the sequences $\{\nu_k\}$ and $\{f_k^*\}$.*

Proof of Lemma 12:

We will use the following notation. For $f \in \mathcal{F}_\infty$ let $\|f\|_* = \inf\{\sum|\gamma_i|,\ f = \sum\gamma_i h_i,\ h_i \in \mathcal{H}\}$.

Recall that by definition $w(f_0) = w(f_0^*)$. Our argument proceeds as follows, We will inductively define m_k satisfying the conclusion of the lemma, and make, if $\varepsilon_{k,m} \equiv w(f_m) - w(f_k^*)$,

$$\varepsilon_{k,m} \leq c_{k,m} \equiv \max\Big\{\nu_k,\ \frac{\sqrt{512}B}{\alpha^2\beta_k} \frac{w(f_{k-1}^*) - w_0}{\Big(\log\Big(1 + \frac{8(w(f_{k-1}^*)-w_0)}{\alpha\beta_k(\tau_k+\rho_k m_{k-1})}(m - m_{k-1} + 1)\Big)\Big)^{1/2}}\Big\}, \tag{12}$$

where

$$\begin{aligned} \beta_k &= \inf\{w''(f;h) : w_0 + \nu_k \leq w(f) \leq w(f_0),\ h \in \mathcal{H}\} \\ B &= \sup\{w''(f;h) : w(f) \leq w(f_o),\ h \in \mathcal{H}\} < \infty. \end{aligned} \tag{13}$$

and

$$\begin{aligned} \tau_k &= 2\|f_0 - f_k^*\|_*^2 \\ \rho_k &= \frac{16}{\alpha\beta_k}\big(w(f_0) - w_0\big). \end{aligned} \tag{14}$$

Having defined m_k we establish (12) as part of our induction hypothesis for $m_{k-1} < m \leq m_k$. We begin by choosing $m = m_1 = 1$ so that (12) holds for $m = M - 1 = 1$. We do do this by choosing $\nu_0 > 0$, sufficiently small. Having established the induction for $m \leq m_{k-1}$ we define m_k as follows. Write now the RHS of (12) as $g(m_{k-1})$, where

$$g(\nu) \equiv \max\Big\{\nu_k,\ \frac{\sqrt{512}B}{\alpha^2\beta_k} \frac{w(f_{k-1}^*) - w_0}{\Big(\log\Big(1 + \frac{8(w(f_{k-1}^*)-w_0)}{\alpha\beta_k(\tau_k+\rho_k \nu}(m - \nu + 1)\Big)\Big)^{1/2}}\Big\},$$

We can now pick $\zeta_k(\nu) \equiv \max\{\nu+1, \min\{m:\ g(\nu) \le \nu_k\}\}$, and define $m_k = \zeta_k(\nu_{k-1})$.

Note that $\{\beta_k\}$, $\{\tau_k\}$, $\{\rho_k\}$, and B depend only the sequences $\{f_k^*\}$ and $\{\nu_k\}$. We now proceed to establish (12). for $m_{k-1} < m \le m_k$. Note first that since $\varepsilon_{k,m}$ as a function of m is non-increasing, (12) holds trivially for $m' > m$ if $\varepsilon_{k,m} \le 0$. By induction (12) holds for $m \le m_{k-1}$, and my hold for some $m > mk-1$. Recall that the definition of the algorithm relates the actual gain at the mth to the maximal gain achieved in this step given the previous steps, see its definition (1). Suppose

$$\inf_\lambda w(f_m + \lambda h_m) \le w_0 + \nu_k. \tag{15}$$

Then

$$\begin{aligned}
w(f_{m+1}) &\le \alpha \inf_\lambda w(f_m + \lambda h_m) + (1-\alpha) w(f_m), \quad \text{by (1)} \\
&\le \alpha(w_0 + \nu_k) + (1-\alpha) w(f_m), \quad \text{by (15)} \\
&\le \alpha(w_0 + \nu_k) + (1-\alpha)\big(w(f_{k-1}^*) + \nu_{k-1}\big), \quad \text{by the outer induction, since } m \ge m_{k-1} \\
&\le \alpha(w_0 + \nu_k) + \big(w(f_k^*) - \alpha w_0 + (1-\alpha)\nu_k\big), \quad \text{by (11)} \\
&= w(f_k^*) + \nu_k,
\end{aligned}$$

so that $\varepsilon_{k,m+1} \le \nu_k$. Therefore, m_k' is not larger than $m+1$, that is $\varepsilon_{k,m'} \le \nu_k$ for $m' > m$ then (12) holds trivially for $m' > m$, and hence, by the second induction assumption for all m. We have established (12) save for m such that,

$$\inf_\lambda w(f_m + \lambda h_m) > w_0 + \nu_k \text{ and } \varepsilon_{k,m} \ge 0. \tag{16}$$

We now deal with this case.

Note first that by convexity,

$$|w'(f_m; f_m - f_k^*)| \ge w(f_m) - w(f_k^*) \equiv \varepsilon_{k,m}. \tag{17}$$

We obtain from (17) and the linearity of the derivative that, if $f_m - f_k^* = \sum \gamma_i \widetilde{h}_i \in \mathcal{F}_\infty$,

$$\varepsilon_{k,m} \le \left| \sum -\gamma_i w'(f_m; \tilde{h}_i) \right| \le \sup_{h \in \mathcal{H}} |w'(f_m; h)| \sum |\gamma_i| \ .$$

Hence

$$\sup_{h \in \mathcal{H}} |w'(f_m; h)| \ge \frac{\varepsilon_{k,m}}{\| f_m - f_k^* \|_*}. \tag{18}$$

Now, if $f_{m+1} = f_m + \lambda_m h_m$ then,

$$w(f_m + \lambda_m h_m) = w(f_m) + \lambda_m w'(f_m; h_m) + \frac{1}{2}\lambda_m^2 w''(\tilde{f}_m; h_m), \quad \lambda \in [0, \lambda_m]. \tag{19}$$

where $\tilde{f}_m = f_m + \tilde{\lambda}_m h_m$ and $0 \le \widetilde{\lambda}_m \le \lambda_m$. By convexity, for $0 \le \lambda \le \lambda_m$,

$$w(f_m + \lambda h_m) = w(f_m(1 - \frac{\lambda}{\lambda_m}) + \frac{\lambda}{\lambda_m} f_{m+1}) \le \max\{w(f_m), w(f_{m+1})\} = w(f_m) \le w(f_1).$$

We obtain from Assumption GS1 that $w''(\tilde{f}_m;h) \in (\beta_k, B)$ given in (13). But then we conclude from (19) that,

$$\begin{aligned} w(f_m+\lambda_m h_m) &\geq w(f_m) + \inf_{\lambda\in\mathbb{R}} \left(\lambda w'(f_m;h_m) + \frac{1}{2}\lambda^2\beta_k\right) \\ &= w(f_m) - \frac{|w'(f_m;h_m)|^2}{2\beta_k} \quad . \end{aligned} \tag{20}$$

Note that $w(f_m+\lambda h) = w(f_m)+\lambda w'(f_m,h)+\lambda^2 w''(f_m+\lambda' h,h)/2$ for some $\lambda' \in [0,\lambda]$, and if $w(f_m+\lambda h)$ is close to $\inf_{\lambda,h} w(f_m+\lambda,h)$ then by convexity, $w(f_m+\lambda' h) \leq w(f_m) \leq w(f_0)$. We obtain from the upper bound on w'' we obtain:

$$\begin{aligned} w(f_m+\lambda_m h_m) &\leq \alpha \inf_{\lambda\in\mathbb{R},\, h\in\mathcal{H}} w(f_m+\lambda h) + (1-\alpha)w(f_m), \qquad \text{by definition,} \\ &\leq \alpha \inf_{\lambda\in\mathbb{R},\, h\in\mathcal{H}} (w(f_m)+\lambda w'(f_m;h) + \frac{1}{2}\lambda^2 B) + (1-\alpha)w(f_m) \\ &= w(f_m) - \frac{\alpha \sup_{h\in\mathcal{H}} |w'(f_m;h)|^2}{2B}, \end{aligned} \tag{21}$$

by minimizing over λ. Hence combining (20) and (21) we obtain,

$$|w'(f_m;h_m)| \geq \alpha \sup_{h\in\mathcal{H}} |w'(f_m;h)| \sqrt{\frac{\beta_k}{B}} \tag{22}$$

By (21) for the LHS and convexity for the RHS:

$$\frac{\alpha \sup_{h\in\mathcal{H}} |w'(f_m;h)|^2}{2B} \leq w(f_m) - w(f_{m+1}) \leq -\lambda_m w'(f_m;h_m)$$

Hence

$$|\lambda_m| \geq \frac{\alpha \sup_{h\in\mathcal{H}} |w'(f_m;h)|}{2B}.$$

Applying (18) we obtain:

$$|\lambda_m| \geq \frac{\alpha}{2B}\frac{\varepsilon_{k,m}}{l_{k,m}}, \tag{23}$$

where $l_{k.m} \equiv \| f_m - f_k^* \|_*$.

Let λ_m^0 be the minimal point of $w(f_m+\lambda h_m)$. Taylor expansion around that point and using the lower bound on the curvature:

$$w(f_m+\lambda h_m) \geq w(f_m+\lambda_m^0 h_m) + \frac{1}{2}\beta_k(\lambda-\lambda_m^0)^2 \tag{24}$$

Hence

$$\begin{aligned} {\lambda_m^0}^2 &\leq \frac{2}{\beta_k}\left(w(f_m) - w(f_m+\lambda_m^0 h_m)\right) \\ &\leq \frac{2}{\alpha\beta_k}\left(w(f_m) - w(f_{m+1})\right), \end{aligned} \tag{25}$$

where the RHS follows (1). Similarly

$$\begin{aligned}(\lambda_m-\lambda_m^0)^2 &\le \frac{2}{\beta_k}\big(w(f_{m+1})-w(f_m+\lambda_m^0 h_m)\big)\\ &\le \frac{2(1-\alpha)}{\alpha\beta_k}\big(w(f_m)-w(f_{m+1})\big)\end{aligned} \tag{26}$$

Combining (25) and (26):

$$\lambda_m^2 \le \frac{8}{\alpha\beta_k}\big(w(f_m)-w(f_{m+1})\big). \tag{27}$$

Since $\varepsilon_{k,m}\ge 0$ by assumption (16), we conclude from (27) that,

$$\sum_{i=m_{k-1}}^{m}\lambda_i^2 \le \frac{8}{\alpha\beta_k}(w(f_{k-1}^*)-w_0). \tag{28}$$

However, by definition,

$$\begin{aligned}l_{k,m+1} &\le l_{k,m}+|\lambda_m|\\ &\le l_k+\sum_{i=m_{k-1}}^{m}|\lambda_i|\\ &\le l_k+(m+1-m_{k-1})^{1/2}\Big(\sum_{i=m_{k-1}}^{m}\lambda_i^2\Big)^{1/2}\end{aligned} \tag{29}$$

by Cauchy-Schwarz, where, similarly,

$$\begin{aligned}l_k = l_{k,m_{k-1}} &= \|f_{m_{k-1}}-f_k^*\|_*\\ &\le \|f_0-f_k^*\|_*+\|f_{m_{k-1}}-f_0\|_*\\ &\le \|f_0-f_k^*\|_*+\sum_{m=0}^{m_{k-1}-1}|\lambda_m|\\ &\le \|f_0-f_k^*\|_*+m_{k-1}^{1/2}\sqrt{\sum_{m=0}^{m_{k-1}-1}\lambda_m^2}\\ &\le \|f_0-f_k^*\|_*+\sqrt{\frac{8m_{k-1}}{\alpha\beta_k}}\sqrt{w(f_0)-w(f_{m_{k-1}})}, \quad \text{by (27)}\\ &\le \|f_0-f_k^*\|_*+\sqrt{\frac{8m_{k-1}}{\alpha\beta_k}}\sqrt{w(f_0)-w_0}\\ &\le \sqrt{\tau_k+\rho_k m_{k-1}}, \quad \text{as defined in (14).}\end{aligned} \tag{30}$$

Together, (23), (28), and (29) yield:

$$\begin{aligned}\frac{8}{\alpha\beta_k}(w(f^*_{k-1})-w_0) &\geq \sum_{i=m_{k-1}}^{m}\lambda_i^2 \\ &\geq \frac{\alpha^2}{4B^2}\sum_{i=m_{k-1}}^{m}\frac{\varepsilon_{k,i}^2}{l_{k,i}^2} \\ &\geq \frac{\alpha^2}{4B^2}\sum_{i=m_{k-1}}^{m}\frac{\varepsilon_{k,i}^2}{(l_k+(8(w(f^*_{k-1})-w_0)/\alpha\beta_k)^{1/2}(i-m_{k-1})^{1/2})^2}\end{aligned} \tag{31}$$

Further, since $\varepsilon_{k,m}$ are decreasing by construction and positive by assumption (16), we can simplify the sum on the RHS of (31):

$$\begin{aligned}&\sum_{i=m_{k-1}}^{m}\frac{\varepsilon_{k,i}^2}{(l_k+(8(w(f^*_{k-1})-w_0)/\alpha\beta_k)^{1/2}(i-m_{k-1})^{1/2})^2} \\ &\qquad\geq \frac{\varepsilon_{k,m}^2}{2}\sum_{i=0}^{m-m_{k-1}}\frac{1}{l_k^2+8i(w(f^*_{k-1})-w_0)/\alpha\beta_k}.\end{aligned} \tag{32}$$

Using the inequality,

$$\sum_{i=0}^{m-m_{k-1}}\frac{1}{a+bi}\geq\int_0^{m-m_{k-1}+1}\frac{1}{a+bt}\,dt=\frac{1}{b}\log\Big(1+\frac{b}{a}(m-m_{k-1}+1)\Big)$$

on the RHS of (32), we obtain from (31) and (32) that (12) holds, for the case (16). This establishes (16) for all k and m.

■

Proof of Theorem 1: Since the lemma established the existence of monotone ζ_k's, it followed from the definition of these function that $w(f_m)\leq w(f^*_{k(m)})$ where $k(m)=\sup\{k:\zeta^{(k)}(f_0^*)\leq m\}$ and $\zeta^{(k)}=\zeta_k\circ\cdots\circ\zeta_1$ is the kth iterate of the ζs. Since $\zeta^{(k)}(f_0^*)<\infty$ for all k, we have established the uniform rate of convergence and can define the sequence $\{c_m\}$, where $c_m=w(f^*_{k(m)})-w_0$.

We now prove the uniform step improvement claim of the theorem and identify a suitable function $\xi(\cdot)$. From (26) and (23) if $\varepsilon_{k,m}\geq 0$

$$w(f_m)-w(f_{m+1})\geq\frac{\alpha\beta_k}{2}\lambda_m^2\geq\frac{\alpha\beta_k}{2}\left(\frac{\alpha}{2B}\frac{\varepsilon_{k,m}}{l_{k,m}}\right)^2, \tag{33}$$

Bound $l_{k,m}$ similarly to (30) by

$$l_{k,m}\leq l_{k,1}+m^{1/2}\Big(\sum_{i=1}^{m}\lambda_i^2\Big)^2\leq l_{k,1}+\sqrt{\frac{8m}{\alpha\beta_k}(w(f_0)-w_0)}. \tag{34}$$

Let $m^*(v)=\inf\{m':c_{m'}\leq v-w_0\}$, which is well defined since $c_m\to 0$. Thus, any realization of the algorithm will cross the v line on or before step number $m^*(v)$. In particular, $m\leq m^*\big(w(f_m)\big)$ for

any m and any realization of the algorithm. We obtain therefore by plugging-in (34) in (33), using the m^* as a bound on m and the identity $(a+b)^2 \leq 2a^2+2b^2$ that:

$$w(f_m) - w(f_{m+1}) \geq \frac{\alpha^3\beta_k}{16B^2} \frac{w(f_m) - w(f_k^*)}{l_{k,1}^2 + 8m^*\big(w(f_m)\big)\big(w(f_0) - w_o\big)/\alpha\beta_k},$$

as long as $\varepsilon_{k,m} \geq 0$. Taking the maximum of the RHS over the permitted range, yields a candidate for the ξ function:

$$\xi(w) \equiv \sup_{k:\, w(f_k^*) \leq w} \Big\{ \frac{\alpha^3\beta_k}{16B^2} \frac{w - w(f_k^*)}{l_{k,1}^2 + m^*\big(w\big)\big(w(f_0) - w_o\big)/\alpha\beta_k} \Big\}.$$

This proves the theorem under GS1. Under GS2, the only inequality which we need to replace is (20) since now $\beta_k = 0$ is possible. However the definition of Algorithm 2 ensures that we have a coefficient of at least γ on λ^2 in (20). The theorem is proved.

■

Appendix B. Proof of Lemmas 10 and 11 and Theorem 8

Proof of Lemma 10 Since by (R2)

$$\begin{aligned}
\lambda_{\max}(G_m(P)) &= \sup_{\|x\|=1} x' G_m(P) x \\
&= \sup_{\|x\|=1} \sum\sum x_i x_j \int f_{m,i} f_{m,d} dP \\
&= \sup_{\|x\|=1} \int (\sum x_i f_{m,i})^2 dP \qquad (35) \\
&\leq \varepsilon^{-1} \sup_{\|x\|=1} \int (\sum x_i f_{m,i})^2 d\mu = \varepsilon^{-1} \\
\lambda_{\max}(G_m(P)) &\geq \varepsilon, \quad \text{similarly.}
\end{aligned}$$

Part a) follows.

For any symmetric matrix M define its operator norm $\|\cdot\|_T$ by $\lambda_{\max}(M)$. For simplicity let $G_m = G_m(P)$ and $\hat{G}_m = G_m(P_n)$. Recall that for any symmetric matrices A and and M:

$$\begin{aligned}
|\lambda_{\max}(A) - \lambda_{\max}(M)| &\leq \|A - M\|_T \\
|\lambda_{\min}(A) - \lambda_{\min}(M)| &\leq \|A - M\|_T.
\end{aligned}$$

Now,

$$\begin{aligned}
&P\left[\left| \frac{\lambda_{\max}\big(\hat{G}_m\big)}{\lambda_{\min}\big(\hat{G}_m\big)} - \frac{\lambda_{\max}\big(G_m\big)}{\lambda_{\min}\big(G_m\big)} \right| \geq t \right) \qquad (36) \\
&\qquad \leq P\Big(\|\hat{G}_m - G_m\|_T > \frac{\varepsilon}{2} \Big) + P\Big(\|\hat{G}_m - G_m\|_T \geq t/(\frac{1}{\varepsilon} + \frac{2}{\varepsilon^3}) \Big)
\end{aligned}$$

Recall that for a banded matrix M of with band of width $2L$,

$$\begin{aligned}\|M\|_T^2 &= \sup_{\|x\|=1} \|Mx\|^2 \\ &= \sup_{\|x\|=1} \sum_a \Big(\sum_b M_{ab}x_b\Big)^2 \\ &\le \sup_{\|x\|=1} \sum_a \sum_{|b-a|<L} x_b^2 M_\infty^2 \\ &\le 2LM_\infty^2 \sup_{\|x\|=1} \sum_a x_a^2 = 2LM_\infty^2,\end{aligned}$$

where $\|M\|_\infty \equiv \max_{a,b}|M_{ab}|$. Since both $\hat{G}_m$ and $G_m(P)$ are banded of width d, say,

$$\|\hat{G}_m - G_m\|_T \le 2L \max\Big\{\Big|\frac{1}{n}\sum_{i=1}^n \big(f_{m,a}f_{m,b}\big)(X_i) - E_P f_{m,a}f_{m,b}(X_i)\big)\Big| : \ |a-b|<L\Big\}. \tag{37}$$

If $\mathcal{H}$ is a VC class, we can conclude from (35)–(37) that,

$$P[\gamma(\hat{G}_m) \ge C_1] \le C_2 \exp\{-C_3 n/L^2 D_m\} \tag{38}$$

since by R1 (i), $\|f_m\|_\infty \le C_\infty D_m^{\frac{1}{2}}$. The constants ε, C_1, C_2 and C_3 depend on the constants of the R conditions only. This is a consequence of Theorem 2.14.16 p. 246 of van der Vaart and Wellner (1996). This complete the proof of part b).

By a standard result for the Gauss-Southwell method, Luenberger (1984), page 229:

$$\|\hat{F}_{m,k+1} - \hat{F}_m\|_n^2 \le \Big(1 - \frac{1}{\hat{\gamma}_m D_m}\Big)\|\hat{F}_{m,k} - \hat{F}_m\|_n^2 \tag{39}$$

Hence

$$\|\hat{F}_{m,k} - \hat{F}_m\|_n^2 - \|\hat{F}_{m,k+1} - \hat{F}_m\|_n^2 \ge \frac{1}{\hat{\gamma}_m D_m}\|\hat{F}_{m,k} - \hat{F}_m\|_n^2$$

Thus, if

$$\frac{1}{n} \ge \|\hat{F}_{m,k} - \hat{F}_m\|_n^2 - \|\hat{F}_{m,k+1} - \hat{F}_m\|_n^2$$

we obtain

$$\|\hat{F}_{m,k} - \hat{F}_m\|_n^2 \le D_m \hat{\gamma}_m / n. \tag{40}$$

¿From (40) part (c) follows. ■

Note: Since

$$\|\hat{F}_{m,k-1} - \hat{F}_m\|_n^2 - \|\hat{F}_{m,k} - \hat{F}_m\|_n^2 \ \ge \ \frac{C}{n}$$

(39) implies that

$$\Big(1 - \frac{1}{\hat{\gamma}_m D_m}\Big)^{\hat{k}(m)} \ \ge \ \frac{1}{n}.$$

Therefore:

$$\hat{k}(m) \le \log n \hat{\gamma}_m D_m \ .$$

If, for instance, as with wavelets $D_m = 2^m, m \le \log_2 n$ we take at most $Cn\log n$ steps total.

Lemma 13 *:*
If $E_{\mathbf{x}}$ denotes conditional expectation give n $X_1,\ldots,X_n$, under R1 *and $F \equiv F_p$,*

$$E_{\mathbf{x}}\|\hat{F}_m - F_m\|_n^2 \le C(\frac{D_m}{n} + \|F_m - F\|_P^2) \tag{41}$$

This is a standard type of result – see Barron, Birgé, Massart (1999). We include the proof for completeness. Note that,

$$\|\hat{F}_m(X) - Y\|_n^2 = \frac{1}{n}\mathbf{Y}^T(I-P)\mathbf{Y}$$

where $\mathbf{Y} \equiv (Y_1,\ldots,Y_n)^T$ and P is the projection matrix of dimension D_m onto the L space spanned by $\big(h_j(X_1),\ldots,h_j(X_n)\big),\ 1 \le j \le D_m$. Then, $(I-P)v = 0$ for all $v \in L$. Hence,

$$E_{\mathbf{X}}\|\hat{F}_m(X) - Y\|_n^2 = \frac{1}{n}E_{\mathbf{X}}\big(\mathbf{Y} - \mathbf{F}_m(\mathbf{X})\big)^T(I-P)\big(\mathbf{Y} - \mathbf{F}_m(\mathbf{X})\big)$$

where $\mathbf{F}_m(\mathbf{X}) = \big(F_m(X_1),\ldots,F_m(X_n)\big)^T$ is the projection of $(F(X_1),\ldots,F(X_n))^T$ onto L. Note also that,

$$\|\hat{F}_m - F_m\|_n^2 = \|\mathbf{Y} - \mathbf{F}_m(\mathbf{X})\|_n^2 - \|\mathbf{Y} - \hat{\mathbf{F}}_m(\mathbf{X})\|_n^2$$

where $\hat{\mathbf{F}}(X) = (\hat{F}_m(X_1),\ldots,\hat{F}_m(X_n))^T$. Hence,

$$\begin{aligned}
E_{\mathbf{X}}\|\hat{F}_m - F_m\|_n^2 &= \tfrac{1}{n}E_{\mathbf{X}}(\mathbf{Y} - \mathbf{F}_m(X))^T P(\mathbf{Y} - \mathbf{F}_m(\mathbf{X})) \\
&= \tfrac{1}{n}E_{\mathbf{X}}(\mathbf{Y} - \mathbf{F}(\mathbf{X}))^T P(\mathbf{Y} - \mathbf{F}(\mathbf{X})) + \tfrac{2}{n}E_{\mathbf{X}}(\mathbf{F}_m - \mathbf{F})^T P(\mathbf{Y} - \mathbf{F}_m(\mathbf{X})) \\
&= \tfrac{1}{n}E_{\mathbf{X}}\,\mathrm{trace}[P(\mathbf{Y} - \mathbf{F}(\mathbf{X}))(\mathbf{Y} - \mathbf{F}(\mathbf{X}))] \\
&\quad + \tfrac{2}{n}E_{\mathbf{X}}(\mathbf{F}_m - \mathbf{F})^T P(\mathbf{F}_m - \mathbf{F})(\mathbf{X})
\end{aligned}$$

But

$$E_{\mathbf{X}}\,\mathrm{trace}[P(\mathbf{Y} - \mathbf{F}(\mathbf{X})))(\mathbf{Y} - \mathbf{F}(\mathbf{X}))^T] = \frac{1}{n}\sum_{i=1}^n Var(Y_i|X_i)p_{ii}(X) \le \max_i Var(Y_i|X_i)\frac{D_m}{n}$$

since

$$\sum_{i=1}^n p_{ii}(X) = \mathrm{trace}\,P = D_m$$

Also, since P is a projection matrix

$$(\mathbf{F}_m - \mathbf{F})^T P(\mathbf{F}_m - \mathbf{F})(\mathbf{X}) \le \|F_m - F\|_n^2$$

and (41) *follows.*

Proof of Lemma 11:
Take $\Delta_{m,n} = 0$. Let $\tilde{\rho}_m = \sup\left\{\frac{\|t(X)\|_P}{\|t(X)\|_n} \ :\ t \in \mathcal{F}_m\right\}$. By Proposition 5.2 of Baraud (2001), if $\rho_0 > h_0^{-1}$,

$$P[\tilde{\rho}_m > \rho_0] \le D_m^2 \exp\{-\frac{(h_0 - \rho_0^{-1})^2}{4h_1}\, c_n \log n\}$$

where $c_n = \frac{n}{CD_m \log n}$. Here $h_0, h_1 C$ are generic constants. Baraud gives a proof for the case $Var(Y|X) =$ constant, but this is immaterial since only functions of $\underset{\sim}{X}$ are involved in $\tilde{\rho}_m$. Therefore,

$$\begin{aligned} & E_P(\hat{F}_m - F_P)^2 1(\rho_m \le \rho_0) \\ \le \; & 2\rho_0^2 E_P\{E_n(\hat{F}_m - F_m)^2 + E_n(F_m - F_P)^2\} \\ \le \; & C(\frac{D_m}{n} + \|F_m - F_P\|^2) \end{aligned} \tag{42}$$

On the other hand,

$$\begin{aligned} E_P(\hat{F}_m - F_P)^2 1(\rho_m > \rho_0) &\le 2P[\rho_m > \rho_0] \\ &= CD_m^2 \exp\{-AC_n \log n\} \end{aligned} \tag{43}$$

Combining (42) and (43) we obtain Lemma 11 for $\Delta_{m,n} = 0$, $\hat{F}_m = \widetilde{F}_m$. Putting in $\widetilde{F}_m$ we add a term $CE_P(\hat{F}_m - \widetilde{F}_m)^2$. We now apply Lemma 10 c) and the argument we used to obtain (42) and (43). ■

Proof of Theorem 8: Note that we are limited to rates of convergence which are slower than $n^{-\frac{1}{2}}$. This comes from the combination of R1(i) and bounding the operator by the l_∞ norm of the Gram matrix. It is not clear how either of these conditions can be relaxed.

We need only check that if the $\{\widetilde{F}_m\}$ are the θ_m of Theorem 6 then the conditions of that theorem are satisfied. By construction, $\|\widetilde{F}_m\|_\infty \le 1$, $B_n = \frac{n}{\log n}$. By Lemma 11 and (R3),

$$r_n \le C_1 \frac{D_m}{n} + C_2 D_m^{-B} \tag{44}$$

and the right hand side of (44) is bounded by $n^{-(\frac{\beta}{\beta+1})}$. ■

References

P. K. Andersen and R. D. Gill. Cox's regression model for counting processes: A large sample study. *Ann. Stat.* 10:1100–1120, 1982.

Y. Baraud. Model selection for regression on a random design.*Tech. Report*, U. Paris Sud, 2001.

A. Barron, L. Birgé, and P. Massart. Risk bounds for model selection under penalization. *Prob. Theory and Related Fields*, 113:301–413, 1999.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Tech. Report* 638, Department of Statistics, University of California at Berkeley, 2003.

P. J. Bickel and P. W. Millar. Uniform convergence of probability measures on classes of functions. Statistica Sinica 2:1-15, 1992.

P. J. Bickel and Y. Ritov. The golden chain. A comment. *Ann. Statist.*, 32:91–96, 2003.

L. Breiman. Arcing classifiers (with discussion). *Ann. Statist.* 26:801–849, 1998.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation* 11:1493-1517, 1999.

L. Breiman. Some infinity theory for predictor ensembles *Technical Report* U.C. Berkeley, 2000.

P. Bühlmann. Consistency for L_2 boosting and matching pursuit with trees and tree type base functions. *Technical Report* ETH Zürich, 2002.

P. Bühlmann and B. Yu. Boosting the L_2 loss: regression and classification. *J. of Amer. Statist. Assoc.*, 98:324–339, 2003

D. Donoho, I.M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: asymptopia (with discussion). *J. Roy. Statist. Soc. Ser. B* 57:371–394, 1995.

J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *Ann. Statist.* 28:337–407, 2000.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation* 121:256–285, 1995.

Y, Freund and R. E. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proc. 13th International Conference*, 148–156. Morgan Kauffman, San Francisco, 1996.

G. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution Free Theory of Nonparametric Regression*. Springer, New York, 2002.

W. Jiang. Process consistency for ADABOOST. Technical Report 00-05, Dept. of Statistics, Northwestern University, 2002.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data . *J. of Amer. Statist. Assoc.*, 99:67–81, 2002.

D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Reading, 1984.

G. Lugosi and N. Vayatis. On the Bayes-risk consistency of boosting methods. *Ann. Statist.* 32:30–55, 2004.

S. Mallat and Z. Zhang. Matching pursuit with time frequency dictionaries. *IEEE Transactions on Signal Processing* 41:3397–3415, 1993.

E. Mammen and A. Tsybakov. Smooth discrimination analysis. *Ann. Statist.* 27:1808–1829, 1999.

S. Mannor, R. Meir, and T. Zhang. Greedy algorithms for classification—consistency, convergence rates and adaptivity. J. of Machine Learning Research 4:713–742, 2004.

L. Mason, P. Bartlett, J. Baxter, and M. Frean. Functional gradient techniques for combining hypotheses. In Schölkopg, Smola, A., Bartlett, P., and Schurmans, D. (edts.) *Advances in Large Margin Classifiers*, MIT Press, Boston, 2000.

R. E. Schapire. The strength of weak learnability. *Machine Learning* 5:197–227, 1990.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence related predictions. *Machine Learning*, 37:297–336, 1999.

A, Tsybakov. Optimal aggregation of classifiers in statistical learning. *Technical Report*, U. of Paris IV, 2001.

A. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer, New York, 1996.

Y. Yang. Minimax nonparametric classification – Part I Rates of convergence, Part II Model selection, *IEEE Trans. Inf. Theory* 45:2271–2292, 1999.

T. Zhang and B. Yu. Boosting with early stopping: convergence and consistency. Tech Report 635, Stat Dept, UCB, 2003.

T. Zhang. Statistical behaviour and consistency of classification methods based on convex risk minimization. *Ann. Statist.*, 32:56–134, 2004.

Journal of Machine Learning Research 7 (2006) 733–769 Submitted 12/05; Revised 4/06; Published 5/06

QP Algorithms with Guaranteed Accuracy and Run Time for Support Vector Machines

Don Hush DHUSH@LANL.GOV
Patrick Kelly KELLY@LANL.GOV
Clint Scovel JCS@LANL.GOV
Ingo Steinwart INGO@LANL.GOV
Modeling, Algorithms and Informatics Group, CCS-3, MS B265
Los Alamos National Laboratory
Los Alamos, NM 87545 USA

Editor: Bernhard Schölkopf

Abstract

We describe polynomial–time algorithms that produce approximate solutions with guaranteed accuracy for a class of QP problems that are used in the design of support vector machine classifiers. These algorithms employ a two–stage process where the first stage produces an approximate solution to a dual QP problem and the second stage maps this approximate dual solution to an approximate primal solution. For the second stage we describe an $O(n\log n)$ algorithm that maps an approximate dual solution with accuracy $(2\sqrt{2K_n}+8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$ to an approximate primal solution with accuracy ε_p where n is the number of data samples, K_n is the maximum kernel value over the data and $\lambda > 0$ is the SVM regularization parameter. For the first stage we present new results for *decomposition* algorithms and describe new decomposition algorithms with guaranteed accuracy and run time. In particular, for τ–*rate certifying* decomposition algorithms we establish the optimality of $\tau = 1/(n-1)$. In addition we extend the recent $\tau = 1/(n-1)$ algorithm of Simon (2004) to form two new *composite* algorithms that also achieve the $\tau = 1/(n-1)$ iteration bound of List and Simon (2005), but yield faster run times in practice. We also exploit the τ–rate certifying property of these algorithms to produce new stopping rules that are computationally efficient and that guarantee a specified accuracy for the approximate dual solution. Furthermore, for the dual QP problem corresponding to the standard classification problem we describe operational conditions for which the Simon and composite algorithms possess an upper bound of $O(n)$ on the number of iterations. For this same problem we also describe general conditions for which a matching lower bound exists for *any* decomposition algorithm that uses working sets of size 2. For the Simon and composite algorithms we also establish an $O(n^2)$ bound on the overall run time for the first stage. Combining the first and second stages gives an overall run time of $O(n^2(c_k+1))$ where c_k is an upper bound on the computation to perform a kernel evaluation. Pseudocode is presented for a complete algorithm that inputs an accuracy ε_p and produces an approximate solution that satisfies this accuracy in low order polynomial time. Experiments are included to illustrate the new stopping rules and to compare the Simon and composite decomposition algorithms.

Keywords: quadratic programming, decomposition algorithms, approximation algorithms, support vector machines

1. Introduction

Solving a quadratic programming (QP) problem is a major component of the support vector machine (SVM) training process. In practice it is common to employ algorithms that produce *approximate* solutions. This introduces a trade-off between computation and accuracy that has not been thoroughly explored. The accuracy, as measured by the difference between the criterion value of the approximate solution and the optimal criterion value, is important for learning because it has a direct influence on the generalization error. For example, since the optimal criterion value plays a key role in establishing the SVM performance bounds in (Steinwart and Scovel, 2004, 2005; Scovel et al., 2005b) the influence of the accuracy can be seen directly through the proofs of these bounds. Since the primal QP problem can be prohibitively large and its Wolfe dual QP problem is considerably smaller it is common to employ a two–stage training process where the first stage produces an approximate solution to the dual QP problem and the second stage maps this approximate dual solution to an approximate primal solution. Existing algorithms for the first stage often allow the user to trade accuracy and computation for the dual QP problem through the choice of a tolerance value that determines when to stop the algorithm, but it is not known how to choose this value to achieve a desired accuracy or run time. Furthermore existing algorithms for the second stage have been developed largely without concern for accuracy and therefore little is known about the accuracy of the approximate primal solutions they produce. In this paper we describe algorithms that accept the accuracy ε_p of the primal QP problem as an input and are guaranteed to produce an approximate solution that satisfies this accuracy in low order polynomial time. To our knowledge these are the first algorithms of this type for SVMs. In addition our run time analysis reveals the effect of the accuracy on the run time, thereby allowing the user to make an informed decision regarding the trade–off between computation and accuracy.

Algorithmic strategies for the dual QP problem must address the fact that when the number of data samples n is large the storage requirements for the kernel matrix can be excessive. This barrier can be overcome by invoking algorithmic strategies that solve a large QP problem by solving a sequence of smaller QP problems where each of the smaller QP problems is obtained by fixing a subset of the variables and optimizing with respect to the remaining variables. Algorithmic strategies that solve a QP problem in this way are called *decomposition* algorithms and a number have been developed for dual QP problems: (Balcazar et al., 2001; Chen et al., 2005, 2006; Cristianini and Shawe-Taylor, 2000; Hsu and Lin, 2002; Hush and Scovel, 2003; Joachims, 1998; Keerthi et al., 2000, 2001; Laskov, 2002; Liao et al., 2002; List and Simon, 2004, 2005; Mangasarian and Musicant, 1999, 2001; Osuna et al., 1997; Platt, 1998; Simon, 2004; Vapnik, 1998).

The key to developing a successful decomposition algorithm is in the method used to determine the *working sets*, which are the subsets of variables to be optimized at each iteration. To guarantee stepwise improvement each working set must contain a *certifying pair* (Definition 3 below). Stronger conditions are required to guarantee convergence: (Chang et al., 2000; Chen et al., 2006; Hush and Scovel, 2003; Lin, 2001a,b; List and Simon, 2004) and even stronger conditions appear necessary to guarantee rates of convergence: (Balcazar et al., 2001; Hush and Scovel, 2003; Lin, 2001a). Indeed, although numerous decomposition algorithms have been proposed few are known to possess polynomial run time bounds. Empirical studies have estimated the run time of some common decomposition algorithms to be proportional to n^p where p varies from approximately 1.7 to approximately 3.0 depending on the problem instance: (Joachims, 1998; Laskov, 2002; Platt, 1998). Although these types of studies can provide useful insights they have limited utility in pre-

dicting the run time for future problem instances. In addition these particular studies do not appear to be calibrated with respect to the accuracy of the final criterion value and so their relevance to the framework considered here is not clear. Lin (2001a) performs a convergence rate analysis that may eventually be used to establish run time bounds for a popular decomposition algorithm, but these results hold under rather restrictive assumptions and more work is needed before the tightness and utility of these bounds is known (a more recent version of this analysis can be found in (Chen et al., 2006)). Balcazar et al. (2001) present a randomized decomposition algorithm whose expected run time is $O\big((n+r(\mathtt{k}^2d^2))\ \mathtt{k}d\log n\big)$ where n is the number of samples, d is the dimension of the input space, $1 \leq \mathtt{k} \leq n$ is a data dependent parameter and $r(\mathtt{k}^2d^2)$ is the run time required to solve the dual QP problem over $\mathtt{k}^2d^2$ samples. This algorithm is very attractive when $\mathtt{k}^2d^2 \ll n$, but in practice the value of $\mathtt{k}$ is unknown and it may be large when the Bayes error is not close to zero. Hush and Scovel (2003) define a class of *rate certifying algorithms* and describe an example algorithm that uses $O\left(\frac{K_n n^5 \log n}{\varepsilon}\right)$ computation to reach an approximate dual solution with accuracy ε, where K_n is the maximum value of the kernel matrix. Recently Simon (2004) introduced a new rate certifying algorithm which can be shown, using the results in (List and Simon, 2005), to use $O\left(\frac{nK_n}{\lambda\varepsilon} + n^2 \log\left(\frac{\lambda n}{K_n}\right)\right)$ computation to reach an approximate dual solution with accuracy ε, where $\lambda > 0$ is the SVM regularization parameter. In this paper we combine Simon's algorithm with the popular *Generalized SMO* algorithm of Keerthi et al. (2001) to obtain a *composite* algorithm that possesses the same computation bound as Simon's algorithm, but appears to use far less computation in practice (as illustrated in our experiments). We also extend this approach to form a second *composite* algorithm with similar properties. In addition we introduce operational assumptions on K_n and the choice of λ and ε that yield a simpler computation bound of $O(n^2)$ for these algorithms. Finally to guarantee that actual implementations of these algorithms produce approximate solutions with accuracy ε we introduce two new stopping rules that terminate the algorithms when an adaptively computed upper bound on the accuracy falls below ε.

The second stage of the design process maps an approximate dual solution to an approximate primal solution. In particular this stage determines how the approximate dual solution is used to form the normal vector and offset parameter for the SVM classifier. It is common practice to use the approximate dual solution as coefficients in the linear expansion of the data that forms the normal vector, and then use a heuristic based on approximate satisfaction of the Karush-Kuhn-Tucker (KKT) optimality conditions to choose the offset parameter. This approach is simple and computationally efficient, but it produces an approximate primal solution whose accuracy is unknown. In this paper we take a different approach based on the work of Hush et al. (2005). This work studies the accuracy of the approximate primal solution as a function of the accuracy of the approximate dual solution and the map from approximate dual to approximate primal. In particular for the SVM problem it appears that choosing this map involves a trade–off between computation and accuracy. Here we employ a map described and analyzed in (Hush et al., 2005) that guarantees an accuracy of ε_p for the primal QP problem when the dual QP problem is solved with accuracy $(2\sqrt{2K_n} + 8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$. This map resembles current practice in that it performs a direct substitution of the approximate dual solution into a linear expansion for the normal vector, but differs in the way that it determines the offset parameter. We develop an $O(n \log n)$ algorithm that computes the offset parameter according to this map.

The main results of this paper are presented in Sections 2 and 3. Proofs for all the theorems, lemmas, and corollaries in these sections can be found in Section 6, except for Theorem 2 which is

established in (Hush et al., 2005). Section 2 describes the SVM formulation, presents algorithms for the first and second stages, and provides theorems that characterize the accuracy and run time for these algorithms. Section 3 then determines specific run time bounds for decomposition algorithms applied to the standard classification problem and the density level detection problem. Section 4 describes experiments that illustrate the new stopping rules and compare the run time of different decomposition algorithms. Section 5 provides a summary of results and establishes an overall run time bound. A complete algorithm that computes an ε_p–optimal solution to the primal QP problem is provided by (Procedure 1, Section 2) and Procedures 3–8 in the appendix.

2. Definitions, Algorithms, and Main Theorems

Let X be a pattern space and $k : X \times X \to \mathbb{R}$ be a kernel function with Hilbert space H and feature map $\phi : X \to H$ so that $k(x_1,x_2) = \phi(x_1) \cdot \phi(x_2), \forall x_1,x_2 \in X$. Define $Y := \{-1,1\}$. Given a data set $((x_1,y_1),...,(x_n,y_n)) \in (X \times Y)^n$ the *primal* QP problem that we consider takes the form

$$\begin{array}{ll} \min_{\psi,b,\xi} & \lambda\|\psi\|^2 + \sum_{i=1}^n u_i\xi_i \\ \text{s.t.} & y_i(\phi(x_i)\cdot\psi + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \;\; i = 1,2,...,n \end{array} \tag{1}$$

where $\lambda > 0$, $u_i > 0$ and $\sum_i u_i = 1$. This form allows a different weight u_i for each data sample. Specific cases of interest include:

1. the *L1–SVM* for the standard supervised classification problem which sets $u_i = 1/n$, $i = 1,...,n$,
2. the *DLD–SVM* for the density level detection problem described in (Steinwart et al., 2005) which sets
$$u_i = \begin{cases} \frac{1}{(1+\rho)n_1}, & y_i = 1 \\ \frac{\rho}{(1+\rho)n_{-1}}, & y_i = -1 \end{cases}$$
where n_1 is the number of samples distributed according to P_1 and assigned label $y = 1$, n_{-1} is the number of samples distributed according to P_{-1} and assigned label $y = -1$, $h = dP_1/dP_{-1}$ is the density function, and $\rho > 0$ defines the ρ–level set $\{h > \rho\}$ that we want to detect.

The *dual* QP problem is

$$\begin{array}{ll} \max_a & -\frac{1}{2}a\cdot Qa + a\cdot 1 \\ \text{s.t.} & y\cdot a = 0 \\ & 0 \leq a_i \leq u_i \;\; i = 1,2,...,n. \end{array} \tag{2}$$

where

$$Q_{ij} = y_i y_j k(x_i,x_j)/2\lambda.$$

The change of variables defined by

$$\alpha_i := y_i a_i + l_i, \quad l_i = \begin{cases} 0 & y_i = 1 \\ u_i & y_i = -1 \end{cases} \tag{3}$$

gives the *canonical dual* QP problem

$$\begin{array}{ll} \max_\alpha & -\frac{1}{2}\alpha\cdot Q\alpha + \alpha\cdot w + w_0 \\ \text{s.t.} & 1\cdot\alpha = c \\ & 0 \leq \alpha_i \leq u_i \;\; i = 1,2,...,n \end{array} \tag{4}$$

where

$$Q_{ij} = k(x_i, x_j)/2\lambda, \;\; c = l \cdot 1, \;\; w = Ql + y, \;\; w_0 = -l \cdot y - \frac{1}{2} l \cdot Ql. \tag{5}$$

We denote the canonical dual criterion by

$$R(\alpha) := -\frac{1}{2}\alpha \cdot Q\alpha + \alpha \cdot w + w_0.$$

Note that this change of variables preserves the criterion value. Also note that the relation between a and α is one–to–one. Most of our work is with the canonical dual because it simplifies the algorithms and their analysis.

We define the set of ε–optimal solutions of a constrained optimization problem as follows.

Definition 1 *Let P be a constrained optimization problem with parameter space Θ, criterion function $G : \Theta \to \mathbb{R}$, feasible set $\tilde{\Theta} \subseteq \Theta$ of parameter values that satisfy the constraints, and optimal criterion value G^* (i.e. $G^* = \sup_{\theta \in \tilde{\Theta}} G(\theta)$ for a maximization problem and $G^* = \inf_{\theta \in \tilde{\Theta}} G(\theta)$ for a minimization problem). Then for any $\varepsilon \geq 0$ we define*

$$o_\varepsilon(P) := \{\theta \in \tilde{\Theta} : |G(\theta) - G^*| \leq \varepsilon\}$$

to be the set of ε–optimal solutions for P.

We express upper and lower computation bounds using $O(\cdot)$ and $\Omega(\cdot)$ notations defined by

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\},$$
$$\Omega(g(n)) = \{f(n) : \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}.$$

We now describe our algorithm for the primal QP problem. It computes an approximate canonical dual solution $\hat{\alpha}$ and then maps to an approximate primal solution $(\hat{\psi}, \hat{b}, \hat{\xi})$ using the map described in the following theorem. This theorem is derived from (Hush et al., 2005, Theorem 2 and Corollary 1) which is proved using the result in (Scovel et al., 2005a).

Theorem 2 *Consider the primal QP problem P_{SVM} in (1) with $\lambda > 0$ and $|\phi(x_i)|^2 \leq K, i = 1, .., n$, and its corresponding canonical dual QP problem D_{SVM} in (4) with criterion R. Let $\varepsilon_p > 0$, $\varepsilon = (2\sqrt{2K} + 8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$ and suppose that $\hat{\alpha} \in o_\varepsilon(D_{SVM})$ and $R(\hat{\alpha}) \geq 0$. If*

$$\hat{\psi} = \frac{1}{2\lambda} \sum_{i=1}^{n} (\hat{\alpha}_i - l_i)\phi(x_i)$$

$$\hat{\xi}_i(b) = \max(0, 1 - y_i(\hat{\psi} \cdot \phi(x_i) + b)), \; i = 1, .., n$$

and

$$\hat{b} \in \arg\min \sum_{i=1}^{n} u_i \hat{\xi}_i(b)$$

then $(\hat{\psi}, \hat{b}, \hat{\xi}(\hat{b})) \in o_{\varepsilon_p}(P_{SVM})$.

This theorem gives an expression for $\hat{\psi}$ that coincides with the standard practice of replacing an optimal dual solution α^* by an approximate dual solution $\hat{\alpha}$ in the expansion for the optimal normal vector determined by the KKT conditions. The remaining variables $\hat{\xi}$ and $\hat{b}$ are obtained by substituting $\hat{\psi}$ into the primal optimization problem, optimizing with respect to the slack variable ξ, and then minimizing with respect to b [1]. To guarantee an accuracy ε_p for the primal problem this theorem stipulates that the value of the dual criterion at the approximate solution be non–negative and that the accuracy for the dual solution satisfy $\varepsilon = (2\sqrt{2K} + 8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$. The first condition is easily achieved by algorithms that start with $\alpha = l$ (so that the initial criterion value is 0) and continually improve the criterion value at each iteration. We will guarantee the second condition by employing an appropriate stopping rule for the decomposition algorithm.

Procedure 1 shows the primal QP algorithm that produces an ε_p–optimal solution $(\hat{\alpha}, \hat{b})$ that defines the SVM classifier

$$\operatorname{sign}\left(\sum_{i=1}^{n}\left(\frac{\hat{\alpha}_i - l}{2\lambda}\right)k(x_i, x) + \hat{b}\right).$$

This algorithm inputs a data set $T_n = ((x_1, y_1), \ldots, (x_n, y_n))$, a kernel function k, and parameter values λ, u and ε_p. Lines 3–6 produce an exact solution for the degenerate case where all the data samples have the same label. The rest of the routine forms an instance of the canonical dual QP according to (5), sets ε according to Theorem 2, sets $\alpha^0 = l$ so that $R(\alpha^0) = 0$, uses the routine `Decomposition` to compute an ε–approximate canonical dual solution $\hat{\alpha}$, and uses the routine `Offset` to compute the offset parameter $\hat{b}$ according to Theorem 2. The parameter g, which is defined in the next section, is a temporary value computed by `Decomposition` that allows a more efficient computation of $\hat{b}$ by `Offset`. The next three sections provide algorithms and computational bounds for the routines `Decomposition` and `Offset`.

Procedure 1 The algorithm for the primal QP problem.

1: `PrimalQP` $(T_n, k, \lambda, u, \varepsilon_p)$
2:
3: **if** $(y_i = y_1, \forall i)$ **then**
4: $\quad \hat{\alpha} \leftarrow l$, $\hat{b} \leftarrow y_1$
5: $\quad$ Return$(\hat{\alpha}, \hat{b})$
6: **end if**
7: Form canonical dual: $Q_{ij} \leftarrow \frac{k(x_i,x_j)}{2\lambda}$, $l_i \leftarrow \frac{(1-y_i)u_i}{2}$, $w \leftarrow Ql + y$, $c \leftarrow l \cdot 1$
8: Compute Desired Accuracy of Dual: $\varepsilon \leftarrow \frac{\lambda\varepsilon_p^2}{(2\sqrt{2K}+8\sqrt{\lambda})^2}$
9: Initialize canonical dual variable: $\alpha^0 \leftarrow l$
10: $(\hat{\alpha}, g) \leftarrow$ `Decomposition`$(Q, w, c, u, \varepsilon, \alpha^0)$
11: $\hat{b} \leftarrow$ `Offset`(g, y, u)
12: Return$(\hat{\alpha}, \hat{b})$

2.1 Decomposition Algorithms

We begin with some background material that describes: optimality conditions for the canonical dual, a model decomposition algorithm, necessary and sufficient conditions for convergence to a

1. This method for choosing the offset was investigated briefly in (Keerthi et al., 2001, Section 4).

solution, and sufficient conditions for rates of convergence. In many cases this background material extends a well known result to the slightly more general case considered here where each component of u may have a different value.

Consider an instance of the canonical dual QP problem given by (Q, w, w_0, c, u). Define the set of feasible values

$$\mathcal{A} := \{\alpha : (0 \leq \alpha_i \leq u_i) \text{ and } (\alpha \cdot 1 = c)\},$$

and the set of optimal solutions

$$\mathcal{A}^* := \arg\max_{\alpha \in \mathcal{A}} R(\alpha).$$

Also define the optimal criterion value $R^* := \sup_{\alpha \in \mathcal{A}} R(\alpha)$ and the gradient at α

$$g(\alpha) := \nabla R(\alpha) = -Q\alpha + w. \tag{6}$$

The optimality conditions established by Keerthi et al. (2001) take the form,

$$\alpha \in \mathcal{A}^* \quad \Leftrightarrow \quad g_j(\alpha) \leq g_k(\alpha) \text{ for all } j : \alpha_j < u_j,\ k : \alpha_k > 0. \tag{7}$$

These conditions motivate the following definition from (Keerthi et al., 2001; Hush and Scovel, 2003).

Definition 3 *A* certifying pair *(also called a* violating pair*) for $\alpha \in \mathcal{A}$ is a pair of indices that witness the non–optimality of α, i.e. it is a pair of indices $j : \alpha_j < u_j$ and $k : \alpha_k > 0$ such that $g_j(\alpha) > g_k(\alpha)$.*

Using the approach in (Hush and Scovel, 2003, Section 3) it can be shown that the requirement that working sets contain a certifying pair is both necessary and sufficient to obtain a stepwise improvement in the criterion value. Thus, since certifying pairs are defined in terms of the gradient component values it appears that the gradient plays an essential role in determining members of the working sets. To compute the gradient at each iteration using (6) requires $O(n^2)$ operations. However since decomposition algorithms compute a sequence of feasible points $(\alpha^m)_{m \geq 0}$ using working sets of size p, the sparsity of $(\alpha^{m+1} - \alpha^m)$ means that the update

$$g(\alpha^{m+1}) = g(\alpha^m) - Q(\alpha^{m+1} - \alpha^m) \tag{8}$$

requires only $O(pn)$ operations. A model decomposition algorithm that uses this update is shown in Procedure 2. After computing an initial gradient vector this algorithm iterates the process of determining a working set, solving a QP problem restricted to this working set, updating the gradient vector, and testing a stopping condition.

The requirement that working sets contain a certifying pair is necessary but not sufficient to guarantee convergence to a solution (e.g. see the examples in Chen et al., 2006; Keerthi and Ong, 2000). However Lin (2002b) has shown that including a *max–violating pair* defined by

$$(j^*, k^*) \ : \ j^* \in \arg\max_{i : \alpha_i < u_i} g_i(\alpha), \quad k^* \in \arg\min_{i : \alpha_i > 0} g_i(\alpha) \tag{9}$$

in each working set does guarantee convergence to a solution. Once the gradient has been computed a max–violating pair can be determined in one pass through the gradient components and therefore requires $O(n)$ computation. The class of *max–violating pair algorithms* that include a max–violating

Procedure 2 A model decomposition algorithm for the canonical dual QP problem.

ModelDecomposition$(Q, w, c, u, \varepsilon, \alpha^0)$
Compute initial gradient $g^0 \leftarrow -Q\alpha^0 + w$
$m \leftarrow 0$
repeat
  Compute a working set W^m
  Compute α^{m+1} by solving the restricted QP determined by α^m and W^m
  Update the gradient: $g^{m+1} \leftarrow g^m - Q(\alpha^{m+1} - \alpha^m)$
  $m \leftarrow m + 1$
until (stopping condition is satisfied)
Return(α^m, g^m)

pair in each working set includes many popular algorithms (e.g. Chang and Lin, 2001; Joachims, 1998; Keerthi et al., 2001). Although asymptotic convergence to a solution is guaranteed for these algorithms, their convergence rate is unknown. In contrast we now describe algorithms based on alternative pair selection strategies that have the same $O(n)$ computational requirements (once the gradient has been computed) but possess known rates of convergence to a solution.

Consider the model decomposition algorithm in Procedure 2. The run time of the main loop is the product of the number of iterations and the computation per iteration, and both of these depend heavily on the size of the working sets and how they are chosen. The smallest size that admits a convergent algorithm is 2 and many popular algorithms adopt this size. We refer to these as *W2* decomposition algorithms. A potential disadvantage of this approach is that the number of iterations may be larger than it would be otherwise. On the other hand adopting working sets of size 2 allows us to solve each 2–variable QP problem in constant time (e.g. see Platt, 1998). In addition *W2* decomposition algorithms require only $O(n)$ computation to update the gradient and have the advantage that the overall algorithm can be quite simple (as demonstrated by the *W2* max–violating pair algorithm). Furthermore adopting size 2 working sets will allow us to implement our new stopping rules in constant time. Thus, while most of the algorithms we describe below allow the working sets to be larger than 2, our experiments will be performed with their *W2* variants.

In addition to their size, the content of the working sets has a significant impact on the run time through its influence on the convergence rate of the algorithm. Hush and Scovel (2003) prove that convergence rates can be guaranteed simply by including a *rate certifying pair* in each working set. Roughly speaking a *rate certifying pair* is a certifying pair that, when used as the working set, provides a sufficient stepwise improvement. To be more precise we start with the following definitions. Define a working set to be a subset of the index set of the components of α, and let W denote a working set of unspecified size and W_p denote a working set of size p. In particular $W_n = \{1, 2, \ldots, n\}$ denotes the entire index set. The set of feasible solutions for the canonical dual QP sub–problem defined by a feasible value α and a working set W is defined

$$\mathcal{A}(\alpha, W) := \{\acute{\alpha} \in \mathcal{A} : \acute{\alpha}_i = \alpha_i \; \forall i \notin W\}.$$

Define

$$\sigma(\alpha|W) := \sup_{\acute{\alpha} \in \mathcal{A}(\alpha, W)} g(\alpha) \cdot (\acute{\alpha} - \alpha)$$

to be the optimal value of the linear programming (LP) problem at α. The following definition is adapted from (Hush and Scovel, 2003).

Definition 4 *For $\tau > 0$ an index pair W_2 is called a* τ–rate certifying pair *for α if $\sigma(\alpha|W_2) \geq \tau\sigma(\alpha|W_n)$. A decomposition algorithm that includes a τ–rate certifying pair in the working set at every iteration is called a* τ–rate certifying algorithm.

For a τ–rate certifying algorithm Hush and Scovel (2003) provide an upper bound on the number of iterations as a function of τ. An improved bound can be obtained as a special case of (List and Simon, 2005, Theorem 1). The next theorem provides a slightly different bound that does not depend on the size of the working sets and therefore slightly improves the bound obtained from (List and Simon, 2005, Theorem 1) when the size of the working sets is larger than 2.

Theorem 5 *Consider the canonical dual QP problem in (4) with criterion function R and Gram matrix Q. Let $L \geq \max_i Q_{ii}$ and $S \geq \max_i u_i$. A τ–rate certifying algorithm that starts with α^0 achieves $R^* - R(\alpha^m) \leq \varepsilon$ after $\lceil \acute{m} \rceil$ iterations of the main loop where*

$$\acute{m} = \begin{cases} \left[\frac{2}{\tau}\ln\left(\frac{R^* - R(\alpha^0)}{\varepsilon}\right)\right]_+, & \varepsilon \geq \frac{4LS^2}{\tau} \\ \frac{2}{\tau}\left(\frac{4LS^2}{\tau\varepsilon} - 1 + \left[\ln\left(\frac{\tau(R^* - R(\alpha^0))}{4LS^2}\right)\right]_+\right), & \varepsilon < \frac{4LS^2}{\tau} \end{cases},$$

$\lceil\theta\rceil$ denotes the smallest integer greater than or equal to θ, and $[\theta]_+ = \max(0,\theta)$.

Chang et al. (2000) have shown that for every $\alpha \in \mathcal{A}$ there exists a τ–rate certifying pair with $\tau \geq 1/n^2$. This result can be used to establish the existence of decomposition algorithms with polynomial run times. The first such algorithm was provided by Hush and Scovel (2003) where the rate certifying pairs satisfied $\tau \geq 1/n^2$. However the value τ can be improved and the bound on the number of iterations reduced if the rate certifying pairs are determined differently. Indeed List and Simon (2005) prove that $\tau \geq 1/n$ for a *max–lp2* pair

$$W_2^* \in \arg\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2)$$

which is a pair with the maximum linear program value. The next theorem provides a slightly better result of $\tau \geq 1/(n-1)$ for this pair and establishes the optimality of this bound [2].

Theorem 6 *For $\alpha \in \mathcal{A}$*

$$\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2) \geq \frac{\sigma(\alpha|W_n)}{n-1}.$$

Furthermore, there exist problem instances for which there exist $\alpha \in \mathcal{A}$ such that

$$\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2) = \frac{\sigma(\alpha|W_n)}{n-1}.$$

2. This result provides a negligible improvement over the $\tau \geq 1/n$ result of List and Simon but is included here because it establishes optimality and because its proof, which is quite different from that of List and Simon, provides additional insight into the construction of certifying pairs that achieve $\tau \geq 1/(n-1)$.

Since a max–lp2 pair gives the largest value of $\sigma(\alpha|W_2)$ it follows from Definition 4 and Theorem 6 that the largest single value of τ that can be valid for all iterations of all problem instances is $1/(n-1)$. Thus a max–lp2 pair is optimal in that it achieves the minimum iteration bound in Theorem 5 with respect to τ. Furthermore Simon (2004) has introduced an algorithm for computing a max–lp2 pair that requires only $O(n)$ computation and therefore coincides with the $O(n)$ computation required to perform the other steps in the main loop. However, in spite of the promise suggested by this analysis experimental results suggest that there is much room to improve the convergence rates achieved with max–lp2 pairs (e.g. see Section 4). The result below provides a simple way to determine pair selection methods whose convergence rates are at least as good as those guaranteed by the max–lp2 pair method and possibly much better. This result is stated as a corollary since it follows trivially from the proof of Theorem 5.

Corollary 7 *Let* DECOMP *be a realization of the model decomposition algorithm for the canonical dual QP in Procedure 2 and let* (α^m) *represent a sequence of feasible points produced by this algorithm. At each iteration* m *let* $\acute{W}_2^m$ *be a* τ*–rate certifying pair and let* $\acute{\alpha}^{m+1}$ *be the feasible point determined by solving the restricted QP determined by* α^m *and* $\acute{W}_2^m$*. If for every* $m \geq 0$ *the stepwise improvement satisfies* $R(\alpha^{m+1}) - R(\alpha^m) \geq R(\acute{\alpha}^{m+1}) - R(\alpha^m)$ *then* DECOMP *will achieve* $R^* - R(\alpha^m) \leq \varepsilon$ *after* $\lceil \acute{m} \rceil$ *iterations of the main loop where* $\acute{m}$ *is given by Theorem 5.*

This theorem implies that any pair whose stepwise improvement is at least as good as that produced by a max–lp2 pair yields a decomposition algorithm that inherits the iteration bound in Theorem 5 with $\tau = 1/(n-1)$. An obvious example is a *max–qp2* pair, which is a pair with the *largest* stepwise improvement. However since determining such a pair may require substantial computation we seek alternatives. In particular Simon's algorithm visits several good candidate pairs in its search for a max–lp2 pair and can therefore be easily extended to form an alternative pair selection algorithm that is computationally efficient and satisfies this stepwise improvement property. To see this we start with a description of Simon's algorithm.

First note that when searching for a max–lp2 pair it is sufficient to consider only pairs (j,k) where $g_j(\alpha) > g_k(\alpha)$. For such a pair it is easy to show that (e.g. see the proof of Theorem 6)

$$\sigma(\alpha|\{j,k\}) = \min(u_j - \alpha_j, \alpha_k)(g_j(\alpha) - g_k(\alpha)) = \Delta_{jk}\,(g_j(\alpha) - g_k(\alpha)) \tag{10}$$

where u_j is the upper bound on α_j specified in (4) and $\Delta_{jk} := \min(u_j - \alpha_j, \alpha_k)$. The key to Simon's algorithm is the recognition that among the $O(n^2)$ index pairs there are at most $2n$ distinct values for Δ:

$$u_1 - \alpha_1,\ \alpha_1,\ u_1 - \alpha_2,\ \alpha_2,\ \ldots,\ u_n - \alpha_n,\ \alpha_n. \tag{11}$$

Consider searching this list of values for one that corresponds to a maximum value of σ. For an entry of the form $u_j - \alpha_j$ for some j, an index k that maximizes $\sigma(\alpha|\{j,k\})$ satisfies

$$k \in \arg\max_{l:\alpha_l \geq u_j - \alpha_j} (g_j(\alpha) - g_l(\alpha)) \;=\; \arg\min_{l:\alpha_l \geq u_j - \alpha_j} g_l(\alpha)\,.$$

Similarly for an entry of the form α_k for some k, an index j that maximizes $\sigma(\alpha|\{j,k\})$ satisfies

$$j \in \arg\max_{l:u_l - \alpha_l \geq \alpha_k} (g_l(\alpha) - g_k(\alpha)) \;=\; \arg\max_{l:u_l - \alpha_l \geq \alpha_k} g_l(\alpha)\,.$$

Now suppose we search the list of values from largest to smallest and keep track of the maximum gradient component value for entries of the form $u_j - \alpha_j$ and the minimum gradient component

value for entries of the form α_k as we go. Then as we visit each entry in the list the index pair that maximizes σ can be computed in constant time. Thus a max–lp2 pair can be determined in one pass through the list. A closer examination reveals that only the nonzero values at the front of the list need to be scanned, since entries with zero values cannot form a certifying pair (i.e. they correspond to pairs for which there is no feasible direction for improvement). In addition, since nonzero entries of the form $u_j - \alpha_j$ correspond to components j where $\alpha_j < u_j$, and nonzero entries of the form α_k correspond to components k where $\alpha_k > 0$, once the scan reaches the last nonzero entry in the list the indices of the maximum and minimum gradient component values correspond to a max–violating pair. Pseudocode for this algorithm is shown in Procedure 4 in Appendix 6. This algorithm requires that the ordered list of values be updated at each iteration. If the entries are stored in a linear array this can be accomplished in $O(pn)$ time by a simple *search and insert* algorithm, where p is the size of the working set. However, with the appropriate data structure (e.g. a red–black tree) this list can be updated in $O(p \log n)$ time. In this case the size of the working sets must satisfy $p = O(n/\log n)$ to guarantee an $O(n)$ run time for the main loop.

Simon's algorithm computes both a max–lp2 pair and a max–violating pair at essentially the same cost. In addition the stepwise improvement for an individual pair can be computed in constant time. Indeed with $W_2^m = \{j,k\}$ and $g(\alpha_j^m) \geq g(\alpha_k^m)$ the stepwise improvement δ_R^m takes the form

$$\delta_R^m = \begin{cases} \Delta\delta_g - \Delta^2 q/2, & \delta_g > q\Delta \\ \frac{\delta_g^2}{2q}, & \text{otherwise} \end{cases} \tag{12}$$

where $\delta_g = g(\alpha_j^m) - g(\alpha_k^m)$, $q = Q_{jj} + Q_{kk} - 2Q_{jk}$ and $\Delta = \min\left(u_j - \alpha_j^m, \alpha_k^m\right)$. Thus we can efficiently compute and compare the stepwise improvements of the max–violating and max–lp2 pairs and choose the one with the largest improvement. We call this the *Composite–I* pair selection method. It adds a negligible amount of computation to the main loop and its stepwise improvement cannot be worse than either the max–violating pair or max–lp2 algorithm alone. We can extend this idea further by computing the stepwise improvement for all certifying pairs visited by Simon's algorithm and then choosing the best. We call this the *Composite–II* pair selection method. This methods adds a *non–negligible* amount of computation to the main loop, but may provide even better stepwise updates. It is worth mentioning that other methods have been recently introduced which examine a subset of pairs and choose the one with the largest stepwise improvement (e.g. see Fan et al., 2005; Lai et al., 2003). The methods described here are different in that they are designed specifically to satisfy the condition in Corollary 7.

We have described four pair selection methods; max–lp2, Composite–I (best of max–violating and max–lp2), Composite–II (best of certifying pairs visited by Simon's algorithm), and max–qp2 (largest stepwise improvement) which all yield decomposition algorithms that satisfy the iteration bound in Theorem 5 with $\tau = 1/(n-1)$, but whose *actual* computational requirements on a specific problem may be quite different. In Section 4 we perform experiments to investigate the actual computational requirements for these methods.

2.2 Stopping Rules

Algorithms derived from the model in Procedure 2 require a stopping rule. Indeed to achieve the run time guarantees described in the previous section the algorithms must be terminated properly. The most common stopping rule is based on the observation that, prior to convergence, a max–violating pair (j^*,k^*) represents the most extreme violator of the optimality conditions in (7). This suggests

the stopping rule: *stop at the first iteration $\acute{m}$ where*

$$g_{j^*}(\alpha^{\acute{m}}) - g_{k^*}(\alpha^{\acute{m}}) \leq tol \tag{13}$$

where $tol > 0$ is a user defined parameter. This stopping rule is employed by many existing decomposition algorithms (e.g. see Chang and Lin, 2001; Chen et al., 2006; Keerthi et al., 2001; Lin, 2002a) and is especially attractive for max–violating pair algorithms since the rule can be computed in constant time once a max–violating pair has been computed. Lin (2002a) justifies this rule by proving that the gap $g_{j^*}(\alpha^m) - g_{k^*}(\alpha^m)$ converges to zero asymptotically for the sequence of feasible points generated by a particular class of decomposition algorithms. In addition Keerthi and Gilbert (2002) prove that (13) is satisfied in a finite number of steps for a specific decomposition algorithm. However the efficacy of this stopping rule is not yet fully understood. In particular we do not know the relation between this rule and the accuracy of the approximate solution it produces, and we do not know the convergence rate properties of the sequence $(g_{j^*}(\alpha^m) - g_{k^*}(\alpha^m))$ on which the rule is based. In contrast we now introduce new stopping rules which guarantee a specified accuracy for the approximate solutions they produce, and whose convergence rate properties are well understood. In addition we will show that these new stopping rules can be computed in constant time when coupled with the pair selection strategies in the previous section.

The simplest stopping rule that guarantees an ε–optimal solution for a τ–rate certifying algorithm is to stop after $\acute{m}$ iterations where $\acute{m}$ is given by Theorem 5 with $R^* - R(\alpha^0)$ replaced by a suitable upper bound (e.g. 1). We call this *Stopping Rule 0*. However the bound in Theorem 5 is conservative. For a typical problem instance the algorithm may reach the accuracy ε in far fewer iterations. We introduce stopping rules that are tailored to the problem instance and therefore may terminate the algorithm much earlier. These rules compute an upper bound on $R^* - R(\alpha)$ *adaptively* and then stop the algorithm when this upper bound falls below ε. There are many ways to determine an upper bound on $R^* - R(\alpha)$. For example the primal-dual gap, which is the difference between the primal criterion value and the dual criterion value, provides such a bound and therefore could be used to terminate the algorithm. However, computing the primal-dual gap would add significant computation to the main loop and so we do not pursue it here. Instead we develop stopping rules that, when coupled with one of the pair selection methods in the previous section, are simple to compute. These rules use the bound $R^* - R(\alpha) \leq \sigma(\alpha|W_2)/\tau$ which was first established by Hush and Scovel (2003) and is reestablished as part of the theorem below. The theorem and corollary below establish the viability of these rules by proving that this bound converges to zero as $R(\alpha^m) \to R^*$, and that if $R(\alpha^m) \to R^*$ at a certain rate then the bound converges to zero at a similar rate.

Theorem 8 *Consider the canonical dual QP problem in (4) with Gram matrix Q, constraint vector u, feasible set $\mathcal{A}$, criterion function R, and optimal criterion value R^*. Let $\alpha \in \mathcal{A}$ and let W_p be a size p working set. Then the gap $R^* - R(\alpha)$ is bounded below and above as follows:*

1. *Let $L \geq max_i Q_{ii}$ and*

$$\sup_{\{V_p : V_p \subseteq W_n\}} \sum_{i \in V_p} u_i^2 \;\; \leq \;\; U_p$$

where the supremum is over all size p subsets of W_n. Then

$$\frac{\sigma(\alpha|W_p)}{2} \min\left(1, \frac{\sigma(\alpha|W_p)}{pLU_p}\right) \;\; \leq \;\; R^* - R(\alpha). \tag{14}$$

2. *If W_p includes a τ–rate certifying pair for α then*

$$R^* - R(\alpha) \;\leq\; \frac{\sigma(\alpha|W_p)}{\tau}. \tag{15}$$

The next corollary follows trivially from Theorem 8.

Corollary 9 *Consider the canonical dual QP problem in (4) with criterion function R. For any sequence of feasible points (α^m) and corresponding sequence of working sets (W^m) that include τ–rate certifying pairs the following holds:*

$$R(\alpha^m) \to R^* \quad \Leftrightarrow \quad \sigma(\alpha^m|W^m) \to 0.$$

In addition, rates for $R(\alpha^m) \to R^*$ imply rates for $\sigma(\alpha^m|W^m) \to 0$.

This corollary guarantees that the following stopping rule will eventually terminate a τ–rate certifying algorithm, and that when terminated at iteration $\acute{m}$ it will produce a solution $\alpha^{\acute{m}}$ that satisfies $R(\alpha^{\acute{m}}) - R^* \leq \varepsilon$.

Definition 10 (Stopping Rule 1) *For a τ–rate certifying algorithm with τ–rate certifying pair sequence (W_2^m), stop at the first iteration $\acute{m}$ where $\sigma(\alpha^{\acute{m}}|W_2^{\acute{m}}) \leq \tau\varepsilon$.*

This rule can be implemented in constant time using (10). The effectiveness of this rule will depend on the tightness of the upper bound in (15) for values of α near the optimum. We can improve this stopping rule as follows. Define

$$\delta_R^m := R(\alpha^{m+1}) - R(\alpha^m)$$

and suppose we have the following bound at iteration m

$$R^* - R(\alpha^m) \leq s.$$

Then at iteration $m+1$ we have

$$R^* - R(\alpha^{m+1}) \leq \min\left(\frac{\sigma(\alpha^{m+1}|W_2^{m+1})}{\tau},\; s - \delta_R^m\right).$$

Thus an initial bound s^0 (e.g. $s^0 = \sigma^0/\tau$) can be improved using the recursion

$$s^{m+1} = \min\left(\frac{\sigma(\alpha^{m+1}|W_2^{m+1})}{\tau}, s^m - \delta_R^m\right)$$

which leads to the following stopping rule:

Definition 11 (Stopping Rule 2) *For a τ–rate certifying algorithm with τ–rate certifying pair sequence (W_2^m), stop at the first iteration $\acute{m}$ where $s^{\acute{m}} \leq \varepsilon$.*

This rule is at least as good as Stopping Rule 1 and possibly better. However it requires that we additionally compute the stepwise improvement $\delta_R^m = R(\alpha^{m+1}) - R(\alpha^m)$ at each iteration. In the worst case, since the criterion can be written $R(\alpha) = \frac{1}{2}\alpha\cdot(g(\alpha)+w)+w_0$, the stepwise improvement δ_R^m can be computed in $O(n)$ time (assuming $g(\alpha^m)$ has already been computed). However for *W2* variants this value can be computed in constant time using (12). In Section 4 we describe experimental results that compare all three stopping rules.

2.3 Computing the Offset

We have concluded our description of algorithms for the Decomposition routine in Procedure 1 and now proceed to describe an algorithm for the Offset routine. According to Theorem 2 this routine must solve

$$\hat{b} \in \arg\min_b \sum_{i=1}^n u_i \max\left(0, 1 - y_i(\hat{\psi}\cdot\phi(x_i) + b)\right).$$

An efficient algorithm for determining $\hat{b}$ is enabled by using (5) and (6) to write

$$\begin{aligned} 1 - y_i\hat{\psi}\cdot\phi(x_i) &= 1 - y_i\left(\frac{1}{2\lambda}\sum_{j=1}^n (\hat{\alpha}_j - l_j)k(x_j, x_i)\right) \\ &= 1 - y_i(Q(\hat{\alpha} - l))_i \;=\; y_i\big(w_i - (Q\hat{\alpha})_i\big) \;=\; y_i g_i(\hat{\alpha})\,. \end{aligned}$$

This simplifies the problem to

$$\hat{b} \in \arg\min_b \sum_{i=1}^n u_i \max\left(0,\; y_i\big(g_i(\hat{\alpha}) - b\big)\right).$$

The criterion $\sum_{i=1}^n u_i \max\left(0,\; y_i\big(g_i(\hat{\alpha}) - b\big)\right)$ is the sum of hinge functions with slopes $-u_i y_i$ and b–intercepts $g_i(\hat{\alpha})$. It is easy to verify that the finite set $\{g_i(\hat{\alpha}), i = 1,\ldots,n\}$ contains an optimal solution $\hat{b}$. To see this note that the sum of hinge functions creates a piecewise linear surface where minima occur at corners, and also possibly along flat spots that have a corner at each end. Since the corners coincide with the points $g_i(\hat{\alpha})$ the set $\{g_i(\hat{\alpha}), i = 1,\ldots,n\}$ contains an optimal solution. The run time of the algorithm that performs a brute force computation of the criterion for every member of this set is $O(n^2)$. However this can be reduced to $O(n\log n)$ by first sorting the values $g_i(\hat{\alpha})$ and then visiting them in order, using constant time operations to update the criterion value at each step. The details are shown in Procedure 8 in Appendix 6.

2.4 A Complete Algorithm

We have now described a complete algorithm for computing an ε_p–optimal solution to the primal QP problem. A specific realization is provided by (Procedure 1,Section 2) and Procedures 3–8 in Appendix 6. Multiple options exist for the Decomposition routine depending on the choice of working set size, pair selection method, and stopping rule. The realization in the appendix implements a *W2* variant of the Composite–I decomposition algorithm with Stopping Rule 2 (and is easily modified to implement the Composite–II algorithm). In the next two sections we complete our run time analysis of decomposition algorithms.

3. Operational Analysis of Decomposition Algorithms

In this section we use Theorem 5 and Corollary 7 to determine run time bounds for rate certifying decomposition algorithms that are applied to the L1–SVM and DLD–SVM canonical dual QP problems. It is clear from Theorem 5 that these bounds will depend on the parameters τ, S, L, R^* and ε. Let us consider each of these in turn. In the algorithms below each working set contains either a max–lp2 pair or a pair whose stepwise improvement is at least as good as that of a max–lp2 pair. Thus by Corollary 7 we can set $\tau = 1/(n-1)$. Instead however we set $\tau = 1/n$ since this value

is also valid and it greatly simplifies the iteration bounds without changing their basic nature. The parameter S will take on a different, but known, value for the L1–SVM and DLD–SVM problems as described below. Using the definition of L in Theorem 5 and the definition of Q in (5) we set $L = \frac{K}{2\lambda}$ where $K \geq \max_{1\leq i\leq n} k(x_i,x_i)$. We consider two possibilities for K. The first is the value

$$K_n = \max_{1\leq i\leq n} k(x_i,x_i)$$

which is used to bound the run time for a specific problem instance and the second is the constant

$$\bar{K} = \sup_{x\in X} k(x,x)$$

which is used to bound the run time for classes of problem instances that use the same kernel, e.g. SVM learning problems where the kernel is fixed. In the second case we are interested in problems where $\bar{K}$ is finite. For example for the Gaussian RBF kernel $k(x,x') = e^{-\sigma\|x-x'\|^2}$ we obtain $\bar{K} = 1$. The optimal criterion value R^* is unknown but restricted to $[0,1]$. To see this we use (5) to obtain

$$R(\alpha) \;=\; -\frac{1}{2}\alpha\cdot Q\alpha + \alpha\cdot w + w_0 \;=\; -\frac{1}{2}(\alpha - l)\cdot Q(\alpha - l) + (\alpha - l)\cdot y.$$

Then since $l \in \mathcal{A}$ it follows that $R^* \geq R(l) = 0$. Furthermore, using the positivity of Q and the definition of l in (3) we obtain that for any $\alpha \in \mathcal{A}$ the bound

$$R(\alpha) \;=\; -\frac{1}{2}(\alpha - l)\cdot Q(\alpha - l) + (\alpha - l)\cdot y \;\leq\; (\alpha - l)\cdot y \;\leq\; u\cdot 1 = 1$$

holds. We have now considered all the parameters that determine the iteration bound except λ and ε which are chosen by the user.

Recent theoretical results by Steinwart and Scovel (2004, 2005); Scovel et al. (2005b) indicate that with a suitable choice of kernel and mild assumptions on the distribution the trained classifier's generalization error will approach the Bayes error at a fast rate if we choose $\lambda \propto n^{-\beta}$, where the rate is determined (in part) by the choice of $0 < \beta < 1$. Although these results hold for exact solutions to the primal QP problem it is likely that similar results will hold for approximate solutions as long as $\varepsilon_p \to 0$ at a sufficiently fast rate in n. However in practice there is little utility in improving the performance once it is sufficiently close to the Bayes error. This suggests that once we reach a suitably large value of n there may be no need to decrease λ and ε_p below some fixed values $\bar{\lambda}$ and $\bar{\varepsilon}_p$. Thus, for fixed values $\bar{\lambda} > 0$ and $\bar{\varepsilon}_p > 0$ we call any (λ, ε_p) that satisfies $\lambda \geq \bar{\lambda}$ and $\varepsilon_p \geq \bar{\varepsilon}_p$ an *operational* choice of these parameters. When $\bar{K}$ is finite Theorem 2 gives a corresponding fixed value $\bar{\varepsilon} = (2\sqrt{2\bar{K}} + 8\sqrt{\bar{\lambda}})^{-2}\bar{\lambda}\bar{\varepsilon}_p^2 > 0$ that we use to define an operational choice of the dual accuracy ε.

We begin our analysis by considering decomposition algorithms for the L1–SVM problem. Although our emphasis is on rate certifying decomposition algorithms, our first theorem establishes a lower bound on the number of iterations for *any W2* decomposition algorithm.

Theorem 12 *Consider the L1–SVM canonical dual with optimal criterion value R^*. Any W2 variant of Procedure 2 that starts with $\alpha^0 = l$ will achieve $R^* - R(\alpha^m) \leq \varepsilon$ in no less than $\lceil \bar{m} \rceil$ iterations where*

$$\bar{m} \;=\; \max\left(0,\; \frac{n(R^* - \varepsilon)}{2}\right).$$

Remark 13 *When $R^* > \varepsilon$ the minimum number of iterations is proportional to n and increases linearly with R^*. Thus it is important to understand the conditions where R^* is significantly larger than ε. Under very general conditions it can be shown that, with high probability, $R^* \geq e^* - \varepsilon_n$ where e^* is the Bayes classification error and ε_n is a term that tends to 0 for large n. Thus, for large n, R^* will be significantly larger than ε when e^* is significantly larger than ε, which we might expect to be common in practice.*

We briefly outline a path that can be used to establish a formal proof of these claims. Since the duality gap for the L1–SVM primal and dual QP problems is zero, R^* is the optimal value of the primal QP problem (e.g. for finite and infinite dimensional problems respectively see Cristianini and Shawe-Taylor, 2000; Hush et al., 2005). Furthermore it is easy to show that R^* is greater than or equal to the corresponding empirical classification error (i.e. the training error). Therefore the error deviance result in (Hush et al., 2003) can be used to establish general conditions on the data set $T_n = ((x_1, y_1), \ldots, (x_n, y_n))$, the kernel k, and the regularization parameter λ such that the bound $R^* \geq e^* - \varepsilon_n$ holds with probability $1 - \delta$, where $\varepsilon_n = O\left(\sqrt{\ln(\sqrt{n}/\delta)/n}\right)$. Since e^* is a constant it can be further shown that with a suitably chosen constant $c > 0$ and a sufficiently large value n_0, then $Pr\left(\text{number of iterations } \geq \frac{n(e^* - \varepsilon)}{2+c}, \forall n \geq n_0)\right) \geq 1 - \delta_{n_0}$ where $\delta_{n_0} \to 0$ at a rate that is exponential in n_0. Thus when $e^* > \varepsilon$ we can prove that the number of iterations is $\Omega(n)$ with probability 1.

We now continue our analysis by establishing upper bounds on the computation required for rate certifying decomposition algorithms applied to the L1–SVM and the DLD–SVM problems. In the examples below we establish two types of computation bounds: *generic bounds* which hold for any value of n, any choice of $\lambda > 0$, and either value of K; and *operational bounds* that hold when $K = \bar{K}$ is finite and operational choices are made for ε and λ. In the latter case we obtain bounds that are uniform in λ and ε and whose constants depend on the operational limits $\underline{\varepsilon}$ and $\bar{\lambda}$. These bounds are expressed using $O(\cdot)$ notation which suppresses their dependence on $\bar{K}$, $\underline{\varepsilon}$ and $\bar{\lambda}$ but reveals their dependence on n. In both examples we first consider a general class of rate certifying decomposition algorithms whose working sets may be larger than 2. For these algorithms we establish generic and operational bounds on the number of iterations. Then we consider the *W2* variants of these algorithms and establish operational bounds on their overall run time.

Example 1 *Consider solving the L1–SVM canonical dual using a decomposition algorithm where each working set includes a certifying pair whose stepwise improvement is at least as good as that produced by a max–lp2 pair. This includes algorithms where each working set includes a max–lp2, Composite–I, Composite–II or max–qp2 pair. Applying Theorem 5 with $S = 1/n$, $L = K/2\lambda$, $R^* - R(\alpha^0) \leq 1$, $\tau = 1/n$ and $\varepsilon < 1$ gives the generic bound*

$$\acute{m} \leq \begin{cases} 2n \ln\left(\dfrac{1}{\varepsilon}\right), & \varepsilon \geq \dfrac{2K}{\lambda n} \\ 2n\left(\dfrac{2K}{\lambda \varepsilon n} - 1 + \ln\left(\dfrac{\lambda n}{2K}\right)\right), & \varepsilon < \dfrac{2K}{\lambda n} \end{cases} \tag{16}$$

on the number of iterations. With $K = K_n$ this expression gives a bound on the number of iterations for a specific problem instance. When $K = \bar{K}$ is finite, operational choices are made for ε and λ,

and n is large the number of iterations is determined by the first case and is $O(n)$. This matches the lower bound in Remark 13 and is therefore optimal in this sense. For a W2 variant that uses an algorithm from Section 2.1 to compute a max–lp2, Composite–I or Composite–II pair at each iteration the main loop requires $O(n)$ computation to determine the pair, $O(\log n)$ computation to update the ordered list M, $O(1)$ computation to update α, and $O(n)$ computation to update the gradient. Thus the main loop requires a total of $O(n)$ computation. Combining the bounds on the number of iterations and the computation per iteration we obtain an overall computational requirement of $O(n^2)$. In contrast, for a W2 variant that computes a max–qp2 pair at each iteration the main loop computation will increase. Indeed the current best algorithm for computing a max–qp2 pair is a brute force search which requires $O(n^2)$ computation and we strongly suspect that this cannot be reduced to the $O(n)$ efficiency of Simon's algorithm. Combining this with the lower bound on the number of iterations in Remark 13 demonstrates that there are cases where the overall run time of the max–qp2 variant is inferior.

Example 2 *Consider solving the DLD–SVM canonical dual using a decomposition algorithm where each working set includes a certifying pair whose stepwise improvement is at least as good as that produced by a max–lp2 pair. In this case we can determine a value for S as follows,*

$$\max_i u_i = \max\left(\frac{1}{(1+\rho)n_1}, \frac{\rho}{(1+\rho)n_{-1}}\right) \leq \max\left(\frac{1}{n_1}, \frac{1}{n_{-1}}\right) = \frac{1}{\min(n_1, n_{-1})} := S$$

where n_1 and n_{-1} are the number of samples with labels $y = 1$ and $y = -1$ respectively as described in Section 2. Suppose that $n_1 \leq n_{-1}$ (results for the opposite case are similar). Applying Theorem 5 with $L = K/2\lambda$, $R^* - R(\alpha^0) \leq 1$, and $\tau = 1/n$ gives the generic bound

$$\acute{m} \leq \begin{cases} 2n\ln\left(\dfrac{1}{\varepsilon}\right), & \varepsilon \geq \dfrac{2Kn}{\lambda n_1^2} \\ 2n\left(\dfrac{2Kn}{\varepsilon\lambda n_1^2} - 1 + \ln\left(\dfrac{\lambda n_1^2}{2Kn}\right)\right), & \varepsilon < \dfrac{2Kn}{\lambda n_1^2} \end{cases} \tag{17}$$

on the number of iterations. The dependence on n_1 distinguishes this bound from the bound in (16). With $K = K_n$ (17) gives a bound on the number of iterations for a specific problem instance. Suppose that $n_1 = \Omega(n)$. Then when $K = \bar{K}$ is finite, operational choices are made for ε and λ, and n is large the number of iterations is determined by the first case and is $O(n)$. For a W2 *variant that uses an algorithm from Section 2.1 to compute a max–lp2, Composite–I or Composite–II pair at each iteration the main loop requires $O(n)$ computation. Thus the overall computational requirement is $O(n^2)$.*

4. Experiments

The experiments in this section are designed to accomplish three goals: to investigate the utility of Stopping Rules 1 and 2 by comparing them with Stopping Rule 0, to compare actual versus worst case computational requirements, and to investigate the computational requirements of *W2* decomposition algorithms that use different pair selection methods. Our focus is on the computational requirements of the main loop of the decomposition algorithm since this loop contributes a

dominating term to our run time analysis, and since the computational requirements of the other algorithmic components can be determined very accurately without experimentation. We compare the four rate certifying pair selection methods (max–qp2, max–lp2, Composite–I, Composite–II) described in Section 2.1, and a max–violating pair method that we call *max–vps*. This max–vps algorithm is identical to the Composite–I algorithm, except that when choosing between a max–lp2 and max–violating pair we always choose the max–violating pair. To provide objective comparisons all algorithms use the same stopping rule. This means that the max–vps algorithm uses a different stopping rule than existing max–violating algorithms. Nevertheless including the max–vps algorithm in our experiments helps provide insight into how the algorithms developed here might compare with existing algorithms.

Our experiments are based on two different problems: a DLD–SVM problem formed from the **Cyber–Security** data set described in (Steinwart et al., 2005) and an L1–SVM problem formed from the **Spambase** data set from the UCI repository (Blake and Merz, 1998). All experiments employ SVMs with a Gaussian RBF kernel $k(x,x') = e^{-\sigma\|x-x'\|^2}$. Since a value of the regularization parameter (λ,σ) that optimizes performance is usually not known ahead of time, the value that is ultimately used to design the classifier is usually determined through some type of search that requires running the algorithm with different values of (λ,σ). Thus it is important to understand how different values, optimal and otherwise, affect the run time. To explore this effect we present results for two different values, (λ^*,σ^*) and $(\bar{\lambda},\bar{\sigma})$, obtained as follows. We train the SVM at a set of grid values and choose (λ^*,σ^*) to be a value that gives the best performance on an independent validation data set [3]. Then $(\bar{\lambda},\bar{\sigma})$ is chosen to be some other grid value encountered during the search that yielded non–optimal but nontrivial performance (i.e. it achieves some separation of the training data). For the DLD–SVM the performance is defined by the risk function $\mathcal{R}$ in (Steinwart et al., 2005) and for the L1–SVM it is the average classification error.

The **Cyber–Security** data set was derived from network traffic collected from a single computer over a 16-month period. The goal is to build a detector that will recognize anomalous behavior from the machine. Each data sample is a 12–dimensional feature vector whose components represent real valued measurements of network activity over a one-hour window (e.g. "average number of bytes per session"). Anomalies are defined by choosing a uniform reference distribution and a density level $\rho = 1$. The parameter values $(\lambda^*,\sigma^*) = (10^{-7},10^{-1})$ and $(\bar{\lambda},\bar{\sigma}) = (.05,.05)$ were obtained by employing a grid search with n_1:n_{-1} = 4000:10,000 training samples and 2000:100,000 validation samples. The solution obtained with parameter values (λ^*,σ^*) separated the training data and gave a validation risk of $\mathcal{R} = 0.00025$. The corresponding *alarm rate* (i.e. the rate at which anomalies are predicted by the classifier once it is placed in operation) is 0.0005.

The **Spambase** data set contains 4601 samples from $\mathbb{R}^d_+ \times \{-1,1\}$ where $d = 57$. This data set contains 1813 samples with label $y = -1$ and 2788 samples with label $y = 1$. The parameter values $(\lambda^*,\sigma^*) = (10^{-6},10^{-3})$ and $(\bar{\lambda},\bar{\sigma}) = (10^{-2},10^{-3})$ are obtained by employing a grid search with 3601 training samples and 1000 validation samples. The solution obtained with parameter values (λ^*,σ^*) did not separate the training data and gave a classification error of 0.093 on the validation set.

We present results for three experiments.

3. More specifically, for each value of $\lambda \in \{1,.5,.1,.05,\ldots,.000005,.0000001\}$ we search a grid of σ values that starts with the set $\{0.001,0.01,0.05,0.1,0.5,1,5,10,100\}$ and is refined using a golden search as described in (Steinwart et al., 2005, Section 4).

Experiment 1 *This experiment investigates the utility of Stopping Rules 1 and 2 by comparing them with Stopping Rule 0. More specifically we compare the actual criterion gap $R^* - R(\alpha^m)$ to the bounds used by these three stopping rules. We refer to the bounds for Stopping Rules 0, 1, and 2 as Bounds 0, 1, and 2 respectively. To obtain an estimate $\hat{R}^*$ of R^* we run the decomposition algorithm in Procedure 3 with $\varepsilon = 10^{-10}$ and compute the resulting criterion value. Then to obtain results for comparison we run this algorithm again and compute: the criterion gap $\hat{R}^* - R(\alpha^m)$, Bound 1 given by $n\sigma(\alpha^m|W_2^m)$, Bound 2 obtained from the recursive rule $s^m = \min(n\sigma(\alpha^m|W_2^m), s^{m-1} - \delta_R^{m-1})$, and Bound 0 given by equation (23) in the proof of Theorem 5.*

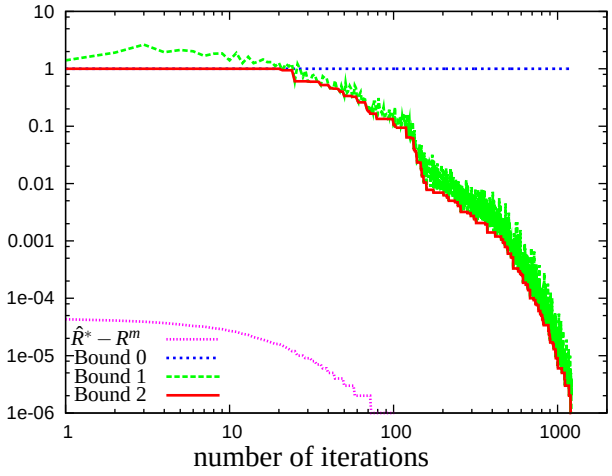


Figure 1: The criterion gap $\hat{R}^* - R^m$ and bounds on this gap employed by Stopping Rules 0, 1 and 2 for the **Cyber–Security** data. Bound 0 and 2 are indistinguishable up to about iteration 25, at which point they separate and Bound 2 becomes a monotonically decreasing lower envelope of Bound 1.

A plot of these values when the algorithm is applied to the **Cyber–Security** *data with $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$ and n_1:n_{-1} = 4000:10000 is shown in Figure 1. While Bound 1 is a bit erratic Bound 2 is monotonic and relatively smooth. Nevertheless both will stop the algorithm at nearly the same iteration (unless ε is very close to 1). In addition while Bounds 1 and 2 may be loose, i.e. they are often several orders of magnitude larger than the actual criterion gap, their behavior tracks that of the criterion gap relatively well and therefore the corresponding stopping rules are very effective relative to Rule 0. For example suppose we choose $\varepsilon = 10^{-5}$. Because the initial criterion gap is so small it takes only about* 25 *iterations for the algorithm to reach this accuracy. Both Stopping Rules 1 and 2 terminate the algorithm after approximately* 1000 *iterations, but Stopping Rule 0 terminates after approximately 1.225×10^{13} iterations (approximately* 10 *orders of magnitude more).*

Results obtained by applying the algorithm to the **Spambase** *data with $(\lambda^*, \sigma^*) = (10^{-6}, 10^{-3})$ and $n = 4601$ are shown in Figure 2. In this case the initial criterion gap is larger so the separation between the criterion gap and the bounds is smaller. Once again Bound 1 is a bit erratic, and this time there are several regions (beyond the initial region) where Bounds 1 and 2 are well separated. This suggests that the monotonic behavior of Bound 2 provides a more robust stopping rule. As before Bounds 1 and 2 are loose, but their behavior tracks that of the criterion gap relatively well and*

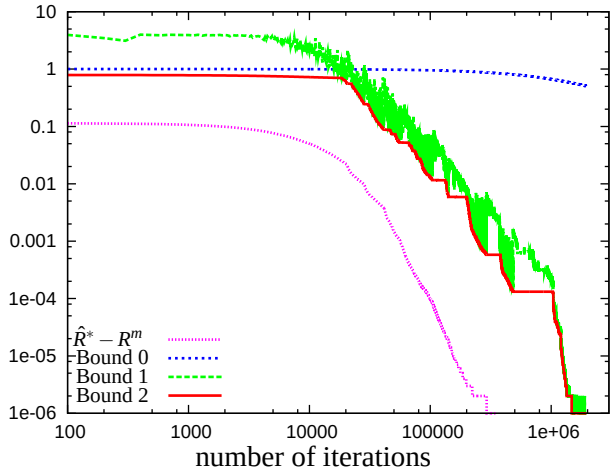


Figure 2: The criterion gap $\hat{R}^* - R^m$ and bounds on this gap employed by Stopping Rules 0, 1 and 2 for the **Spambase** data. Bound 0 and 2 are close up to about iteration 20,000, at which point they separate and Bound 2 becomes a monotonically decreasing lower envelope of Bound 1.

therefore the corresponding stopping rules are very effective. For example it takes about $200,000$ *iterations for the algorithm to reach an accuracy* $\varepsilon = 10^{-5}$*, while both Stopping Rules 1 and 2 terminate the algorithm after approximately* $2,000,000$ *iterations and Stopping Rule 0 terminates after approximately* 4×10^{11} *iterations (approximately* 5 *orders of magnitude more). More generally the number of the excess iterations for Stopping Rule 2 appears to be less than an order of magnitude for a large range of values of* ε*.*

In both cases above it is clear that Stopping Rules 1 and 2 are far superior to Stopping Rule 0.

Experiment 2 *This experiment compares actual computational requirements for the main loop of various decomposition algorithms applied to the* **Cyber–Security** *data. With density level* $\rho = 1$*, accuracy* $\varepsilon = 10^{-6}$*, parameter values* $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$ *and* $(\bar{\lambda}, \bar{\sigma}) = (.05, .05)$*, and five different problem sizes* $n_1{:}n_{-1}$ *= 2000:4000, 2500:5000, 3000:6000, 3500:7000, and 4000:8000 we employed the decomposition algorithm with Stopping Rule 2 and pair selection methods max–lp2, Composite–I, Composite–II, max–vps and max–qp2. For each problem size we generated ten different training sets by randomly sampling (without replacement) the original data set. Then we ran the decomposition algorithm on each training set and recorded the number of iterations and the wallclock time of the main loop. The minimum, maximum and average values of these quantities for parameter values* $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$ *are shown in Figure 3* [4]*. There is much to discern from the plot on the left. It is easy to verify that for all pair selection methods the numbers of iterations are several orders of magnitude smaller than the worst case bound given in Example 2. On average the convergence rate of the max–lp2 method is much worse than the other methods. This may be partly due to the fact that this method uses only first order information to determine its pair.*

4. In Figures 3–6 the x–axis values of some points are slightly offset so that their y–axis values can be more easily visualized.

However, this is also true of the max–vps method whose convergence rate is much faster. Indeed, it is curious that the max–lp2 method, which chooses a stepwise direction based on a combination of **steepness** *and* **room to move**, *has a worse convergence rate than the max–vps method, which chooses a stepwise direction based on* **steepness** *alone. By slightly modifying the max–lp2 method to obtain the Composite–I method a much faster convergence rate is observed. The Composite–I*

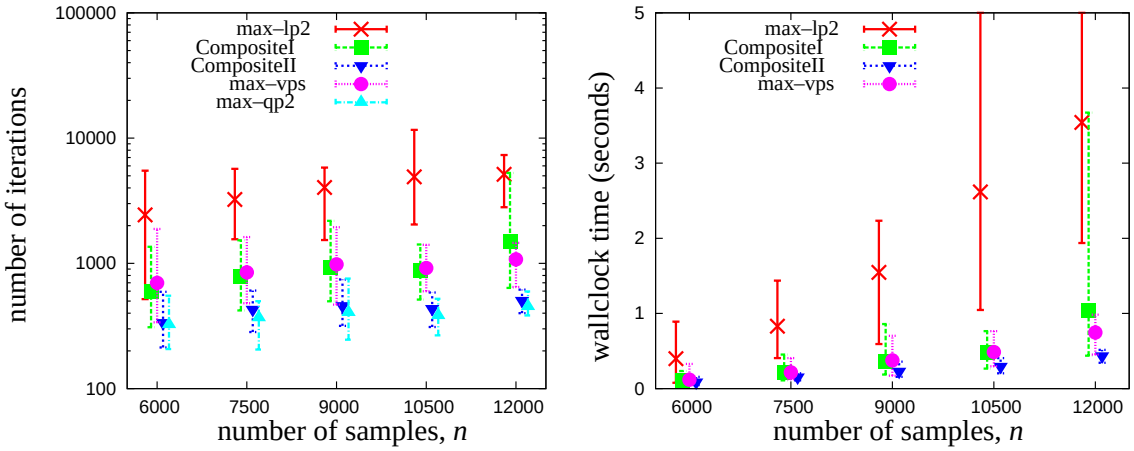


Figure 3: Main loop computation for **Cyber–Security** data with $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$.

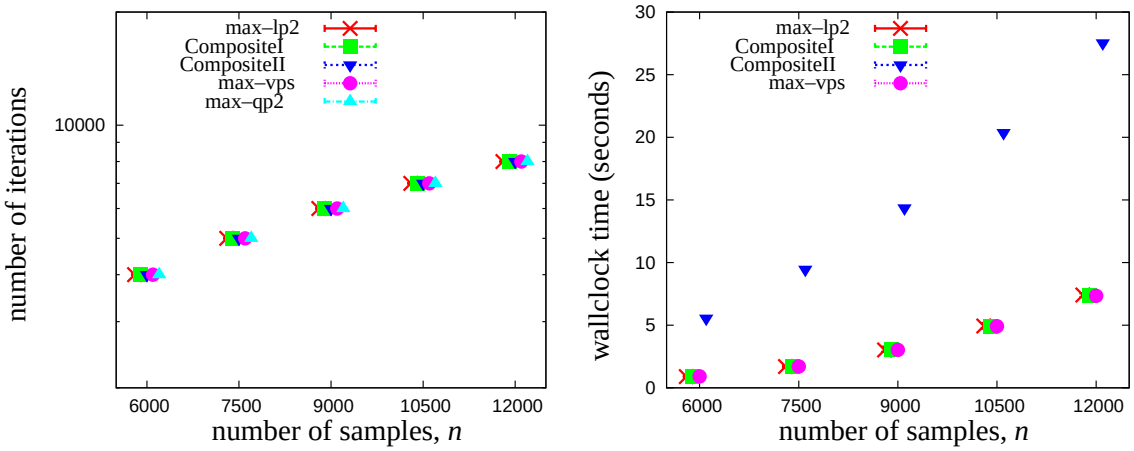


Figure 4: Main loop computation for **Cyber–Security** data with $(\bar{\lambda}, \bar{\sigma}) = (.05, .05)$. The number of iterations in the left plot is identical for all five methods for all values of n. The wallclock time in the right plot is indistinguishable for the Composite–I, max–vps and max–lp2 methods.

and max–vps methods have roughly the same convergence rate. This suggests that Composite–I may be achieving its improved rate by choosing a max–violating pair a large fraction of the time. Indeed, on a typical run of the Composite–I method we found that, among the 53% of the iterations where the max–lp2 and max–violating pairs were different, a max–violating pair was chosen 4.3 times as

often. Although a larger stepwise improvement does not guarantee a faster convergence rate the max–qp2 method, which gives the largest stepwise improvement, also gave the fastest convergence rate. However the Composite–II method, which requires far less computation than the max–qp2 method, gave nearly the same convergence rate. Quantitatively the average number of iterations for the max–lp2 method is roughly 9 times that of Composite–II, while the average number of iterations for Composite–I is roughly 2 times that of Composite–II. The variation in the number of iterations is smallest for Composite–II and max–qp2, followed by Composite–I and max–vps, and then max–lp2. This variation ranges from 2x to 8x across the different sample sizes and methods. The plot on the right shows the wallclock times. The times for the max–qp2 method are omitted because they are much larger than the rest. Indeed they are roughly n times larger than the wallclock times for Composite–II. The Composite–II method achieved the fastest average wallclock times which were roughly 6.8 times faster than the max–lp2 method and 1.6 times faster than the Composite–I and max–vps methods.

Results for parameter value $(\bar{\lambda}, \bar{\sigma}) = (.05, .05)$ *are shown in Figure 4. The computational requirements here are greater than with the previous parameter value. We attribute this primarily to the fact that* R^* *is larger so that the initial criterion gap is larger. The larger value of* λ *corresponds to a strong regularization term that produces a solution where all components of* α *are forced from their initial values at one bound to their final values at the opposite bound. To move all* $n_1 + n_{-1}$ *components of* α *to their opposite bound using working sets that contain one sample from each class requires* n_{-1} *iterations (since* $n_{-1} > n_1$*) and this is exactly what the algorithms did for all five pair selection methods on every training set. This is a quintessential example of a problem where the number of iterations must be (at least) a significant fraction of the number of training samples regardless of which algorithm is used. The resulting solution has the simple interpretation that its normal vector is the difference in class means. The wallclock times of the max–lp2, Composite–I and max–vps algorithms are roughly 5 times faster than the Composite–II algorithm because of the extra computation per iteration employed by Composite–II. The relationship between the number of iterations and the training set size is demonstrably linear, and the relationship between the wallclock times and the training set size is demonstrably quadratic. These relations coincide with the linear and quadratic forms predicted by the analysis in Section 3.*

Experiment 3 *This experiment is similar to the previous experiment except that the algorithms are applied to the* **Spambase** *data. With accuracy* $\varepsilon = 10^{-6}$*, parameter values* $(\lambda^*, \sigma^*) = (10^{-6}, 10^{-3})$ *and* $(\bar{\lambda}, \bar{\sigma}) = (10^{-2}, 10^{-3})$*, and seven different problem sizes* $n = 1000, 1500, 2000, 2500, 3000, 3500, 4000$ *we employed the decomposition algorithm with Stopping Rule 2 and pair selection methods max–lp2, Composite–I, Composite–II, max–vps and max–qp2. We ran the decomposition algorithm on ten different training sets for each problem size and recorded the number of iterations and the wallclock time of the main loop. The minimum, maximum and average values of these quantities for runs with parameter values* $(\lambda^*, \sigma^*) = (10^{-6}, 10^{-3})$ *are shown in Figure 5. Once again it is easy to verify that for all pair selection methods the numbers of iterations in the left plot are several orders of magnitude smaller than the worst case bound given in Example 1. In addition the convergence rate is fastest for the Composite–II and max–qp2 methods, followed by the Composite–I and max–vps methods, and then the max–qp2 method. In this case it appears that the max–vps method has a slight edge on the Composite–I method. On a typical run of the Composite–I method we found that, among the 64% of the iterations where the max–lp2 and max–violating pairs were different, a max–violating pair was chosen 3.9 times as often. The variation in the number of iterations, which*

ranges from 2x to 4x across the different sample sizes and methods, is smallest for Composite–II and max–qp2, followed by Composite–I and max–vps, and then max–lp2. Quantitatively the average number of iterations for max–lp2, Composite–I and max–vps is roughly 92, 13 and 11 times that of Composite–II respectively. In addition the average wallclock times of the max–lp2, Composite–I and max–vps are roughly 23.6, 3.8 and 2.5 times that of Composite–II respectively. Once again the plot on the right does not show the wallclock times for the max–qp2 method, but they are roughly $n/4$ times that of the Composite–II method.

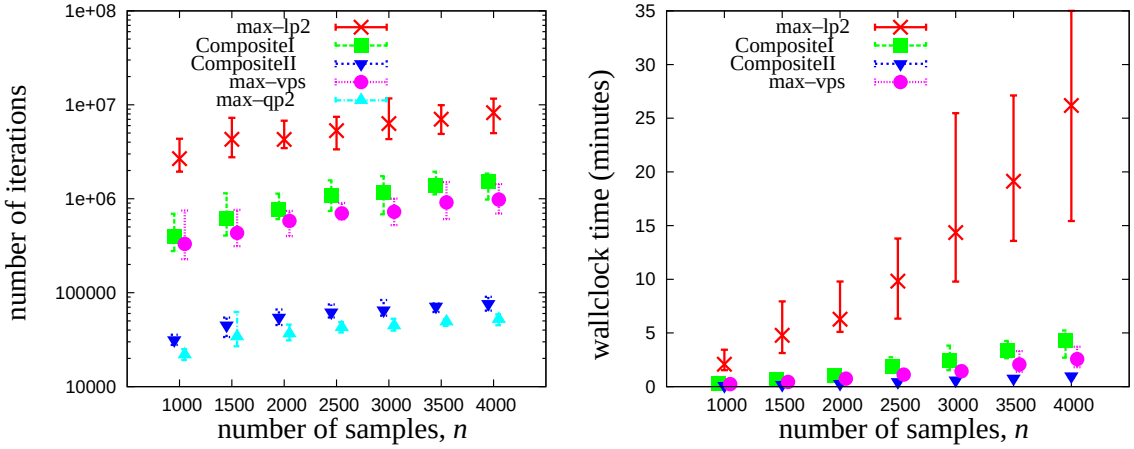


Figure 5: Main loop computation for **Spambase** data: $(\lambda^*, \sigma^*) = (10^{-6}, 10^{-3})$.

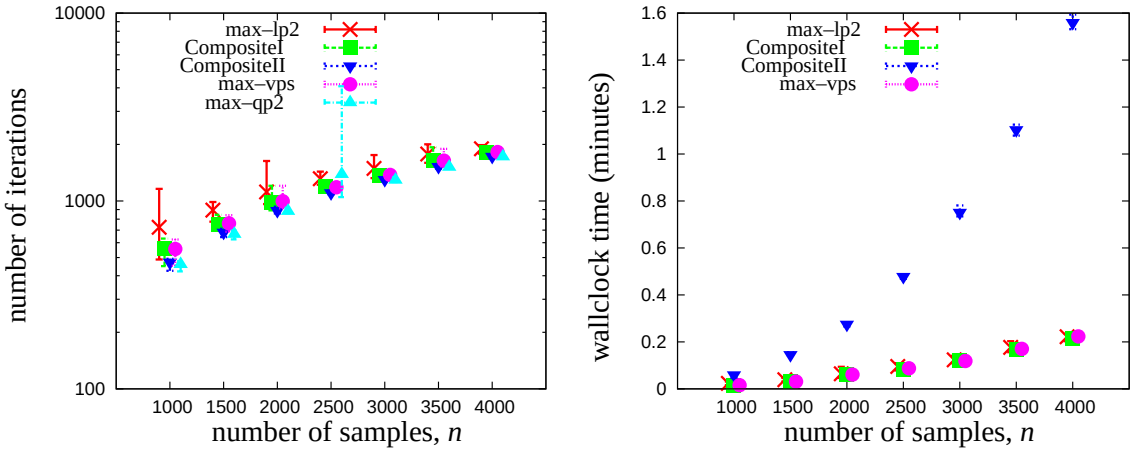


Figure 6: Main loop computation for **Spambase** data: $(\bar{\lambda}, \bar{\sigma}) = (10^{-2}, 10^{-3})$. The number of iterations in the left plot is similar for all three methods. The wallclock time in the right plot is nearly indistinguishable for the Composite–I and max–lp2 methods.

The results for parameter values $(\bar{\lambda}, \bar{\sigma}) = (10^{-2}, 10^{-3})$ are shown in Figure 6 and indicate a significant decrease in the computational requirements. This decrease in computation as a result of a larger λ is opposite to what we observed in Experiment 2. We attribute this to the fact that

the switch from (λ^*, σ^*) *to* $(\bar{\lambda}, \bar{\sigma})$ *did not yield a big change in the initial criterion gap as it did in Experiment 2. However most other characteristics of the solutions produced here are similar to those in Experiment 2. Indeed the number of iterations is roughly the same for all five pair selection methods and the wallclock times for the max–lp2, Composite–I and max–vps algorithms are approximately 5 times faster than Composite–II. In addition the relationships between the number of iterations, the wallclock times, and the training set size coincide with the linear and quadratic forms predicted by the analysis in the previous section.*

For the L1–SVM the gap between the lower and upper iteration bounds is smaller when λ is larger. Indeed, for large λ and large n the lower bound is $\frac{n}{2}(R^* - \varepsilon)$ and the upper bound is $2n \ln \frac{R^*}{\varepsilon}$. When R^* is large these two values may differ by no more than a factor of 10. This partially explains why the computational requirements for the strongly regularized problem instances in Experiments 2 and 3 exhibit such a low variance and coincide so well with the predicted linear and quadratic forms. In these cases the max–lp2, Composite–I and max–vps algorithms are fastest because they require less computation per iteration. On the other hand, in instances where (λ, σ) give near–optimal performance the values of λ are smaller and so the gaps between the lower and upper bounds are often much larger. In these cases the actual computation is often not close to either bound, the variance is higher, and the Composite–II algorithm is the fastest because it requires far fewer iterations. In addition these near–optimal values of λ can give a smaller value for R^*, especially when they yield a solution that separates the training data. In such cases the initial criterion gap is smaller and the run times are often faster. This is the most likely explanation for the significantly lower computational requirements for the **Cyber–Security** experiments.

5. Summary

We have described SVM classifier design algorithms that allow a different weight for each training sample. These algorithms accept an accuracy ε_p of a primal QP problem as input and are guaranteed to produce an approximate solution that satisfies this accuracy in low order polynomial time. They employ a two–stage process where the first stage produces an approximate solution to a dual QP problem and the second stage maps this approximate dual solution to an approximate primal solution. For the second stage we have described a simple $O(n \log n)$ algorithm that maps an approximate dual solution with accuracy $(2\sqrt{2K} + 8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$ to an approximate primal solution with accuracy ε_p. For the first stage we have presented new results for decomposition algorithms and we have described decomposition algorithms that employ new pair selection methods and new stopping rules.

For τ–*rate certifying* decomposition algorithms we have established the optimality of $\tau = 1/(n-1)$ and described several pair selection methods (max–qp2, max–lp2, Composite–I, Composite–II) that achieve the $\tau = 1/(n-1)$ iteration bound. We have also introduced new stopping rules that are computationally efficient and that guarantee a specified accuracy for the approximate dual solution. While these stopping rules can be used by any decomposition algorithm they are especially attractive for the algorithms developed here because they add a negligible amount of computation to the main loop.

Since the pair selection methods (max–lp2, Composite–I, Composite–II) require $O(n)$ computation they yield *W2* decomposition algorithms that require only $O(n)$ computation in the main loop. In addition, for the L1–SVM dual QP problem we have described operational conditions for which these *W2* decomposition algorithms possess an upper bound of $O(n)$ on the number of iterations.

For this same problem we have presented a lower bound for *any W2* decomposition algorithm and we have described general conditions for which this bound is $\Omega(n)$. Combining the bounds on main loop computation with the bounds on number of iterations yields an overall run time of $O(n^2)$. Our experiments suggest that the pair selection algorithms with the most promise are the Composite–I and Composite–II algorithms which were obtained through a simple extension of Simon's algorithm.

Once the run time of the decomposition algorithm has been established it is straightforward to determine the run time of the main routine in Procedure 1. Let c_k be an upper bound on the time it takes to perform a kernel evaluation. For an instance of L1–SVM where $\bar{K}$ is finite and operational choices are made for ε_p and λ Procedure 1 takes $O(c_k n^2)$ time to compute the parameters for the canonical dual on lines 7-8, $O(n)$ time to set α^0 on line 9, $O\left(n^2\right)$ time to compute an approximate dual solution on line 10, and $O(n \log n)$ time to compute the offset $\hat{b}$ on line 11. Thus, the overall run time is $O(n^2(c_k+1))$. This run time analysis assumes that the matrix Q is computed once and stored in main memory for fast (constant time) access. However the storage requirements for this matrix may exceed the size of main memory. If this issue is resolved by computing a kernel evaluation each time an element of Q is accessed then the time to compute an approximate dual solution is multiplied by c_k. On the other hand if the elements of Q are cached in a block of main memory so that the average access time for an element of Q is βc_k, where $0 < \beta \leq 1$ is determined by the size and replacement strategy for the cache, then the multiplier is reduced to βc_k for the average case. It is an interesting topic of future research to determine how the different pair selection methods affect the efficiency of the cache.

We note that algorithmic enhancements such as the shrinking heuristic in (Joachims, 1998) can easily be adapted to the algorithms presented here. In addition, the algorithms in this paper have been developed for the SVM formulation in (1), but similar algorithms with the same run time guarantees can be developed for the 1-CLASS formulation of Schölkopf et al. (2001) which has a similar form for the dual.

6. Proofs

The following lemma is used in the proofs of Theorems 5 and 8. It provides upper and lower bounds on the improvement in criterion value obtained by solving the restricted QP problem determined by a feasible point α and an *arbitrary* working set W_q.

Lemma 14 *Consider the canonical dual QP problem in (4) with Gram matrix Q, constraint vector u, feasible set $\mathcal{A}$, criterion function R, and optimal criterion value R^*. For $\alpha \in \mathcal{A}$ and a size q working set W_q let*

$$\alpha_q \in \arg \max_{\gamma \in \mathcal{A}(\alpha, W_q)} R(\gamma)$$

be a solution to the QP problem at (α, W_q). Then

$$R(\alpha_q) - R(\alpha) \;\leq\; \sigma(\alpha | W_q). \tag{18}$$

Furthermore, for $(\bar{\sigma}, L, U_q)$ satisfying $\bar{\sigma} \leq \sigma(\alpha|W_q)$, $L \geq max_i Q_{ii}$, and

$$\sup_{\{V_q : V_q \subseteq W_n\}} \sum_{i \in V_q} u_i^2 \;\leq\; U_q$$

where the supremum is over all size p subsets of W_n, the following bound holds,

$$R(\alpha_q) - R(\alpha) \;\geq\; \begin{cases} \bar{\sigma}/2, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}^2}{2qLU_q}, & \bar{\sigma} < qLU_q \end{cases} . \tag{19}$$

Proof First we prove the upper bound. From the positivity of Q and the definition of σ we obtain

$$R(\alpha_q) - R(\alpha) \;=\; g(\alpha)\cdot(\alpha_q - \alpha) - \frac{1}{2}(\alpha_q - \alpha)\cdot Q(\alpha_q - \alpha) \;\leq\; g(\alpha)\cdot(\alpha_q - \alpha) \;\leq\; \sigma(\alpha|W_q).$$

Now we prove the lower bound. Let

$$\acute{\alpha}_q \in \arg\max_{\gamma \in \mathcal{A}(\alpha, W_q)} g(\alpha)\cdot(\gamma - \alpha)$$

be a solution to the LP problem at (α, W_q) and consider the direction $d_q := \acute{\alpha}_q - \alpha$. The improvement in criterion value for any feasible point in this direction cannot be larger than the improvement for α_q, i.e.

$$R(\alpha_q) - R(\alpha) \geq R(\alpha + \omega d_q) - R(\alpha), \;\; 0 \leq \omega \leq 1. \tag{20}$$

To obtain a lower bound for the right side we start by writing

$$R(\alpha + \omega d_q) - R(\alpha) \;=\; \omega g(\alpha)\cdot d_q - \frac{\omega^2}{2} d_q \cdot Q d_q \;=\; \omega\sigma(\alpha|W_q) - \frac{\omega^2}{2} d_q \cdot Q d_q \;\geq\; \omega\bar{\sigma} - \frac{\omega^2}{2} d_q \cdot Q d_q.$$

Note that d_q has at most q nonzero components determined by the members of W_q. Let Q_q be the $q \times q$ matrix formed from the elements $Q_{ij} : i, j \in W_q$, and let $\lambda_{max}(Q_q)$ and $\mathrm{trace}(Q_q)$ be the largest eigenvalue and the trace of Q_p. Since $Q \geq 0 \Rightarrow Q_p \geq 0$ we have $\lambda_{max}(Q_q) \leq \mathrm{trace}(Q_q) \leq qL$. Thus

$$d_q \cdot Q d_q \;\leq\; \lambda_{max}(Q_q)(d_q \cdot d_q) \;\leq\; qL \sum_{i \in W_q} u_i^2 \;\leq\; qLU_q$$

and therefore

$$R(\alpha + \omega d_q) - R(\alpha) \;\geq\; \omega\bar{\sigma} - \frac{\omega^2}{2} qLU_q.$$

Choosing $\omega \in [0,1]$ to maximize the right side gives

$$\omega^* = \begin{cases} 1, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}}{qLU_q}, & \bar{\sigma} < qLU_q \end{cases}$$

so that

$$R(\alpha + \omega^* d_q) - R(\alpha) \geq \begin{cases} \bar{\sigma} - \frac{qLU_q}{2}, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}^2}{2qLU_q}, & \bar{\sigma} < qLU_q \end{cases} . \tag{21}$$

The first case satisfies

$$\bar{\sigma} - \frac{qLU_q}{2} \;\geq\; \bar{\sigma}/2$$

so that

$$R(\alpha + \omega^* d_q) - R(\alpha) \geq \begin{cases} \bar{\sigma}/2, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}^2}{2qLU_q}, & \bar{\sigma} < qLU_q \end{cases} .$$

Combining this result with (20) gives the result in (19). ■

Proof [Proof of Theorem 5] This proof is a slight modification of the proof in (List and Simon, 2005, Section 3.3) so we describe only the main differences. The basic approach is to obtain an upper bound on the number of iterations by deriving a lower bound on the stepwise improvement. The first difference is based on an idea from the proof of Hush and Scovel (2003, Theorem 5). Let $W_2^m \subseteq W^m$ be a τ–rate certifying pair for α^m. The stepwise improvement with W^m is at least as good as the stepwise improvement with W_2^m and therefore

$$R(\alpha^{m+1}) - R(\alpha^m) \geq R(\acute{\alpha}^{m+1}) - R(\alpha^m) \tag{22}$$

where $\acute{\alpha}^{m+1}$ is a solution to the two–variable QP problem at (α^m, W_2^m). Define

$$\Delta^m := R^* - R(\alpha^m).$$

Since $\sigma(\alpha^m|W_2^m) \geq \tau\Delta^m$ (see Hush and Scovel, 2003; List and Simon, 2005) we can bound the right side of (22) by applying the lower bound in (Lemma 14, Equation (19)) with $q = 2$, $\overline{\sigma} = \tau\Delta^m$, and $U_2 = 2S$ to obtain

$$R(\acute{\alpha}^{m+1}) - R(\alpha^m) \;\geq\; \begin{cases} \frac{\tau\Delta^m}{2}, & \Delta^m \geq \frac{4LS^2}{\tau} \\ \frac{(\tau\Delta^m)^2}{8LS}, & \Delta^m < \frac{4LS^2}{\tau} \end{cases} .$$

Combining this result with (22) and using $R(\alpha^{m+1}) - R(\alpha^m) = \Delta^m - \Delta^{m+1}$ we obtain

$$\Delta^{m+1} \leq \left(1 - \frac{\tau}{2}\right)\Delta^m, \quad \text{when } \Delta^m \geq \frac{4LS^2}{\tau}$$
$$\Delta^{m+1} \leq \Delta^m - \gamma(\Delta^m)^2, \quad \text{when } \Delta^m < \frac{4LS^2}{\tau}$$

where $\gamma = \tau^2/8LS^2$. This is essentially the same result obtained in (List and Simon, 2005, p. 316) except that here we have $4L$ in place of the term qL_{max} in (List and Simon, 2005) where q is the size of the working set W^m and L_{max} is the largest among the eigenvalues of all the principle $q \times q$ submatrices of Q. To complete the proof we follow the steps in (List and Simon, 2005, Section 3.3) until the bottom of page 317 where we retain the (slightly) tighter bound

$$\delta_m \geq \delta_{m_0} + \gamma(m - m_0)$$

where, in our case, $\delta_{m_0} \geq \tau/4LS^2$. This gives a bound on the criterion gap

$$\Delta^m \leq \frac{1}{\delta_{m_0} + \gamma(m - m_0)} \tag{23}$$

which leads to the "-1" term in the second part of our expression for $\acute{m}$ and ensures that the expressions in first and second parts match at the boundary where $\varepsilon = \frac{4LS^2}{\tau}$. ■

Proof [Proof of Theorem 6:] We start by proving the first assertion. To simplify notation we write g as a shorthand for $g(\alpha)$. Since setting $\acute{\alpha} = \alpha$ gives $g \cdot (\acute{\alpha} - \alpha) = 0$ it follows that $\sigma(\alpha|W_n) \geq 0$. Similarly $\sigma(\alpha|W_2) \geq 0$ for all $W_2 \subseteq W_n$. Thus when $\sigma(\alpha|W_n) = 0$ it follows that the assertion is true. Therefore let us assume $\sigma(\alpha|W_n) > 0$.

Let

$$W_2^* \in \arg\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2).$$

We start by deriving an expression for $\sigma(\alpha|W_2^*)$. A two–variable problem with working set $W_2 = \{j,k\}$ satisfies

$$\begin{aligned}
\sigma(\alpha|W_2) &= \sup_{\acute{\alpha} \in \mathcal{A}(\alpha,W_2)} g \cdot (\acute{\alpha} - \alpha) = \sup_{\alpha + d \in \mathcal{A}(\alpha,W_2)} g \cdot d \\
&= \sup_{\substack{d_j = -d_k \\ -\alpha_j \le d_j \le u_j - \alpha_j \\ -\alpha_k \le d_k \le u_k - \alpha_k}} d_j g_j + d_k g_k \\
&= \sup_{\substack{-\alpha_j \le d_j \le u_j - \alpha_j \\ \alpha_k - u_k \le d_j \le \alpha_k}} d_j (g_j - g_k) \\
&= \Delta_{jk}(g_j - g_k)
\end{aligned}$$

where

$$\Delta_{jk} = \begin{cases} \min(u_j - \alpha_j, \alpha_k), & g_j > g_k \\ -\min(\alpha_j, u_k - \alpha_k), & g_j < g_k \\ 0, & g_j = g_k \end{cases}.$$

The expression for $\sigma(\alpha|W_2^*)$ is obtained by maximizing over all pairs,

$$\sigma(\alpha|W_2^*) = \max_{\{j,k\} \subseteq W_n} \Delta_{jk}(g_j - g_k). \tag{24}$$

Now write

$$\sigma(\alpha|W_n) = \sup_{\acute{\alpha} \in \mathcal{A}} g \cdot (\acute{\alpha} - \alpha) = \sup_{\alpha + d \in \mathcal{A}} g \cdot d = \sup_{d \in \mathcal{D}} g \cdot d$$

where

$$\mathcal{D} = \{d : d \cdot 1 = 0, -\alpha_i \le d_i \le u_i - \alpha_i\}.$$

Let d^* be a solution so that

$$\sigma(\alpha|W_n) = g \cdot d^*.$$

The intuition for what follows is that we will (implicitly) decompose d^* into

$$d^* = \overline{d^1} + \ldots + \overline{d^p}$$

such that $p \le n-1$, and for every $i \in \{1,\ldots,p\}$ $\overline{d^i}$ has only two non-zero components and $\alpha + \overline{d^i}$ is feasible at α. Define the index sets

$$I_+ = \{i : d_i^* > 0\}, \quad I_- = \{i : d_i^* < 0\}$$

and write

$$\sigma(\alpha|W_n) = \sum_{i\in I_+} d_i^* g_i + \sum_{i\in I_-} d_i^* g_i.$$

Note that $\sigma(\alpha|W_n) \neq 0$ and $d^* \cdot 1 = 0$ imply that both I_+ and I_- are non–empty. We decompose the right hand side into a sum of two–variable terms by applying the following recursion. Initialize with $m = 0$, $d_i^0 = d_i^*$, $I_+^0 = I_+$, $I_-^0 = I_-$, and

$$h^0 = \sum_{i\in I_+^0} d_i^0 g_i + \sum_{i\in I_-^0} d_i^0 g_i.$$

Then while $h^m \neq 0$ choose an index pair $(j_m, k_m) \in I_+^m \times I_-^m$, define $\delta_{j_m k_m} = \min(d_{j_m}^m, -d_{k_m}^m)$, and define

$$h^{m+1} = \sum_{i\in I_+^{m+1}} d_i^{m+1} g_i + \sum_{i\in I_-^{m+1}} d_i^{m+1} g_i$$

where

$$d_i^{m+1} = \begin{cases} d_i^m - \delta_{j_m k_m}, & i = j_m \\ d_i^m + \delta_{j_m k_m}, & i = k_m \\ d_i^m, & i \neq j_m \text{ or } k_m \end{cases} \tag{25}$$

and

$$I_+^{m+1} = \{i : d_i^{m+1} > 0\}, \quad I_-^{m+1} = \{i : d_i^{m+1} < 0\}.$$

This gives the recursion

$$h^{m+1} = h^m - \delta_{j_m k_m}(g_{j_m} - g_{k_m}).$$

From equation (25) it follows that

$$d^m \in D \quad \Rightarrow \quad d^{m+1} \in D.$$

Thus if $h^{m+1} \neq 0$ then both I_+^{m+1} and I_-^{m+1} are non–empty verifying the existence of an index pair for the next iteration. Furthermore, the definition of $\delta_{j_m k_m}$ implies that either $d_{j_m}^{m+1} = 0$ or $d_{k_m}^{m+1} = 0$ (or both) so that the size of the index sets decreases by at least one at each iteration, i.e.

$$|I_+^{m+1} \cup I_-^{m+1}| \leq |I_+^m \cup I_-^m| - 1.$$

Therefore at least one of the index sets becomes empty after at most $n-1$ iterations. Furthermore $d^m \cdot 1 = 0$ implies that both index sets become empty at the same iteration. Thus this recursion decomposes the original sum as follows

$$\sigma(\alpha|W_n) = h^0 = \delta_{j_1 k_1}(g_{j_1} - g_{k_1}) \; + \; \delta_{j_2 k_2}(g_{j_2} - g_{k_2}) \; + \; \ldots \; + \; \delta_{j_p k_p}(g_{j_p} - g_{k_p}) \tag{26}$$

where $p \leq n-1$. Let q be the index corresponding to the largest of these terms and let $\sigma_{j_q k_q} = \delta_{j_q k_q}(g_{j_q} - g_{k_q})$ be its value. Then (26) implies

$$\sigma_{j_q k_q} \geq \frac{\sigma(\alpha|W_n)}{p} \geq \frac{\sigma(\alpha|W_n)}{n-1}. \tag{27}$$

Furthermore if we combine the fact that $\sigma_{j_q k_q} > 0$ implies $g_{j_q} - g_{k_q} > 0$ with the definitions of d^q and $\Delta_{j_q k_q}$ we obtain

$$\delta_{j_q k_q} = \min(d_{j_q}^q, -d_{k_q}^q) \leq \min(u_{j_q} - \alpha_{j_q}, \alpha_{k_q}) = \Delta_{j_q k_q}.$$

Finally, combining this result with (27) and (24) gives

$$\frac{\sigma(\alpha|W_n)}{n-1} \;\leq\; \sigma_{j_q k_q} \;=\; \delta_{j_q k_q}(g_{j_q} - g_{k_q}) \;\leq\; \Delta_{j_q k_q}(g_{j_q} - g_{k_q}) \;=\; \sigma(\alpha|\{j_q, k_q\}) \leq \sigma(\alpha|W_2^*)$$

which completes the proof of the first assertion.

To prove the second assertion it suffices to give an example of a problem instance and a value $\alpha \in \mathcal{A}$ such that the equality holds. For the primal problem in (1) let $y = (y^+, y^-)$ where $y^+ = (1, \ldots, 1)$ and $y^- = (-1, \ldots, -1)$ are vectors whose lengths are not yet specified. Let u be decomposed into corresponding components so that $u = (u^+, u^-)$. For the corresponding canonical dual problem consider the feasible value $\alpha = (0, u^-)$ (which is the initial value of α in Procedure 1). This gives $g = y$. If $u^+ \cdot 1 = u^- \cdot 1$ then it is easy to verify that

$$\begin{aligned}\sigma(\alpha|W_n) = \sup_{\acute{\alpha} \in \mathcal{A}(\alpha, W_n)} g \cdot (\acute{\alpha} - \alpha) &= y \cdot \big((u^+, 0) - (0, u^-)\big) \\ &= \big(y^+, y^-\big) \cdot \big(u^+, -u^-\big) \\ &= 1 \cdot u = 1 \,,\end{aligned}$$

and

$$\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2) = \max_{(j,k)} \big(\min(u_j - \alpha_j, \alpha_k)(g_j - g_k)\big) = 2\min(u_*^+, u_*^-)$$

where u_*^+ is the largest component value of u^+, and u_*^- is the largest component value of u^-. Thus any problem where $u^+ \cdot 1 = u^- \cdot 1$ and $\min(u_*^+, u_*^-) = \frac{1}{2(n-1)}$ yields the relationship we seek and finishes the proof. For example this condition is satisfied by $u^+ = (1/2)$ and $u^- = \left(\frac{1}{2(n-1)}, \ldots, \frac{1}{2(n-1)}\right)$. ■

Proof [Proof of Corollary 7:] This proof follows directly from the proof of Theorem 5 since by assumption the stepwise improvement of `DECOMP` satisfies (22) and the rest of the proof follows without modification. ■

Proof [Proof of Theorem 8:] Let α_p and α_n be solutions to the QP problem at (α, W_p) and (α, W_n) respectively. Since $R(\alpha_p) \leq R^*$ and $R(\alpha_n) = R^*$ we obtain

$$R(\alpha_p) - R(\alpha) \;\leq\; R^* - R(\alpha) = R(\alpha_n) - R(\alpha).$$

Applying (Lemma 14, Equation (19)) with $q = p$ and $\bar{\sigma} = \sigma(\alpha|W_p)$ on the left, and (Lemma 14, Equation (18)) with $q = n$ on the right gives

$$\frac{\sigma(\alpha|W_p)}{2} \min\left(1, \frac{\sigma(\alpha|W_p)}{pLU_p}\right) \;\leq\; R^* - R(\alpha) \;\leq\; \sigma(\alpha|W_n).$$

If W_p contains a τ–rate certifying pair then $\sigma(\alpha|W_n) \leq \frac{\sigma(\alpha|W_p)}{\tau}$ and the proof is finished. ■

Proof [Proof of Theorem 12:] Use (3) to determine the dual variables $\hat{a}$ and a^0 corresponding to $\hat{\alpha}$ and α^0 respectively. This gives $a^0 = 0$. Let s be the number of nonzero components of $\hat{a}$. Since

$a^0 = 0$ and a *W2* decomposition algorithm can change only two components at each iteration the number of iterations m required to reach $\hat{a}$ satisfies $m \geq s/2$. Furthermore since $\hat{a}_i \leq 1/n$ we obtain $s/n \geq \hat{a} \cdot 1$ and therefore

$$m \;\geq\; \frac{n\hat{a}\cdot 1}{2}\;.$$

Since $\hat{a}$ is an ε–optimal solution

$$-\frac{1}{2}\hat{a}\cdot \mathsf{Q}\hat{a} + \hat{a}\cdot 1 \;\geq\; R^* - \varepsilon$$

and since Q is positive semi–definite this implies $\hat{a}\cdot 1 \;\geq\; R^* - \varepsilon$ and therefore $m \;\geq\; \frac{n(R^*-\varepsilon)}{2}$. ■

Appendix A. Algorithms

A complete algorithm that computes an ε_p–optimal solution to the primal QP problem is provided by the (Procedure 1,Section 2) and Procedures 3–8 in this appendix. Procedure 3 implements a *W2* variant of the Composite–I decomposition algorithm with Stopping Rule 2. Procedure 4 implements Simon's algorithm where the values in (11) are stored in a list of 3–tuples of the form (μ, i, ς) where μ is a value from (11), i is the index of the corresponding component of α, and $\varsigma \in \{+,-\}$ is a symbol indicating the entry type (in particular $\varsigma_l = +$ when $\mu_l = u_{i_l} - \alpha_{i_l}$ and $\varsigma_l = -$ when $\mu_{i_l} = \alpha_{i_l}$). The algorithm scans the ordered list and saves the index pair that maximizes $\sigma(\alpha|\{j,k\})$ as described in Section 2.1. Since this algorithm tracks the indices of the maximum and minimum gradient values it also produces a max–violating pair when it exits the loop. Procedure 5 computes the initial gradient, the initial list M, and an initial upper bound $s^0 = 1$ on the criterion gap $R^* - R(\alpha^0)$. The run time of this procedure is $O(n^2)$ as determined by the gradient computation. Procedure 6 computes the stepwise improvement for the W_{mlp2} and W_{mv} pairs and then updates α according to the pair with the largest improvement. This routine runs in $O(1)$ time. Procedure 7 shows the deletions and insertions required to update the M–list. With the appropriate data structure each of these insert and delete operations can be performed in $O(\log n)$ time. Procedure 8 implements the $O(n \log n)$ algorithm described in Section 2.3.

References

Jose L. Balcazar, Yang Dai, and Osamu Watanabe. Provably fast training algorithms for support vector machines. In *Proceedings of the 1st International Conference on Data Mining ICDM*, pages 43–50, 2001. URL `citeseer.ist.psu.edu/590348.html`.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

C.C. Chang, C.W. Hsu, and C.J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM : a library for support vector machines, 2001.

Procedure 3 The Composite–I Decomposition Algorithm.

Decomposition$(Q, w, c, u, \varepsilon, \alpha^0)$

$(g^0, M^0, s^0) \leftarrow$ Initialize (Q, w, u, α^0)
$m \leftarrow 0$
repeat
  $(W^m_{mlp2}, W^m_{mv}, \sigma^m) \leftarrow$ Simon(g^m, M^m)
  if $(\sigma^m = 0)$ **then**
    Return(α^m, g^m)
  end if
  $(\alpha^{m+1}, \delta^m_R, W^m) \leftarrow$ CompositeUpdate$(\alpha^m, g^m, Q, W^m_{mlp2}, W^m_{mv})$
  $g^{m+1} \leftarrow g^m - Q(\alpha^{m+1} - \alpha^m)$
  $M^{m+1} \leftarrow$ UpdateMlist$(M^m, W^m, \alpha^m, \alpha^{m+1})$
  $s^{m+1} \leftarrow \min((n-1)\sigma^m, s^m) - \delta^m_R$
  $m \leftarrow m+1$
until $\left(s^m \le \varepsilon\right)$
Return(α^m, g^m)

Procedure 4 This routine uses Simon's algorithm to compute a max–lp2 pair W_{mlp2}. It also computes and returns a max–violating pair W_{mv} and the value $\sigma^* = \sigma(\alpha|W_{mlp2})$. It assumes that M is an sorted list arranged in nonincreasing order by the value of first component.

Simon(g, M) $\quad \{ M = \left[(\mu, i, \varsigma)_1, (\mu, i, \varsigma)_2, ..., (\mu, i, \varsigma)_{2n}\right] \}$

$i_{max} \leftarrow 0, \quad i_{min} \leftarrow 0, \quad g_{max} \leftarrow -\infty, \quad g_{min} \leftarrow \infty, \quad \sigma^* \leftarrow 0, \quad W_{mlp2} \leftarrow \emptyset$
$k \leftarrow 1$
while $(\mu_k > 0)$ **do**
  if $((\varsigma_k = +1)$ and $(g_{i_k} > g_{max}))$ **then**
    $g_{max} \leftarrow g_{i_k}, \quad i_{max} \leftarrow i_k$
    if $(\mu_k(g_{max} - g_{min}) > \sigma^*)$ **then**
      $W_{mlp2} \leftarrow \{i_{max}, i_{min}\}, \quad \sigma^* \leftarrow \mu_k(g_{max} - g_{min})$
    end if
  else if $((\varsigma_k = -1)$ and $(g_{i_k} < g_{min}))$ **then**
    $g_{min} \leftarrow g_{i_k}, \quad i_{min} \leftarrow i_k$
    if $(\mu_k(g_{max} - g_{min}) > \sigma^*)$ **then**
      $W_{mlp2} \leftarrow \{i_{max}, i_{min}\}, \quad \sigma^* \leftarrow \mu_k(g_{max} - g_{min})$
    end if
  end if
  $k \leftarrow k+1$
end while
$W_{mv} \leftarrow \{i_{max}, i_{min}\}$
Return$(W_{mlp2}, W_{mv}, \sigma^*)$

Procedure 5 This routine accepts a feasible value α and computes the corresponding gradient g, a list M of 3–tuples (μ, i, ς) sorted by μ, and a trivial bound $s = 1$ on $R^* - R(\alpha)$.

Initialize(Q, w, u, α)

$g \leftarrow -Q\alpha + w$
$M \leftarrow \emptyset$
for $(i = 1, ..., n)$ **do**
  $M \leftarrow$ Insert$(M, (\alpha_i, i, -))$
  $M \leftarrow$ Insert$(M, (u_i - \alpha_i, i, +))$
end for
$s \leftarrow 1$
Return(g, M, s)

Procedure 6 This routine computes the stepwise improvements for a max–lp2 pair W_{mlp2} and a max–violating pair W_{mv}, and then updates α using the pair with the largest stepwise improvement. It returns the new value of α, and the corresponding stepwise improvement value and index pair.

CompositeUpdate$(\alpha^{old}, g, Q, W_{mlp2}, W_{mv})$

$\{i_1, i_2\} \leftarrow W_{mlp2}$
$\delta_g \leftarrow g_{i_1} - g_{i_2}, \quad q \leftarrow Q_{i_1 i_1} + Q_{i_2 i_2} - 2Q_{i_1 i_2}, \quad \Delta_{mlp2} = \min\left(u_{i_1} - \alpha_{i_1}^{old}, \alpha_{i_2}^{old}\right)$
if $(\delta_g > q\Delta_{mlp2})$ **then**
  $\delta_{mlp2} \leftarrow \Delta_{mlp2}\left(\delta_g - \frac{q\Delta_{mlp2}}{2}\right)$
else
  $\delta_{mlp2} \leftarrow \frac{\delta_g^2}{2q}, \quad \Delta_{mlp2} \leftarrow \frac{\delta_g}{q}$
end if

$\{j_1, j_2\} \leftarrow W_{mv}$
$\delta_g \leftarrow g_{j_1} - g_{j_2}, \quad q \leftarrow Q_{j_1 j_1} + Q_{j_2 j_2} - 2Q_{j_1 j_2}, \quad \Delta_{mv} = \min\left(u_{j_1} - \alpha_{j_1}^{old}, \alpha_{j_2}^{old}\right)$
if $(\delta_g > q\Delta_{mv})$ **then**
  $\delta_{mv} \leftarrow \Delta_{mv}\left(\delta_g - \frac{q\Delta_{mv}}{2}\right)$
else
  $\delta_{mv} \leftarrow \frac{\delta_g^2}{2q}, \quad \Delta_{mv} \leftarrow \frac{\delta_g}{q}$
end if

if $(\delta_{mlp2} > \delta_{mv})$ **then**
  $\alpha_{i_1}^{new} \leftarrow \alpha_{i_1}^{old} + \Delta_{mlp2}, \quad \alpha_{i_2}^{new} \leftarrow \alpha_{i_2}^{old} - \Delta_{mlp2}$
  Return$(\alpha^{new}, \delta_{mlp2}, W_{mlp2})$
else
  $\alpha_{j_1}^{new} \leftarrow \alpha_{j_1}^{old} + \Delta_{mv}, \quad \alpha_{j_2}^{new} \leftarrow \alpha_{j_2}^{old} - \Delta_{mv}$
  Return$(\alpha^{new}, \delta_{mv}, W_{mv})$
end if

Procedure 7 This routine updates the sorted list M.

UpdateMlist $(M, W, \alpha^{old}, \alpha^{new})$

$\{i_1, i_2\} \leftarrow W$
$M \leftarrow \texttt{Delete}\big(M, (\alpha^{old}_{i_1}, i_1, -)\big)$
$M \leftarrow \texttt{Delete}\big(M, (u_{i_1} - \alpha^{old}_{i_1}, i_1, +)\big)$
$M \leftarrow \texttt{Delete}\big(M, (\alpha^{old}_{i_2}, i_2, -)\big)$
$M \leftarrow \texttt{Delete}\big(M, (u_{i_2} - \alpha^{old}_{i_2}, i_2, +)\big)$
$M \leftarrow \texttt{Insert}\big(M, (\alpha^{new}_{i_1}, i_1, -)\big)$
$M \leftarrow \texttt{Insert}\big(M, (u_{i_1} - \alpha^{new}_{i_1}, i_1, +)\big)$
$M \leftarrow \texttt{Insert}\big(M, (\alpha^{new}_{i_2}, i_2, -)\big)$
$M \leftarrow \texttt{Insert}\big(M, (u_{i_2} - \alpha^{new}_{i_2}, i_2, +)\big)$
Return(M)

Procedure 8 This routine determines the offset parameter according to Theorem 2. Note that the input g is the gradient vector from the canonical dual solution.

Offset(g, y, u)

$s^+ \leftarrow \sum_{i:y_i=1} u_i, \quad s^- \leftarrow 0$
$\big((\bar{g}_1, \bar{y}_1, \bar{u}_1), \ldots, (\bar{g}_n, \bar{y}_n, \bar{u}_n)\big) \leftarrow \texttt{SortIncreasing}\big((g_1, y_1, u_1), \ldots, (g_n, y_n, u_n)\big)$
$L \leftarrow \sum_{i:\bar{y}_i=1} \bar{u}_i (\bar{g}_i - \bar{g}_1)$
$L^* \leftarrow L, \quad b \leftarrow \bar{g}_1$
for $(i = 1, \ldots, n-1)$ **do**
  if $(\bar{y}_i = 1)$ **then**
    $s^+ \leftarrow s^+ - \bar{u}_i$
  else
    $s^- \leftarrow s^- + \bar{u}_i$
  end if
  $L \leftarrow L - (\bar{g}_{i+1} - \bar{g}_i)(s^+ - s^-)$
  if $(L < L^*)$ **then**
    $L^* \leftarrow L, \quad b \leftarrow \bar{g}_{i+1}$
  end if
end for
Return(b)

P.-H. Chen, R.-E. Fan, and C.-J. Lin. Training support vector machines via SMO–type decomposition methods. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory*, pages 45–62, 2005.

P.-H. Chen, R.-E. Fan, and C.-J. Lin. A study on SMO-type decomposition methods for support vector machines. Technical report, 2006. URL http://www.csie.ntu.edu.tw/∼cjlin/papers.html. to appear in IEEE Transactions on Neural Networks.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Canbridge ; United Kingdom, 1st edition, 2000.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

C.-W. Hsu and C.-J. Lin. A simple decomposition algorithm for support vector machines. *Machine Learning*, 46:291–314, 2002.

D. Hush and C. Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003.

D. Hush, C. Scovel, and I. Steinwart. Stability of unstable learning algorithms. Technical report, Los Alamos National Laboratory LA-UR-03-4845, 2003. URL http://wwwc3.lanl.gov/ml/pubs_ml.shtml. submitted for publication.

D. Hush, C. Scovel, and I. Steinwart. Polynomial time algorithms for computing approximate SVM solutions with guaranteed accuracy. Technical report, Los Alamos National Laboratory LA-UR 05-7738, 2005. URL http://wwwc3.lanl.gov/ml/pubs_ml.shtml.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, 1998.

S.S. Keerthi and E.G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.

S.S. Keerthi and C.J. Ong. On the role of the threshold parameter in SVM training algorithms. Control Division Technical Report CD-00-09, Dept. of Mechanical and Production Engineering, National University of Singapore, 2000.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11:637–649, 2000.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

D. Lai, N. Mani, and M. Palaniswami. A new method to select working sets for faster training for support vector machines. Technical Report MESCE–30–2003, Dept. Electrical and Computer Systems Engineering, Monash University, Australia, 2003.

P. Laskov. Feasible direction decomposition algorithms for training support vector machines. *Machine Learning*, 46(1–3):315–349, 2002.

S.-P. Liao, H.-T. Lin, and C.-J. Lin. A note on the decomposition methods for support vector regression. *Neural Computation*, 14:1267–1281, 2002.

C.-J. Lin. Linear convergence of a decomposition method for support vector machines. Technical Report, 2001a. URL http://www.csie.ntu.edu.tw/∼cjlin/papers.html.

C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001b.

C.-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13:1045–1052, 2002a.

C.-J. Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13:248–250, 2002b.

N. List and H.U. Simon. A general convergence theorem for the desomposition method. In J. Shawe-Taylor and Y. Singer, editors, *17th Annual Conference on Learning Theory, COLT 2004, volume 3120 of Lecture Notes in Computer Science*, pages 363–377, 2004.

N. List and H.U. Simon. General polynomial time decomposition algorithms. In P. Auer and R. Meir, editors, *18th Annual Conference on Learning Theory, COLT 2005*, pages 308–322, 2005.

O.L. Mangasarian and D.R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001.

O.L. Mangasarian and D.R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.

E.E. Osuna, R. Freund, and F. Girosi. Support vector machines: training and applications. Technical Report AIM-1602, MIT, 1997.

J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 41–64. MIT Press, Cambridge, MA, 1998.

B. Schölkopf, J.C. Platt, J. Shawe-Taylor, and A.J. Smola. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

C. Scovel, D. Hush, and I. Steinwart. Approximate duality. Technical report, Los Alamos National Laboratory LA-UR 05-6766, 2005a. URL http://wwwc3.lanl.gov/ml/pubs_ml.shtml. to appear in Journal of Optimization Theory and Applications.

C. Scovel, D. Hush, and I. Steinwart. Learning rates for density level detection. *Analysis and Applications*, 3(4):356–371, 2005b.

H.U. Simon. On the complexity of working set selection. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory*, 2004. URL http://eprints.pascal-network.org/archive/00000125/.

I. Steinwart and C. Scovel. Fast rates for support vector machines. In P. Auer and R. Meir, editors, *18th Annual Conference on Learning Theory, COLT 2005*, pages 279–294, 2005.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. Technical report, Los Alamos National Laboratory LA-UR 04-8796, 2004. URL http://www.c3.lanl.gov/∼ingo/publications/pubs.shtml. submitted to Annals of Statistics (2004).

I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6:211–232, 2005.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, Inc., New York, NY, 1998.

Journal of Machine Learning Research 7 (2006) 771–791 Submitted 5/05; Published 5/06

Policy Gradient in Continuous Time

Rémi Munos REMI.MUNOS@POLYTECHNIQUE.FR
Centre de Mathématiques Appliquées
Ecole Polytechnique
91128 Palaiseau, France

Editor: Michael Littman

Abstract

Policy search is a method for approximately solving an optimal control problem by performing a parametric optimization search in a given class of parameterized policies. In order to process a local optimization technique, such as a gradient method, we wish to evaluate the sensitivity of the performance measure with respect to the policy parameters, the so-called *policy gradient*. This paper is concerned with the estimation of the policy gradient for continuous-time, deterministic state dynamics, in a *reinforcement learning* framework, that is, when the decision maker does not have a model of the state dynamics.

We show that usual likelihood ratio methods used in discrete-time, fail to proceed the gradient because they are subject to variance explosion when the discretization time-step decreases to 0. We describe an alternative approach based on the approximation of the pathwise derivative, which leads to a policy gradient estimate that converges almost surely to the true gradient when the time-step tends to 0. The underlying idea starts with the derivation of an explicit representation of the policy gradient using pathwise derivation. This derivation makes use of the knowledge of the state dynamics. Then, in order to estimate the gradient from the observable data only, we use a stochastic policy to discretize the continuous deterministic system into a stochastic discrete process, which enables to replace the unknown coefficients by quantities that solely depend on known data. We prove the almost sure convergence of this estimate to the true policy gradient when the discretization time-step goes to zero.

The method is illustrated on two target problems, in discrete and continuous control spaces.

Keywords: optimal control, reinforcement learning, policy search, sensitivity analysis, parametric optimization, gradient estimate, likelihood ratio method, pathwise derivation

1. Introduction and Statement of the Problem

We consider an optimal control problem with continuous state $(x_t \in \mathbb{R}^d)_{t\geq 0}$ whose state dynamics is defined according to the controlled differential equation:

$$\frac{dx_t}{dt} = f(x_t, u_t), \tag{1}$$

where the control $(u_t)_{t\geq 0}$ is a Lebesgue measurable function with values in a control space U. Note that the state-dynamics f may also depend on time, but we omit this dependency in the notation, for simplicity. We intend to maximize a functional J that depends on the trajectory $(x_t)_{0\leq t\leq T}$ over a finite-time horizon $T > 0$. For simplicity, in the paper, we illustrate the case of a terminal reward

only:

$$J(x;(u_t)_{t\geq 0}) := r(x_T), \tag{2}$$

where $r : \mathbb{R}^d \to \mathbb{R}$ is the reward function. Extension to the case of general functional of the kind

$$J(x;(u_t)_{t\geq 0}) = \int_0^T r(t,x_t)dt + R(x_T), \tag{3}$$

with r and R being current and terminal reward functions, would easily follow, as indicated in Remark 1.

The optimal control problem of finding a control $(u_t)_{t\geq 0}$ that maximizes the functional is replaced by a parametric optimization problem for which we search for a good feed-back control law in a given class of parameterized policies $\{\pi_\alpha : [0,T] \times \mathbb{R}^d \to U\}_\alpha$, where $\alpha \in \mathbb{R}^m$ is the parameter. The control $u_t \in U$ (or action) at time t is $u_t = \pi_\alpha(t,x_t)$, and we may write the dynamics of the resulting feed-back system as

$$\frac{dx_t}{dt} = f_\alpha(x_t), \tag{4}$$

where $f_\alpha(x_t) := f(x,\pi_\alpha(t,x))$. In the paper, we will make the assumption that f_α is $\mathcal{C}^2$, with bounded derivatives. Let us define the **performance measure**

$$V(\alpha) := J(x;\pi_\alpha(t,x_t)_{t\geq 0}),$$

where its dependency with respect to (w.r.t.) the parameter α is emphasized. One may also consider an average performance measure according to some distribution μ for the initial state: $V(\alpha) := \mathbb{E}[J(x;\pi_\alpha(t,x_t)_{t\geq 0})|x \sim \mu]$.

In order to find a local maximum of $V(\alpha)$, one may perform a local search, such as a gradient ascent method

$$\alpha \leftarrow \alpha + \eta \nabla_\alpha V(\alpha), \tag{5}$$

with an adequate step η (see for example (Polyak, 1987; Kushner and Yin, 1997)). The computation of the gradient $\nabla_\alpha V(\alpha)$ is the object of this paper.

A first method would be to approximate the gradient by a finite-difference quotient for each of the m components of the parameter:

$$\partial_{\alpha_i} V(\alpha) \simeq \frac{V(\alpha+\varepsilon e_i) - V(\alpha)}{\varepsilon},$$

for some small value of ε (we use the notation ∂_α instead of ∇_α to indicate that it is a single-dimensional derivative). This finite-difference method requires the simulation of $m+1$ trajectories to compute an approximation of the true gradient. When the number of parameters is large, this may be computationally expensive. However, this simple method may be efficient if the number of parameters is relatively small.

In the rest of the paper we will not consider this approach, and will aim at computing the gradient using one trajectory only.

Pathwise estimation of the gradient. We now illustrate that if the decision-maker has access to a model of the state dynamics, then a pathwise derivation would directly lead to the policy gradient. Indeed, let us define the gradient of the state with respect to the parameter: $z_t := \nabla_\alpha x_t$ (i.e. z_t is defined as a $d \times m$-matrix whose (i,j)-component is the derivative of the ith component of x_t w.r.t. α_j). Our smoothness assumption on f_α allows to differentiate the state dynamics (4) w.r.t. α, which provides the dynamics on (z_t):

$$\frac{dz_t}{dt} = \nabla_\alpha f_\alpha(x_t) + \nabla_x f_\alpha(x_t) z_t, \tag{6}$$

where the coefficients $\nabla_\alpha f_\alpha$ and $\nabla_x f_\alpha$ are, respectively, the derivatives of f w.r.t. the parameter (matrix of size $d \times m$) and the state (matrix of size $d \times d$). The initial condition for z is $z_0 = 0$. When the reward function r is smooth (i.e. continuously differentiable), one may apply a pathwise differentiation to derive a gradient formula (see e.g. (Bensoussan, 1988) or (Yang and Kushner, 1991) for an extension to the stochastic case):

$$\nabla_\alpha V(\alpha) = \nabla_x r(x_T) z_T. \tag{7}$$

Remark 1 *In the more general setting of a functional (3), the gradient is deduced (by linearity) from the above formula:*

$$\nabla_\alpha V(\alpha) = \int_0^T \nabla_x r(t, x_t) z_t \, dt + \nabla_x R(x_T) z_T.$$

What is known from the agent? The decision maker (call it the agent) that intends to design a good controller for the dynamical system may or may not know a model of the state dynamics f. In case the dynamics is known, the state gradient $z_t = \nabla_\alpha x_t$ may be computed from (6) along the trajectory and the gradient of the performance measure w.r.t. the parameter α is deduced at time T from (7), which allows to perform the gradient ascent step (5).

However, in this paper we consider a *Reinforcement Learning* (Sutton and Barto, 1998) setting in which the state dynamics is unknown from the agent, but we still assume that the state is fully observable. The agent knows only the response of the system to its control. To be more precise, the available information to the agent at time t is its own control policy π_α and the trajectory $(x_s)_{0 \le s \le t}$ up to time t. At time T, the agent receives the reward $r(x_T)$ and, in this paper, we assume that the gradient $\nabla r(x_T)$ is available to the agent.

From this point of view, it seems impossible to derive the state gradient z_t from (6), since $\nabla_\alpha f$ and $\nabla_x f$ are unknown. The term $\nabla_x f(x_t)$ may be approximated by a least squares method from the observation of past states $(x_s)_{s \le t}$, as this will be explained later on in subsection 3.2. However the term $\nabla_\alpha f(x_t)$ cannot be calculated analogously.

In this paper, we introduce the idea of using stochastic policies to approximate the state (x_t) and the state gradient (z_t) by discrete-time stochastic processes (X_t^Δ) and (Z_t^Δ) (with Δ being some discretization time-step). We show how Z_t^Δ can be computed without the knowledge of $\nabla_\alpha f$, but only from information available to the agent.

We prove the convergence (with probability one) of the gradient estimate $\nabla_x r(X_T^\Delta) Z_T^\Delta$ derived from the stochastic processes to $\nabla_\alpha V(\alpha)$ when $\Delta \to 0$. Here, almost sure convergence is obtained using the *concentration of measure phenomenon* (Talagrand, 1996; Ledoux, 2001).

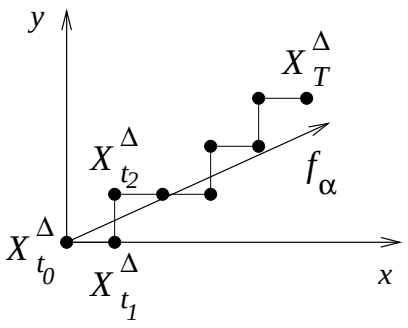


Figure 1: A trajectory $(X^{\Delta}_{t_n})_{0\leq n\leq N}$ and the state dynamics vector f_α of the continuous process $(x_t)_{0\leq t\leq T}$.

Likelihood ratio method? It is worth mentioning that this strong convergence result contrasts with the usual *likelihood ratio method* (also called *score method*) in discrete time (see e.g. (Reiman and Weiss, 1986; Glynn, 1987) or more recently in the reinforcement learning literature (Williams, 1992; Sutton et al., 2000; Baxter and Bartlett, 2001; Marbach and Tsitsiklis, 2003)) for which the policy gradient estimate is subject to variance explosion when the discretization time-step Δ tends to 0. The intuitive reason for that problem lies in the fact that the number of decisions before getting the reward grows to infinity when $\Delta \to 0$ (the variance of likelihood ratio estimates being usually linear with the number of decisions).

Let us illustrate this problem on a simple 2 dimensional process. Consider the deterministic continuous process $(x_t)_{0\leq t\leq 1}$ defined by the state dynamics:

$$\frac{dx_t}{dt} = f_\alpha := \begin{pmatrix} \alpha \\ 1-\alpha \end{pmatrix}, \tag{8}$$

$(0 < \alpha < 1)$ with initial condition $x_0 = (0\,0)'$ (where $'$ denotes the transpose operator). The performance measure $V(\alpha)$ is the reward at the terminal state at time $T = 1$, with the reward function being the first coordinate of the state $r((xy)') := x$. Thus $V(\alpha) = r(x_{T=1}) = \alpha$ and its derivative is $\nabla_\alpha V(\alpha) = 1$.

Let $(X^{\Delta}_{t_n})_{0\leq n\leq N} \in \mathbb{R}^2$ be a discrete time stochastic process (the discrete times being $\{t_n = n\Delta\}_{n=0\ldots N}$ with the discretization time-step $\Delta = 1/N$) that starts from initial state $X^{\Delta}_0 = x_0 = (0\,0)'$ and makes N random moves of length Δ towards the right (action u_1) or the top (action u_2) (see Figure 1) according to the stochastic policy (i.e., the probability of choosing the actions in each state x) $\pi_\alpha(u_1|x) = \alpha$, $\pi_\alpha(u_2|x) = 1-\alpha$.

The process is thus defined according to the dynamics:

$$X^{\Delta}_{t_{n+1}} = X^{\Delta}_{t_n} + \begin{pmatrix} U_n \\ 1-U_n \end{pmatrix} \Delta, \tag{9}$$

where $(U_n)_{0\leq n<N}$ are N independent Bernoulli random variables that equal 1 with probability α and 0 with probability $1-\alpha$. The stochastic discrete process (X^{Δ}_t) is consistent with the deterministic continuous one (x_t) in the sense that the jump average direction of the former equals the state

dynamics vector of the latter:

$$\mathbb{E}\Big[\frac{X_{t_{n+1}}-X_{t_n}}{\Delta}|X_{t_n}=x\Big]=\pi_\alpha(u_1,x)\begin{pmatrix}1\\0\end{pmatrix}+\pi_\alpha(u_2,x)\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}\alpha\\1-\alpha\end{pmatrix}.$$

Thus, when the discretization time-step Δ tends to 0, the process (X_t^Δ) converges almost surely to (x_t) (this statement will be proved in Section 2).

Now, write $V^\Delta(\alpha)$ the performance measure of the discrete process, taken as the expected reward at the terminal state: $V^\Delta(\alpha):=\mathbb{E}[r(X_1^\Delta)]=\frac{1}{N}\sum_{n=0}^{N-1}U_n$. The likelihood ratio estimate $g(\Delta)$ of the gradient $\nabla_\alpha V^\Delta(\alpha)=\mathbb{E}[g(\Delta)]$ is

$$\begin{aligned}g(\Delta) &= r(X_1^\Delta)\sum_{n=0}^{N-1}\frac{\nabla_\alpha\pi_\alpha(u_{t_n}|X_{t_n}^\Delta)}{\pi_\alpha(u_{t_n}|X_{t_n}^\Delta)}\\ &= \Big(\frac{1}{N}\sum_{n=0}^{N-1}U_n\Big)\sum_{n=0}^{N-1}\Big(\frac{U_n}{\alpha}-\frac{1-U_n}{1-\alpha}\Big).\end{aligned}\tag{10}$$

The expectation and variance of this estimate are given now (a proof is provided in Appendix A).

Proposition 2 *The expectation and variance of the estimate (10) are*

$$\begin{aligned}\mathbb{E}\big[g(\Delta)\big] &= 1,\\ Var\big[g(\Delta)\big] &= \frac{1-5(1-\alpha)+(2-3\alpha)\alpha N+\alpha^2N^2}{\alpha(1-\alpha)N}.\end{aligned}\tag{11}$$

Thus $g(\Delta)$ is an unbiased estimated of the true gradient $\nabla_\alpha V(\alpha)=1$. However we notice that the dominant term (when N is large) of the variance is $\frac{\alpha}{1-\alpha}N$, with N being the number of decisions before getting the reward, which grows to infinity when the discretization time-step $\Delta=1/N$ tends to 0. Therefore it is impossible to use this likelihood ratio estimate whenever the time discretization is too fine. In contrast, the gradient estimate introduced in this paper has a variance that decreases to 0 when Δ tends to 0 (this will be illustrated on this same example in subsection 3.4).

Outline of the paper. The paper is organized as follows: in Section 2, we state a general approximation result of a continuous deterministic process by a consistent stochastic discrete process and apply it to prove the convergence of the discretized state and state gradient processes when using a stochastic policy. In Section 3, we establish the convergence of the policy gradient estimate and describe a reinforcement learning algorithm that replaces the unknown coefficients about the state dynamics by information available to the agent. In the last Section, we illustrate the method on two (6 dimensional) target problems in both a discrete and a continuous control space cases. All proofs are in the Appendices.

2. Discretized Stochastic Processes

In this section, we start with a general result for approximating a deterministic continuous process by a stochastic discrete one. This is subsequently applied to the convergence analysis of processes (the state X_t^Δ and the state gradient Z_t^Δ) related to the introduction of stochastic policies.

2.1 A General Convergence Result

Let $(x_t)_{0\leq t\leq T}$ be a deterministic continuous process defined by some dynamics

$$\frac{dx_t}{dt} = f(x_t)$$

with some initial condition x_0. We assume that f is of class $\mathcal{C}^2$ with bounded derivatives. The following result state the almost sure convergence of a consistent discrete stochastic process.

Theorem 3 *Let $\Delta = T/N$ be a discretization time-step (with N being the number of steps) and write $\{t_n = n\Delta\}_{0\leq n\leq N}$ the discrete times. Let $(U_{t_n})_{0\leq n<N}$ be a sequence of independent random variables with values in a set U. We define a discrete stochastic process $(X^\Delta_{t_n})_{0\leq n\leq N}$, starting at $X^\Delta_0 = x_0$, according to some discrete state dynamics $f^\Delta : \mathbb{R}^d \times U \to \mathbb{R}^d$, assumed to be bounded: for $t \in \{t_n\}_{0\leq n<N}$,*

$$X^\Delta_{t+\Delta} = X^\Delta_t + f^\Delta(X^\Delta_t, U_t). \tag{12}$$

If f^Δ satisfies the consistency property*:*

$$\mathbb{E}[f^\Delta(x, U_t)] = f(x)\Delta + o(\Delta), \tag{13}$$

and the following bounding condition:

$$f^\Delta = O(\Delta), \tag{14}$$

(where the notation $O(\cdot)$ is to be understood in the sense uniformly w.r.t. the variable of f^Δ) then, the random variable X^Δ_T converges almost surely to (the deterministic) x_T when $\Delta \to 0$. We write

$$\lim_{\Delta\to 0} X^\Delta_T = x_T, \text{ with probability } 1.$$

Appendix B gives a proof of this result. Note that a weaker convergence result (i.e. convergence in probability) may be obtained from general results in approximation of diffusion processes by Markov chains (Kloeden and Platen, 1995). Here, almost sure convergence is obtained using the *concentration of measure phenomenon* (Talagrand, 1996; Ledoux, 2001), detailed in Appendix B.

Remark 4 *If we assume a slightly better consistency error of $O(\Delta^2)$ instead of $o(\Delta)$ in (13), then we may prove (straightforwardly from the Appendix) that $\mathbb{E}[X^\Delta_T] = x_T + O(\Delta)$ and $\mathbb{E}[||X^\Delta_T - x_T||^2] = O(\Delta)$.*

2.2 Discretization of the State

Let us go back to our initial control problem (1). We consider the case of a finite control space U (extension to a continuous control space is straightforward and is detailed in subsection 3.5). Let π_α be a **stochastic policy** , i.e. $\pi_\alpha(u|t,x)$ denotes the probability of choosing action $u \in U$ at time t in state x. We write $u \sim \pi_\alpha(\cdot|t,x)$ a random choice of an action u according to such a policy.

Now, we define the **stochastic discrete state process** $(\mathbf{X^\Delta_{t_n}})_{\mathbf{0\leq n\leq N}}$ (where we use the same notations for the time-steps (t_n) as in the previous subsection), starting at a state $X^\Delta_0 = x$, as follows:

At time $t \in \{(t_n)_{0\leq n<N}\}$, we select an action $u_t \sim \pi_\alpha(\cdot|t, X^\Delta_t)$. Then, $X^\Delta_{t+\Delta}$ is the state at time $t+\Delta$ resulting from keeping the action u_t constant for a period of time Δ. We write:

$$\left\{ \begin{array}{rcl} u_t & \sim & \pi_\alpha(\cdot|t, X^\Delta_t) \\ X^\Delta_{t+\Delta} & := & X^\Delta_t + f^\Delta(X^\Delta_t, u_t) \end{array} \right. \tag{15}$$

where $f^\Delta(x,u)$ represents the jump in the state resulting from the state dynamics (1) with initial condition $x_0 = x$, using a constant control u for a period of time Δ.

The next proposition states the convergence of the discrete stochastic process (X_t^Δ) to the continuous deterministic one (x_t).

Proposition 5 ***Convergence of the discrete state process*** $(\mathbf{X_t^\Delta})$. *When the discretization time-step $\Delta \to 0$, the random variable X_T^Δ converges almost surely to the state x_T defined according to the state dynamics (4) with*

$$f_\alpha(x) := \sum_{u \in U} \pi_\alpha(u|t,x) f(x,u).$$

and initial condition $x_0 = x$.

Proof This is an immediate consequence of Theorem 3 with the discrete state dynamics $f^\Delta(x,u)$. From Taylor's formula,

$$f^\Delta(x,u_t) = f(x,u_t)\Delta + O(\Delta^2),$$

to derive the property on the average jumps:

$$\mathbb{E}[f^\Delta(x,u_t)] \;=\; \sum_{u \in U} \pi_\alpha(u|t,x) f(x,u)\Delta + O(\Delta^2) = f_\alpha(x)\Delta + O(\Delta^2),$$

and the consistency conditions (13) holds, as well as the bound on the jumps (14). ■

2.3 Discretization of the State Gradient

Now, we build an approximation of the state gradient $z_t = \nabla_\alpha x_t$. We define the **stochastic discrete state gradient process** $(\mathbf{Z_{t_n}^\Delta})_{\mathbf{0 \le n \le N}}$, starting with $Z_0^\Delta = 0$, as follows:

At time $t \in \{(t_n)_{0 \le n < N}\}$, let (u_t) and (X_t^Δ) be defined according to (15). Then define

$$Z_{t+\Delta}^\Delta := Z_t^\Delta + f(X_t^\Delta, u_t)\big[l_\alpha(t, X_t^\Delta, u_t)' + l_x(t, X_t^\Delta, u_t)' Z_t^\Delta\big]\Delta + \nabla_x f(X_t^\Delta, u_t) Z_t^\Delta \Delta, \tag{16}$$

where

$$l_\alpha(t,x,u) := \frac{\nabla_\alpha \pi_\alpha(u|t,x)}{\pi_\alpha(u|t,x)} \text{ and } l_x(t,x,u) := \frac{\nabla_x \pi_\alpha(u|t,x)}{\pi_\alpha(u|t,x)}$$

are the likelihood ratios of π_α w.r.t. α and x (defined as vectors of size m and d respectively).

Proposition 6 ***Convergence of the discrete state gradient process*** $(\mathbf{Z_T^\Delta})$***:***

The random variable Z_T^Δ converges almost surely to z_T when $\Delta \to 0$.

Proof The discrete state dynamics (12) for (Z_t^Δ) is defined by the right hand side of (16). Now, from the property

$$\begin{aligned}\mathbb{E}[Z_{t+\Delta}^\Delta - Z_t^\Delta | X_t^\Delta = x, Z_t^\Delta = z] \;&=\; \sum_{u \in U} \pi_\alpha(u|t,x)\Big\{ f(x,u)[l_\alpha(t,x,u)' + l_x(t,x,u)' z] \\ &\qquad + \nabla_x f(x,u) z \Big\}\Delta \\ &=\; \big[\nabla_\alpha f_\alpha(x) + \nabla_x f_\alpha(x) z\big]\Delta,\end{aligned}$$

we deduce that the coupled process (X_t^Δ, Z_t^Δ) is consistent with (x_t, z_t) in the sense of (13):

$$\mathbb{E}\left[\begin{pmatrix} X_{t+\Delta}^\Delta \\ Z_{t+\Delta}^\Delta \end{pmatrix} - \begin{pmatrix} X_t^\Delta \\ Z_t^\Delta \end{pmatrix} \Big| \begin{pmatrix} X_t^\Delta \\ Z_t^\Delta \end{pmatrix} = \begin{pmatrix} x \\ z \end{pmatrix}\right] = \begin{pmatrix} f_\alpha(x) \\ \nabla_\alpha f_\alpha(x) + \nabla_x f_\alpha(x) z \end{pmatrix} \Delta + o(\Delta) \tag{17}$$

and $X_{t+\Delta}^\Delta - X_t^\Delta = O(\Delta)$ and $Z_{t+\Delta}^\Delta - Z_t^\Delta = O(\Delta)$. Thus, as a consequence of Theorem 3, the random variable Z_T^Δ converges almost surely to z_T when $\Delta \to 0$. ■

3. Model-Free Reinforcement Learning Algorithm

We show how to use the approximation results of the previous section to design a model-free reinforcement learning algorithm for estimating the policy gradient $\nabla_\alpha V(\alpha)$ using one trajectory only. First, we state the convergence of the policy gradient estimate computed from the discretized process, then show how to approximate the unknown coefficient $\nabla_x f$ using least-squares regression from the observed trajectory, and finally describe the reinforcement learning algorithm.

3.1 Convergence of the Policy Gradient Estimate

One may use formula (7) to define a gradient estimate of the performance measure w.r.t. the parameter α based on the discrete process (X_t^Δ, Z_t^Δ):

$$g(\Delta) := \nabla_x r(X_T^\Delta) Z_T^\Delta. \tag{18}$$

This estimate converges almost surely to the true gradient, as stated now.

Proposition 7 *Assume that r is continuously differentiable. Then*

$$\lim_{\Delta \to 0} g(\Delta) = \nabla_\alpha V(\alpha) \text{ with probability } 1.$$

Proof This is a direct consequence of the almost sure convergence of (X_T^Δ, Z_T^Δ) to (x_T, z_T) and the continuity of $\nabla_x r$. ■

Now, let us illustrate how Z_t^Δ may be approximated with information available to the agent. The definition (16) of Z_t^Δ requires the term $\nabla_x f(X_t^\Delta, u)$. We now explain how to built a consistent approximation $\widehat{\nabla_x f}(X_t^\Delta, u)$ of this term from the past of the trajectory $(X_s^\Delta)_{0 \le s \le t}$.

3.2 Least-Squares Approximation of $\nabla_x f(X_t^\Delta, u)$

For clarity, in this subsection, we omit reference to Δ, for example writing X_s instead of X_s^Δ. Write $\Delta X_t = X_{t+\Delta} - X_t$ the jump of the state. Let $c > 0$ be a constant (independent of Δ). Define $S(t) := \{s \in [t - c\Delta, t] \,|\, u_s = u_t\}$ the set of past discrete times $t - c\Delta \le s \le t$ when action u_t have been chosen. Note that the cardinality of $S(t)$ is independent from Δ, and solely depends on c and the actual sequence of controls chosen according to the stochastic policy π_α.

From Taylor's formula, for all discrete time s,

$$\Delta X_s = X_{s+\Delta} - X_s = f(X_s, u_t)\Delta + \nabla_x f(X_s, u_t) f(X_s, u_t) \frac{\Delta^2}{2} + O(\Delta^3). \tag{19}$$

Now, for $s \in S(t)$ we have $X_t - X_s = O(\Delta)$, thus

$$f(X_s,u_t) = f(X_t,u_t) + \nabla_x f(X_t,u_t)(X_s - X_t) + O(\Delta^2),$$

from which we deduce (using the fact that $\nabla_x f(X_s,u_t) = \nabla_x f(X_t,u_t) + O(\Delta)$) that

$$\begin{aligned}
\Delta X_s &= \Delta X_t + \big[\nabla_x f(X_s,u_t) f(X_s,u_t) - \nabla_x f(X_t,u_t) f(X_t,u_t)\big]\frac{\Delta^2}{2} \\
&\quad + \nabla_x f(X_t,u_t)(X_s - X_t)\Delta + O(\Delta^3) \\
&= \Delta X_t + \nabla_x f(X_t,u_t)[X_s - X_t + \frac{1}{2}(\Delta X_s - \Delta X_t)]\Delta + O(\Delta^3) \qquad (20) \\
&= b + A(X_s + \frac{1}{2}\Delta X_s)\Delta + O(\Delta^3)
\end{aligned}$$

with $b := \Delta X_t - \nabla_x f(X_t,u_t)(X_t + \frac{1}{2}\Delta X_t)\Delta$ and $A := \nabla_x f(X_t,u_t)$. Based on the observation of several jumps $\{\Delta X_s\}_{s\in S(t)}$, one may derive an approximation of $\nabla_x f(X_t,u_t)$ by solving the least-squares problem:

$$\min_{A,b} \frac{1}{n_t} \sum_{s\in S(t)} \Big\| \Delta X_s - b - A\big(X_s + \frac{1}{2}\Delta X_s\big)\Delta \Big\|^2, \qquad (21)$$

where n_t is the cardinality of $S(t)$. Write $X_s^+ := X_s + \frac{1}{2}\Delta X_s = \frac{1}{2}(X_s + X_{s+\Delta})$ and use the simplified notations: $\overline{X}$, $\overline{XX'}$, $\overline{\Delta X}$, and $\overline{\Delta X X'}$, to denote the average values, when $s \in S(t)$, of X_s^+, $X_s^+(X_s^+)'$, ΔX_s, and $\Delta X_s (X_s^+)'$, respectively. For example,

$$\overline{X} := \frac{1}{n_t} \sum_{s\in S(t)} X_s^+.$$

The optimality condition for (21) holds when the matrix $Q_t := \overline{XX'} - \overline{X}\,\overline{X}'$ is invertible, and in that case, the least squares solution provides the approximation $\widehat{\nabla_x f}(X_t,u_t)$ of $\nabla_x f(X_t,u_t)$:

$$\widehat{\nabla_x f}(X_t,u_t) = \frac{1}{\Delta}\big(\overline{\Delta X X'} - \overline{\Delta X}\,\overline{X}'\big)\big(\overline{XX'} - \overline{X}\,\overline{X}'\big)^{-1}. \qquad (22)$$

This optimality condition does not hold when the set of points $(X_s^+)_{s\in S(t)}$ lies in a vector space of dimension $< d$ (then, Q_t is degenerate). In order to circumvent this problem, we assume that the eigenvalues of the matrix Q_t are bounded away from 0, in the sense given in the following proposition (whose proof in provided in Appendix C).

Proposition 8 *The matrix $Q_t = \overline{XX'} - \overline{X}\,\overline{X}'$ is symmetric non-negative. Let $\nu(\Delta) \geq 0$ be the smallest eigenvalue of Q_t, for all $0 \leq t \leq T$. Then, if $\nu(\Delta) > 0$ and $\nu(\Delta)$ satisfies*

$$\frac{1}{\nu(\Delta)} = o(\Delta^{-4}), \qquad (23)$$

then, for all $0 \leq t \leq T$, the least squares estimate $\widehat{\nabla_x f}(X_t,u_t)$ defined by (22) is consistent with the gradient $\nabla_x f(X_t,u_t)$, that is:

$$\lim_{\Delta\to 0} \widehat{\nabla_x f}(X_t,u_t) = \nabla_x f(X_t,u_t).$$

The condition (23) is not easy to check since it depends on the state dynamics and the policy. Note however that, when we use a strict stochastic policy (i.e., $\pi_\alpha > 0$), a sufficient condition for the set of points $(X_s^+)_{s\in S(t)}$ to span a vector space of dimension d is that the system be (at least locally) controllable. In the case of linear systems $dx/dt = Ax + Bu$, where $u \in U = \mathbb{R}^q$, and A and B being $d \times d$ and $d \times q$-matrices respectively, a necessary and sufficient condition for controllability is that the $d \times (qd)$ controllability matrix $[B : AB : A^2B : \cdots : A^{d-1}B]$ has rank d (this is the so-called *Kalman rank condition* (Kalman et al., 1969)). In more general settings, for example when f is a linear combination of vector fields $h_i(x)$ weighted by the control components, i.e. $f(x,u) = \sum_{i=1}^q h_i(x)u_i$, a sufficient condition for controllability is that the dimension of the Lie algebra generated by the fields $\{h_i\}$ is d (see e.g. (LaValle, 2006)). Intuitively, this dimension represents the number of possible independent directions of movement when following any sequence of controls.

In our numerical experiments, we observed the convergence of the $\nabla_x f$ estimate.

Remark 9 *A simple on-line way for approximating $\nabla_x f$ is to consider a weighted least-squares problem using an exponential weight (with some coefficient $\lambda \in (0,1)$) instead of the rectangular window $s \in [t - c\Delta, t]$. The piece of information related to a time $s < t$ is weighted by λ^p, where p is the number of times the control u has been chosen between s and t. It is easy to adapt the proof of Proposition 8 to derive that a such weighted least squares estimate for $\nabla_x f$ is consistent, for any $\lambda \in (0,1)$, under the same condition (23).*

An on-line update rule would consider tables for the average values $\overline{Y}(u)$ (where $\overline{Y}$ means $\overline{X}$, $\overline{XX'}$, $\overline{\Delta X}$, or $\overline{\Delta X X'}$) for all $u \in U$. The values are initialized (at the first time t each action u is encountered) by Y_t, where Y_t means X_t^+, $X_t^+(X_t^+)'$, ΔX_t, and $\Delta X_t(X_t^+)'$, respectively. Then, the values are updated at time t, according to

$$\begin{array}{lll} \overline{Y}(u) \leftarrow \lambda \overline{Y}(u) + (1-\lambda) Y_t & \text{for} & u = u_t, \\ \overline{Y}(u) \text{ stays unchanged} & \text{for} & u \neq u_t. \end{array}$$

The quantities $\overline{X}$, $\overline{XX'}$, $\overline{\Delta X}$, and $\overline{\Delta X X'}$ are easily updated and the estimate $\widehat{\nabla_x f}$ may advantageously be computed from (22) by using an iterative matrix inversion, such as with the Sherman-Morrison formula (see for example (Golub and Loan, 1996)).

Note that for the first discrete times t, the matrix $\overline{XX'} - \overline{X}\,\overline{X}'$ may not be invertible, simply because there is not enough points $(X_s)_{s<t}$ to form a subspace of dimension d. We may simply set $\widehat{\nabla_x f}$ to 0, which has no impact on the general convergence result.

3.3 The Reinforcement Learning Algorithm

Here, we derive a convergent policy gradient estimate in which all information required to build the state gradient Z_t^Δ is the past trajectory $(X_s^\Delta)_{0\le s\le t}$.

Choose a time step Δ. For a given stochastic policy π_α, the algorithm proceeds as follows:

1. At time $t = 0$, initialise $X_0^\Delta = x$ and $Z_0^\Delta = 0$.
2. For each discrete time $t \in \{(t_n)_{0\le n<N}\}$, choose an action $u_t \sim \pi_\alpha(t, X_t^\Delta)$ according to the stochastic policy π_α and keep this action for a period of time Δ, which moves the system from X_t^Δ to $X_{t+\Delta}^\Delta$ (summarized by the dynamics (15)).

3. Update the average values $\overline{X}$, $\overline{XX'}$, $\overline{\Delta X}$, or $\overline{\Delta X X'}$, for all $u \in U$, as described in subsection 3.2, for example by using an exponential trace with parameter $\lambda \in (0,1)$ as mentioned in Remark 9.

4. Compute the state dynamics gradient approximation $\widehat{\nabla_x f}(X_t^\Delta, u_t)$ according to

$$\widehat{\nabla_x f}(X_t^\Delta, u_t) = \frac{1}{\Delta}\big(\overline{\Delta X X'} - \overline{\Delta X}\,\overline{X}'\big)\big(\overline{XX'} - \overline{X}\,\overline{X}'\big)^{-1}.$$

5. Update Z_t^Δ according to

$$\begin{aligned} Z_{t+\Delta}^\Delta &= Z_t^\Delta + \Delta X_t^\Delta \Big[\frac{\big[\nabla_\alpha \pi_\alpha(u_t|t,X_t^\Delta)\big]'}{\pi_\alpha(u_t|t,X_t^\Delta)} + \frac{\big[\nabla_x \pi_\alpha(u_t|t,X_t^\Delta)\big]'}{\pi_\alpha(u_t|t,X_t^\Delta)} Z_t^\Delta \Big] \\ &\quad + \widehat{\nabla_x f}(t,X_t^\Delta,u_t) Z_t^\Delta \Delta. \end{aligned} \tag{24}$$

6. Repeat steps 2-5 until $t = T$. Then return the policy gradient estimate $\nabla_x r(X_T^\Delta) Z_T^\Delta$.

This algorithm returns a consistent approximation of the policy gradient $\nabla_\alpha V(\alpha)$, as stated now.

Proposition 10 *Assume that the property (23) of Proposition 8 holds, and that the reward function is continuously differentiable. Then the estimate $\nabla_x r(X_T^\Delta) Z_T^\Delta$ returned by the RL algorithm is a consistent approximation of the policy gradient $\nabla_\alpha V(\alpha)$, in the sense that $\nabla_x r(X_T^\Delta) Z_T^\Delta$ converges almost surely to $\nabla_\alpha V(\alpha)$ when $\Delta \to 0$.*

Proof From Proposition 8, $\widehat{\nabla_x f}$ is a consistent approximation of $\nabla_x f$, thus the process (Z_t^Δ) built from (24) also satisfies the consistency condition (17), and the proof follows like in Proposition 7.■

3.4 Illustration on a Simple Example

Let us illustrate this algorithm on the simple example described in the introduction (for which we observed the infinite variance of the likelihood ratio estimate in the continuous time limit).

The continuous process is defined by (8) and the discrete time stochastic process by (9). With the notations used in the introduction, the state gradient dynamics (24) is:

$$Z_{t_{n+1}}^\Delta = Z_{t_n}^\Delta + (X_{t_{n+1}}^\Delta - X_{t_n}^\Delta)\frac{\nabla_\alpha \pi_\alpha(u_{t_n}|t,X_{t_n}^\Delta)}{\pi_\alpha(u_{t_n}|t,X_{t_n}^\Delta)} = Z_t^\Delta + \begin{pmatrix} U_n/\alpha \\ (1-U_n)/(\alpha-1) \end{pmatrix}\Delta.$$

Thus the gradient estimate (18) is

$$g(\Delta) = \nabla r(X_{T=1}^\Delta) Z_{T=1,1}^\Delta = \frac{1}{\alpha N}\Big(\sum_{n=0}^{N-1} U_n\Big).$$

Since $\mathbb{E}[g(\Delta)] = 1$, this is an unbiased estimate of the true gradient $\nabla_\alpha V(\alpha) = \nabla_\alpha r(x_1) = 1$. Moreover, its variance $\mathrm{Var}[g(\Delta)] = \frac{1}{\alpha^2 N}\mathrm{Var}[U_n] = \frac{1-\alpha}{\alpha N}$ decreases to 0 when N goes to infinity, which contrast with the variance of the likelihood ratio estimate (11).

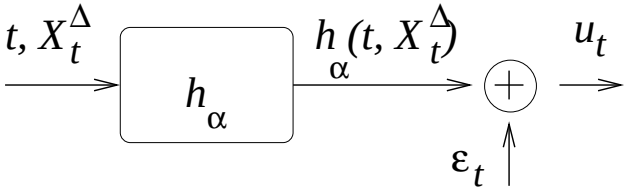


Figure 2: A stochastic policy $u_t = h_\alpha(t, X_t^\Delta) + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, v(\Delta))$.

3.5 The Continuous Control Space Case

So far, we have used notations for a finite control space U. However, the same results hold in the case of a continuous control space $U \in \mathbb{R}^q$. Let us illustrate a simple way for defining a stochastic policy based on a parameterized deterministic policy. Let $h_\alpha : [0,T] \times \mathbb{R}^d \to U = \mathbb{R}^q$ be a deterministic policy parameterized by α (which may be implemented by a neural network, or with any other function approximator). We search for a value of the parameter α that maximizes the performance of the corresponding policy.

We build a stochastic policy by perturbing h_α with a centered Gaussian noise of covariance matrix $v(\Delta)$ (i.e. which depends on the discretization time-step Δ). Thus $u_t = h_\alpha(t, X_t^\Delta) + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, v(\Delta))$. See Figure 2. We assume that $\lim_{\Delta \to 0} v(\Delta) = 0$.

This stochastic policy admits a probability density representation $\pi_\alpha(u|t,x)$:

$$\pi_\alpha(u|t,x) = \frac{1}{\sqrt{(2\pi)^p |v(\Delta)|}} \exp\Big[-\frac{1}{2}(u - h_\alpha(t,x))' v(\Delta)^{-1} (u - h_\alpha(t,x)) \Big].$$

The stochastic process (X_t^Δ) built according to (15) from this stochastic policy π_α is consistent with the continuous process (x_t) defined by the parameterized deterministic policy h_α:

$$\frac{dx_t}{dt} = f(x, h_\alpha(t,x)).$$

Indeed, from the continuity of f, and the assumption that $v(\Delta) \stackrel{\Delta \to 0}{\longrightarrow} 0$, the average state dynamics vector using the stochastic policy π_α tends to the state dynamics vector using the deterministic policy h_α:

$$\lim_{\Delta \to 0} \int_{\mathbb{R}^q} f(x,u) \pi_\alpha(u|t,x) du = f(x, h_\alpha(t,x)),$$

and the consistency property (13) as well as the bound (14) hold (for the same reasons as those invoked in subsection 2.2). Thus, the reinforcement learning algorithm of subsection 3.3 applies directly.

Note that from the specific form of the policy $\pi_\alpha(u|t,x)$, the likelihood ratios are easily computed: for each parameter α_i, $1 \le i \le m$, $\frac{\partial_{\alpha_i} \pi_\alpha(u|t,x)}{\pi_\alpha(u|t,x)} = \partial_{\alpha_i} h_\alpha(t,x) v(\Delta)^{-1} (u - h_\alpha(t,x))$, and for each coordinate x_i, $1 \le i \le d$, $\frac{\partial_{x_i} \pi_\alpha(u|t,x)}{\pi_\alpha(u|t,x)} = \partial_{x_i} h_\alpha(t,x) v(\Delta)^{-1} (u - h_\alpha(t,x))$.

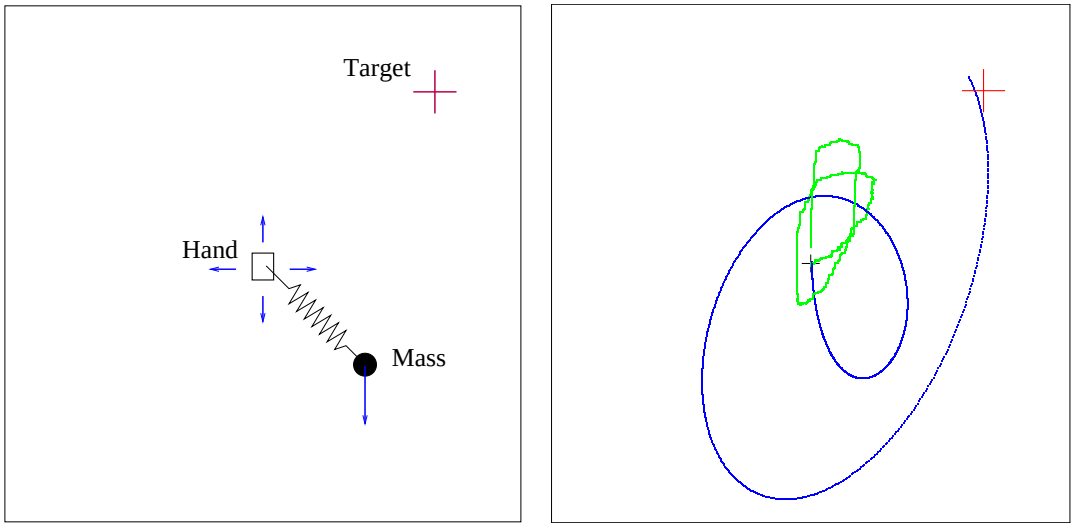
4. Numerical Experiments

We provide two experiments, a target problem and an inverted pendulum, that illustrate the reinforcement learning algorithm described in subsection 3.3 in the case of a finite and a continuous control space, respectively.

4.1 A Target Problem

This is a 6 dimensional system $(x_0, y_0, x, y, v_x, v_y)$ that represents a hand ((x_0, y_0) position) holding a spring to which is attached a mass (defined by its position (x, y) and velocity (v_x, v_y)) subject to gravitation. The control is the movement of the hand, in any 4 possible directions (up, down, left, right). The goal is to reach a target (x_G, y_G) with the mass at a specific time T (see Figure 3a), while keeping the hand close to the origin. For that purpose, the terminal reward function is defined by

$$r = -x_0^2 - y_0^2 - (x - x_G)^2 - (y - y_G)^2.$$



(a) The physical system

(b) A trajectory (the mass and the hand) starting from the origin

Figure 3: (a) the physical system. (b) A trajectory obtained after 1000 gradient steps. For that specific trajectory, the performance (terminal reward) was -0.087.

The state dynamics is:

$$\begin{aligned} \dot{x_0} &= u_x, \quad \dot{x} = v_x, \quad \dot{v_x} = -\tfrac{k}{m}(x - x_0), \\ \dot{y_0} &= u_y, \quad \dot{y} = v_y, \quad \dot{v_y} = -\tfrac{k}{m}(y - y_0) - g, \end{aligned}$$

with k being the spring constant, m the mass, g the gravitational constant, and $(u_x, u_y) = u \in U := \{(1,0),(0,1),(-1,0),(0,-1)\}$ the control. We consider a Boltzmann-like stochastic policy

$$\pi_\alpha(u|t,x) = \frac{\exp Q_\alpha(t,x,u)}{\sum_{u' \in U} \exp Q_\alpha(t,x,u')}$$

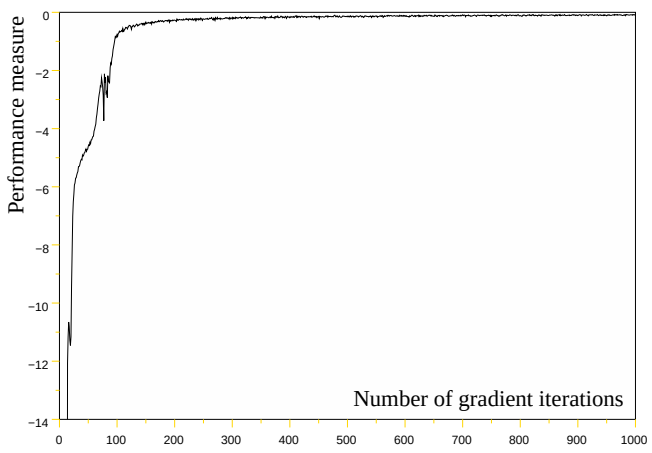


Figure 4: Performance of successive parameterized controllers.

with a linear parameterization of the Q_α values: $Q_\alpha(t,x,u) = \alpha_0^u + \alpha_1^u t + \alpha_2^u x_0 + \alpha_3^u y_0 + \alpha_4^u x + \alpha_5^u y + \alpha_6^u v_x + \alpha_7^u v_y$, for each 4 possible actions $u \in U$. Thus the parameter $\alpha \in I\!R^{32}$. We initialized α with uniform random values in the range $[-0.01, 0.01]$. In our experiments we chose $k=1$, $m=1$, $g=1$, $x_G = y_G = 2$, $\lambda = 0.9$, $\Delta = 0.01$, $T = 10$.

At each iteration, we run one trajectory $(X_t)_{0 \le t \le T}$ using the stochastic policy, and calculate the policy gradient estimate according to the RL algorithm described in subsection 3.3. We then perform a gradient ascent step (5) (with a fixed step $\eta = 0.01$). Figure 4 shows the performance of the parameterized controller as a function of the number of gradient iterations.

For that problem, we chose initial states uniformly distributed over the domain $[-0.1, 0.1]^6$. We found that the randomness introduced in the choice of the initial state helped in not getting stuck in local minima. Here, convergence of the gradient method occurs to a controller close to optimality (for which $r = 0$). We illustrate in Figure 3b the trajectory (where only the hand and the mass positions are shown) obtained after 1000 gradient steps, starting from the initial state $(x_0, y_0, x, y, v_x, v_y)_{t=0} = 0$.

4.2 Double Inverted Pendulum

We illustrate the approach described in subsection 3.5 on this continuous control space problem. This is an double inverted pendulum defined in the 6-dimensions: the position of the cart, its velocity, the two angles, and their angular velocity $x = (y, v, \theta_1, \omega_1, \theta_2, \omega_2)' \in I\!R^6$ (see Figure 5). The control $u \in U = I\!R$ (continuous variable) is the force applied to the cart. The state dynamics are described in (Bogdanov, 2004). The goal is to reach the unstable equilibrium $(y, v, \theta_1, \omega_1, \theta_2, \omega_2) = 0$ at time $T = 5$. We consider the quadratic reward function $r(x) = -(y^2 + v^2 + \theta_1^2 + \omega_1^2 + \theta_2^2 + \omega_2^2)$.

Like in subsection 3.5, we build a stochastic policy by adding a Gaussian noise of variance $v(\Delta) = \Delta I$ (where I is the identity matrix) to a linearly parameterized (time independent) determin-

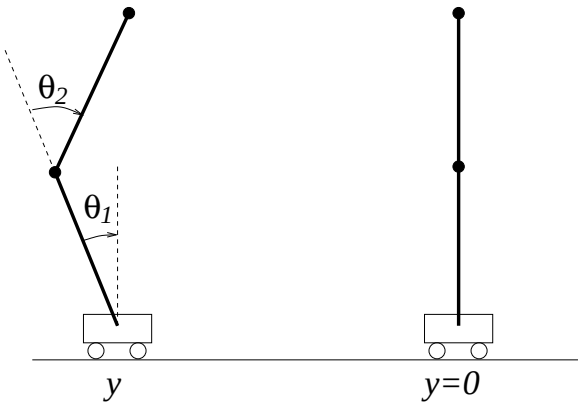


Figure 5: The double inverted pendulum. Current position and target position.

istic policy $h_\alpha(t,x) = \alpha_1 + \alpha_2 y + \alpha_3 v + \alpha_4\theta_1 + \alpha_5\omega_1 + \alpha_6\theta_2 + \alpha_7\omega_2$, i.e. the control at time t is $u_t \sim h_\alpha(t,x_t) + \mathcal{N}(0, v(\Delta))$.

We wish to find a local maximum of the performance measure $V(\alpha) = r(x_T)$ in the space of the policy parameters $\alpha \in \mathbb{R}^7$. We initialized α with uniform random values in the range $[-0.01, 0.01]$, and perform a stochastic gradient algorithm (5) where the gradient $\nabla_\alpha V(\alpha)$ is computed according to the reinforcement learning algorithm defined in subsection 3.3.

A gradient step update (5) is performed (with $\eta = 1$) at the end of each sample trajectory starting from an initial state, chosen uniformly randomly in the domain defined by $y \in [-1,1]$, $\theta_1 \in [-0.3, 0.3]$, $\theta_2 \in [-0.3, 0.3]$, and $v = 0$, $\omega_1 = 0$, $\omega_2 = 0$. We use a discretization time-step $\Delta = 10^{-3}$ which is low enough to provide a very good approximation of the true gradient, that is the gradient that would be obtained from the continuous (but unknown from the agent) state dynamics by using the deterministic policy $h_\alpha(t,x)$.

Figure 6 shows (in bold) the performance measure (terminal reward) at the end of each trajectory as a function of the number of gradient iterations. The other curves give the values of the $(\alpha_1, \ldots, \alpha_7)$ during simulations.

After 1000 gradient iterations, the obtained policy is $h_\alpha(t,x) = -0.0023 - 5.31y - 1.74v + 11.16\theta_1 + 0.92\omega_1 - 7.77\theta_2 - 3.94\omega_2$, and the resulting average performance is -0.097 for trajectories starting randomly from the same domain as during learning. In this problem, a linear controller is sufficient to derive a controller close to optimality. However, we should mention that for initial states in another domain (say, if the angles were not close to 0, and loops would be required to reach the target position), the problem would not possibly be solved with such a simple class of policies.

5. Conclusion

We described a reinforcement learning method for approximating the gradient of the performance measure of a continuous-time deterministic problem, with respect to the control parameters. This was obtained by using a stochastic policy to approximate the continuous system by a consistent stochastic discrete process. We showed how using a perturbated parameterized deterministic policy enables to process a consistent (when the perturbation goes to 0) gradient estimate only from the observable data.

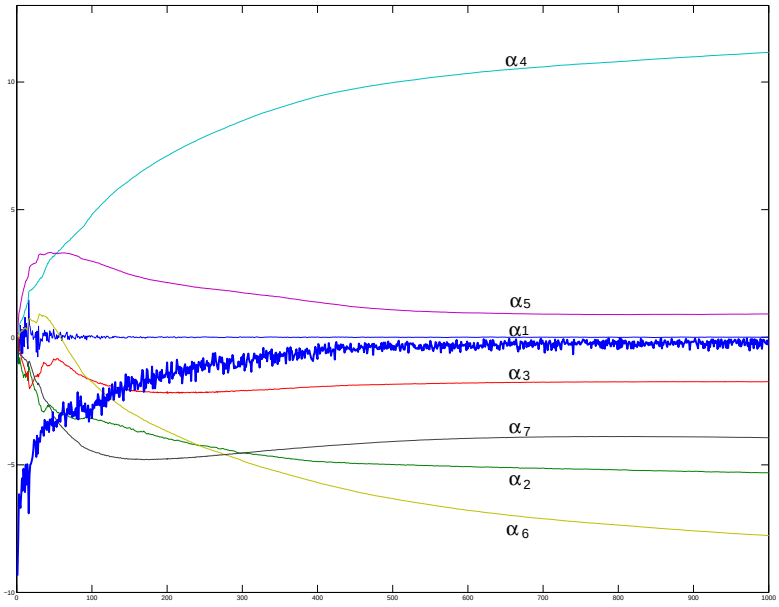


Figure 6: The bold curve shows the performance measure $V(\alpha)$, and the other curves the values of $(\alpha_1, \ldots, \alpha_7)$, as a function of the number of gradient iterations.

In future work, it would be interesting to extend this method to the case of stochastic dynamics, and to non-smooth reward functions (or in case the reward gradient is unknown from the agent), by using integration-by-part formula for the gradient estimate, such as the *likelihood ratio method* of (Yang and Kushner, 1991) or the *martingale approach* of (Gobet and Munos, 2005).

Appendix A. Proof of Proposition 2

The likelihood ratio estimate (10) may be rewritten

$$\begin{aligned} g(\Delta) &= \frac{1}{\alpha(1-\alpha)N}\Big(\sum_{n=0}^{N-1} U_n\Big)\sum_{n=0}^{N-1}\big(U_n-\alpha\big) \\ &= \frac{1}{\alpha(1-\alpha)N}\Big[\Big(\sum_{n=0}^{N-1} V_n\Big)^2 + \alpha N\sum_{n=0}^{N-1} V_n\Big], \end{aligned}$$

with $V_n := U_n - \alpha$. From the fact that $\mathbb{E}[V_n^2] = \alpha(1-\alpha)$, the expectation of the estimate is

$$\mathbb{E}[g(\Delta)] = \frac{1}{\alpha(1-\alpha)N}\mathbb{E}\Big[\Big(\sum_{n=0}^{N-1} V_n\Big)^2\Big] = 1.$$

Now its variance $\mathrm{Var}[g(\Delta)]$ is

$$\frac{1}{[\alpha(1-\alpha)N]^2}\mathrm{Cov}\Big[\Big(\sum_{n=0}^{N-1} V_n\Big)\Big(\sum_{p=0}^{N-1} V_p\Big) + \alpha N\sum_{n=0}^{N-1} V_n, \Big(\sum_{n'=0}^{N-1} V_{n'}\Big)\Big(\sum_{p'=0}^{N-1} V_{p'}\Big) + \alpha N\sum_{n=0}^{N-1} V_{n'}\Big]. \qquad (25)$$

Notice that from the independence of the Bernoulli random variables (U_n), all terms $\mathrm{Cov}(V_n, V_{n'}) = 0$ for $n \neq n'$, and $\mathrm{Cov}(V_n, V_n) = \mathbb{E}[(U_n - \alpha)^2] = \alpha(1-\alpha)$.

The terms $\mathrm{Cov}(V_n, V_{n'}V_{p'}) = \mathbb{E}\big[V_n(V_{n'}V_{p'} - \mathbb{E}[V_{n'}V_{p'}])\big] = \mathbb{E}[V_nV_{n'}V_{p'}]$ (because V_n is centered) equal 0 whenever $n \neq n'$ or $n \neq p'$. And $\mathrm{Cov}(V_n, V_n^2) = \mathbb{E}[V_n^3] = \alpha(1-\alpha)(1-2\alpha)$.

Now, $\mathrm{Cov}(V_nV_p, V_{n'}V_{p'}) = 0$ when $n \neq n'$, $n \neq p'$, $p \neq n'$, and $p \neq p'$ (because the variables V_nV_p and $V_{n'}V_{p'}$ are independent). The terms $\mathrm{Cov}(V_nV_p, V_nV_{p'}) = \mathbb{E}\big[(V_nV_p - \mathbb{E}[V_nV_p])(V_nV_{p'} - \mathbb{E}[V_nV_{p'}])\big] = \mathbb{E}[V_nV_pV_nV_{p'}] = 0$ for $n \neq p$, $n \neq p'$, and $p \neq p'$ (independence of V_p and $V_n^2V_{p'}$). Now, $\mathrm{Cov}(V_nV_p, V_nV_p) = \mathbb{E}[(V_nV_p)^2] = \alpha^2(1-\alpha)^2$ when $n \neq p$. Finally, $\mathrm{Cov}(V_n^2, V_n^2) = \mathbb{E}[V_n^4] - \big(\mathbb{E}[V_n^2]\big)^2 = \alpha(1-\alpha)(1-3\alpha+3\alpha^2) - \alpha^2(1-\alpha)^2 = \alpha(1-\alpha)(1-4\alpha+4\alpha^2)$.

Thus, the covariance term in (25) is

$$N\alpha(1-\alpha)(1-4\alpha+4\alpha^2) + N(N-1)\alpha^2(1-\alpha)^2 + \alpha N^2\alpha(1-\alpha)(1-2\alpha) + \alpha^2N^3\alpha(1-\alpha)$$

and the variance of the likelihood ratio estimate is

$$\mathrm{Var}[g(\Delta)] = \frac{1 - 5(1-\alpha) + (2-3\alpha)\alpha N + \alpha^2N^2}{\alpha(1-\alpha)N}.$$

Appendix B. Proof of Theorem 3

For convenience, we write x_n for x_{t_n}, X_n for $X^\Delta_{t_n}$, u_n for u_{t_n}, and U_n for U_{t_n}, $0 \leq n \leq N$. Let us define the average approximation errors $m^\Delta_n = \mathbb{E}[||X_n - x_n||]$ and the squared errors $v^\Delta_n = \mathbb{E}[||X_n - x_n||^2]$. Here, we prove the convergence at the terminal time T, i.e. that $X_T \to x_T$ almost surely when $\Delta \to 0$.

B.1 Convergence of the Squared Error $\mathbb{E}[||X^\Delta_T - x_T||^2]$:

We use the decomposition:

$$\begin{aligned} v^\Delta_{n+1} &= \mathbb{E}[||X_{n+1} - X_n||^2] + \mathbb{E}[||X_n - x_n||^2] + \mathbb{E}[||x_n - x_{n+1}||^2] \\ &\quad + 2\mathbb{E}[(X_n - x_n)'(X_{n+1} - X_n + x_n - x_{n+1})] \\ &\quad + 2\mathbb{E}[(X_{n+1} - X_n)'(x_n - x_{n+1})]. \end{aligned} \tag{26}$$

From the bounded jumps property (14), $\mathbb{E}[||X_{n+1} - X_n||^2] = O(\Delta^2)$. From Taylor's formula,

$$x_{n+1} - x_n = f(x_n)\Delta + O(\Delta^2), \tag{27}$$

thus $\mathbb{E}[||x_n - x_{n+1}||^2] = O(\Delta^2)$ (since f is Lipschitz, and x_t and $f(x_t)$ are uniformly bounded on $[0,T]$) and from Cauchy-Schwarz inequality, $|\mathbb{E}[(X_{n+1} - X_n)'(x_n - x_{n+1})]| = O(\Delta^2)$. From (13) and (27),

$$\mathbb{E}[X_{n+1} - X_n + x_n - x_{n+1}|X_n] = [f(X_n) - f(x_n)]\Delta + o(\Delta). \tag{28}$$

Now, from (14) we deduced that $||X_n - x_0|| = O(1)$ thus X_n is bounded (for all n and N), as well as x_n. Let B a constant such that $||X_n|| \leq B$ and $||x_n|| \leq B$ for all $n \leq N$, $N \geq 0$. Since f is $\mathcal{C}^2$, from Taylor's formula, there exists a constant k, such that, for all $n \leq N$,

$$||f(X_n) - f(x_n) - \nabla_x f(x_n)(X_n - x_n)|| \leq k||X_n - x_n||^2. \tag{29}$$

We deduce that

$$\begin{aligned}|\mathbb{E}[(X_n - x_n)'(X_{n+1} - X_n + x_n - x_{n+1})]| &= \left|\mathbb{E}\left[(X_n - x_n)'(f(X_n) - f(x_n))\right]\right|\Delta + o(\Delta) \\ &\leq \left|\mathbb{E}\left[(X_n - x_n)'\nabla_x f(x_n)(X_n - x_n)\right]\right|\Delta + 2kBv_n\Delta + o(\Delta) \\ &\leq Mv_n^\Delta\Delta + o(\Delta)\end{aligned}$$

with $M = \sup_{||x||\leq B} ||\nabla_x f(x)|| + 2kB$. Thus, (26) leads to the recurrent bound

$$v_{n+1}^\Delta \leq (1 + M\Delta)v_n^\Delta + o(\Delta).$$

This actually means that there exists a function $e(\Delta) \to 0$ when $\Delta \to 0$, such that $v_{n+1}^\Delta \leq (1 + M\Delta)v_n^\Delta + e(\Delta)\Delta$. Thus,

$$v_N^\Delta \leq \frac{(1+M\Delta)^N - 1}{(1+M\Delta) - 1}e(\Delta)\Delta \leq (e^{NM\Delta} - 1)\frac{1}{M}e(\Delta)$$

thus $v_N^\Delta = o(1)$, that is $\mathbb{E}[||X_T^\Delta - x_T||^2] \xrightarrow{\Delta\to 0} 0$.

B.2 Convergence of the Mean $\mathbb{E}[||X_T^\Delta - x_T||]$:

From (28), we have

$$\mathbb{E}[X_{n+1} - x_{n+1}|X_n] = X_n - x_n + [f(X_n) - f(x_n)]\Delta + o(\Delta).$$

Thus from (29),

$$\begin{aligned}m_{n+1}^\Delta = \mathbb{E}[||X_{n+1} - x_{n+1}||] &\leq (1 + ||\nabla_x f(x_n)||\Delta)\mathbb{E}[||X_n - x_n||] + kv_n^\Delta\Delta + o(\Delta) \\ &\leq (1 + M'\Delta)m_n^\Delta + o(\Delta),\end{aligned}$$

since $v_n^\Delta = o(1)$ (with $M' = \sup_{||x||\leq B} ||\nabla_x f(x)||$). Using the same deduction as above, we obtain that $m_N^\Delta = o(1)$, that is $\mathbb{E}[||X_T^\Delta - x_T||] \xrightarrow{\Delta\to 0} 0$.

B.3 Almost Sure Convergence

Here, we use the *concentration-of-measure phenomenon* (Talagrand, 1996; Ledoux, 2001), which states that under mild conditions, a function (say Lipschitz or with bounded differences) of many independent random variables concentrates around its mean, in the sense that the tail probability decreases exponentially fast.

From the definition of the discrete state process (12), one may write the state X_N as a function h of the independent random variables $(U_n)_{0\leq n<N}$, i.e.

$$X_N - x_0 = h(U_0, \ldots, U_{N-1}) := \sum_{n=0}^{N-1}(X_{n+1} - X_n). \tag{30}$$

Observe that $h - \mathbb{E}[h] = \sum_{n=0}^{N-1} d_n$ with $d_n = X_{n+1} - X_n - \mathbb{E}[X_{n+1} - X_n]$ being a martingale difference sequence (that is $\mathbb{E}[d_n|U_0, \ldots, U_{n-1}] = 0$). Now, from (Ledoux, 2001, lemma 4.1), one has:

$$\mathbb{P}(||h - \mathbb{E}[h]|| \geq \varepsilon) \leq 2e^{-\varepsilon^2/(2D^2)} \tag{31}$$

for any $D^2 \geq \sum_{n=0}^{N-1} ||d_n||_\infty^2$. Thus, from (14), and since $f^\Delta(X_n)$ is bounded (for all $n < N$ and all $N > 0$), there exists a constant C that does not depend on N such that $d_n \leq C/N$. Thus we may take $D^2 = C^2/N$.

Now, from the previous paragraph, $||\mathbb{E}[X_N] - x_N|| \leq e(N)$, with $e(N) \to 0$ when $N \to \infty$. This means that $||h - \mathbb{E}[h]|| + e(N) \geq ||X_N - x_N||$, thus

$$\mathbb{P}(||h - \mathbb{E}[h]|| \geq \varepsilon + e(N)) \geq \mathbb{P}(||X_N - x_N|| \geq \varepsilon),$$

and we deduce from (31) that

$$\mathbb{P}(||X_N - x_N|| \geq \varepsilon) \leq 2e^{-N(\varepsilon + e(N))^2/(2C^2)}.$$

Thus, for all $\varepsilon > 0$, the series $\sum_{N \geq 0} \mathbb{P}(||X_N - x_N|| \geq \varepsilon)$ converges. Now, from Borel-Cantelli lemma, we deduce that for all $\varepsilon > 0$, there exists N_ε such that for all $N \geq N_\varepsilon$, $||X_N - x_N|| < \varepsilon$, which proves the almost sure convergence of X_N to x_N as $N \to \infty$ (i.e. $X_T \overset{\Delta \to 0}{\longrightarrow} x_T$ almost surely).

Appendix C. Proof of Proposition 8

First, note that $Q_t = \overline{X\,X'} - \overline{X}\,\overline{X}'$ is a symmetric, non-negative matrix, since it may be rewritten as

$$\frac{1}{n_t} \sum_{s \in S(t)} (X_s^+ - \overline{X})(X_s^+ - \overline{X})'.$$

In solving the least squares problem (21), we deduce $b = \overline{\Delta X} + A\overline{X}\Delta$, thus

$$\begin{aligned} \min_{A,b} \frac{1}{n_t} \sum_{s \in S(t)} \left\| \Delta X_s - b - A(X_s + \frac{1}{2}\Delta X_s)\Delta \right\|^2 &= \min_A \frac{1}{n_t} \sum_{s \in S(t)} \left\| \Delta X_s - \overline{\Delta X} - A(X_s^+ - \overline{X})\Delta \right\|^2 \\ &\leq \frac{1}{n_t} \sum_{s \in S(t)} \left\| \Delta X_s - \overline{\Delta X} - \nabla_x f(\overline{X}, u_t)(X_s^+ - \overline{X})\Delta \right\|^2. \end{aligned} \tag{32}$$

Now, since $X_s = \overline{X} + O(\Delta)$ one may obtain like in (19) and (20) (by replacing X_t by $\overline{X}$) that:

$$\Delta X_s - \overline{\Delta X} - \nabla_x f(\overline{X}, u_t)(X_s^+ - \overline{X})\Delta = O(\Delta^3). \tag{33}$$

We deduce from (32) and (33) that

$$\frac{1}{n_t} \sum_{s \in S(t)} \left\| \left[\widehat{\nabla_x f}(X_t, u_t) - \nabla_x f(\overline{X}, u_t) \right] (X_s^+ - \overline{X})\Delta \right\|^2 = O(\Delta^6).$$

By developing each component,

$$\sum_{i=1}^{d} \left[\widehat{\nabla_x f}(X_t, u_t) - \nabla_x f(\overline{X}, u_t) \right]_{row\,i} Q_t \left[\widehat{\nabla_x f}(X_t, u_t) - \nabla_x f(\overline{X}, u_t) \right]'_{row\,i} = O(\Delta^4).$$

Now, from the definition of $\nu(\Delta)$, for all vector $u \in \mathbb{R}^d$, $u' Q_t u \geq \nu(\Delta)||u||^2$, thus

$$\nu(\Delta) ||\widehat{\nabla_x f}(X_t, u_t) - \nabla_x f(\overline{X}, u_t)||^2 = O(\Delta^4).$$

Condition (23) yields $\widehat{\nabla_x f}(X_t, u_t) = \nabla_x f(\overline{X}, u_t) + o(1)$, and since $\nabla_x f(X_t, u_t) = \nabla_x f(\overline{X}, u_t) + O(\Delta)$, we deduce

$$\lim_{\Delta \to 0} \widehat{\nabla_x f}(X_t, u_t) = \nabla_x f(X_t, u_t).$$

References

J. Baxter and P. L. Bartlett. Infinite-horizon gradient-based policy search. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

A. Bensoussan. *Perturbation methods in optimal control*. Wiley/Gauthier-Villars Series in Modern Applied Mathematics. John Wiley & Sons Ltd., Chichester, 1988. Translated from the French by C. Tomson.

A. Bogdanov. Optimal control of a double inverted pendulum on a cart. *Technical report CSE-04-006, CSEE, OGI School of Science and Engineering, OHSU*, 2004.

P. W. Glynn. Likelihood ratio gradient estimation: an overview. In A. Thesen, H. Grant, and W. D. Kelton, editors, *Proceedings of the 1987 Winter Simulation Conference*, pages 366–375, 1987.

E. Gobet and R. Munos. Sensitivity analysis using Itô-Malliavin calculus and martingales. application to stochastic optimal control. *SIAM journal on Control and Optimization*, 43(5):1676–1713, 2005.

G. H. Golub and C. F. Van Loan. *Matrix Computations, 3rd ed.* Baltimore, MD: Johns Hopkins, 1996.

R. E. Kalman, P. L. Falb, and M. A. Arbib. *Topics in Mathematical System Theory*. New York: McGraw Hill, 1969.

P. E. Kloeden and E. Platen. *Numerical Solutions of Stochastic Differential Equations*. Springer-Verlag, 1995.

H. J. Kushner and G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, Berlin and New York, 1997.

S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

M. Ledoux. *The concentration of measure phenomenon*. American Mathematical Society, Providence, RI, 2001.

P. Marbach and J. N. Tsitsiklis. Approximate gradient methods in policy-space optimization of Markov reward processes. *Journal of Discrete Event Dynamical Systems*, 13:111–148, 2003.

B. T. Polyak. *Introduction to Optimization*. Optimization Software Inc., New York, 1987.

M. I. Reiman and A. Weiss. Sensitivity analysis via likelihood ratios. In J. Wilson, J. Henriksen, and S. Roberts, editors, *Proceedings of the 1986 Winter Simulation Conference*, pages 285–289, 1986.

R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *Bradford Book*, 1998.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Neural Information Processing Systems. MIT Press*, pages 1057–1063, 2000.

M. Talagrand. A new look at independence. *Annals of Probability*, 24:1–34, 1996.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

J. Yang and H. J. Kushner. A Monte Carlo method for sensitivity analysis and parametric optimization of nonlinear stochastic systems. *SIAM J. Control Optim.*, 29(5):1216–1249, 1991.

Journal of Machine Learning Research 7 (2006) 793–815 Submitted 6/05; Revised 11/05; Published 5/06

Learning Image Components for Object Recognition

Michael W. Spratling MICHAEL.SPRATLING@KCL.AC.UK
Division of Engineering
King's College London, UK, and

Centre for Brain and Cognitive Development
Birkbeck College, London, UK

Editor: Peter Dayan

Abstract

In order to perform object recognition it is necessary to learn representations of the underlying components of images. Such components correspond to objects, object-parts, or features. Non-negative matrix factorisation is a generative model that has been specifically proposed for finding such meaningful representations of image data, through the use of non-negativity constraints on the factors. This article reports on an empirical investigation of the performance of non-negative matrix factorisation algorithms. It is found that such algorithms need to impose additional constraints on the sparseness of the factors in order to successfully deal with occlusion. However, these constraints can themselves result in these algorithms failing to identify image components under certain conditions. In contrast, a recognition model (a competitive learning neural network algorithm) reliably and accurately learns representations of elementary image features without such constraints.

Keywords: non-negative matrix factorisation, competitive learning, dendritic inhibition, object recognition

1. Introduction

An image usually contains a number of different objects, parts, or features and these components can occur in different configurations to form many distinct images. Identifying the underlying components which are combined to form images is thus essential for learning the perceptual representations necessary for performing object recognition. Non-negative matrix factorisation (NMF) has been proposed as a method for finding such parts-based decompositions of images (Lee and Seung, 1999; Feng et al., 2002; Liu et al., 2003; Liu and Zheng, 2004; Li et al., 2001; Hoyer, 2002, 2004). However, the performance of this method has not been rigorously or quantitatively tested. Instead, only a subjective assessment has been made of the quality of the components that are learnt when this method is applied to processing images of, for example, faces (Lee and Seung, 1999; Hoyer, 2004; Li et al., 2001; Feng et al., 2002). This paper thus aims to quantitatively test, using several variations of a simple standard test problem, the accuracy with which NMF identifies elementary image features. Furthermore, non-negative matrix factorisation assumes that images are composed of a linear combination of features. However, in reality the superposition of objects or object parts does not always result in a linear combination of sources but, due to occlusion, results in a non-linear combination. This paper thus also aims to investigate, empirically, how NMF performs when tested in more realistic environments where occlusion takes place. Since competitive learning algorithms

have previously been applied to this test problem, and neural networks are a standard technique for learning object representations, the performance of NMF is compared to that of an unsupervised neural network learning algorithm applied to the same set of tasks.

2. Method

This section describes the NMF algorithms, and the neural network algorithms, which are explored in this paper. The performance of these algorithms is compared in the Results section.

2.1 Non-Negative Matrix Factorisation

Given an m by p matrix $\mathbf{X} = [\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_p]$, each column of which contains the pixel values of an image (*i.e.,* $\mathbf{X}$ is a set of training images), the aim is to find the factors $\mathbf{A}$ and $\mathbf{Y}$ such that

$$\mathbf{X} \approx \mathbf{AY},$$

where $\mathbf{A}$ is an m by n matrix the columns of which contain basis vectors, or components, into which the images can be decomposed, and $\mathbf{Y} = [\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_p]$ is an n by p matrix containing the activations of each component (*i.e.,* the strength of each basis vector in the corresponding training image). A training image ($\vec{\mathbf{x}}_k$) can therefore be reconstructed as a linear combination of the image components contained in $\mathbf{A}$, such that $\vec{\mathbf{x}}_k \approx \mathbf{A}\vec{\mathbf{y}}_k$.

A number of different learning algorithms can be defined depending on the constraints that are placed on the factors $\mathbf{A}$ and $\mathbf{Y}$. For example, vector quantization (VQ) restricts each column of $\mathbf{Y}$ to have only one non-zero element, principal components analysis (PCA) constrains the columns of $\mathbf{A}$ to be orthonormal and the rows of $\mathbf{Y}$ to be mutually orthogonal, and independent components analysis (ICA) constrains the rows of $\mathbf{Y}$ to be statistically independent. Non-negative matrix factorisation is a method that seeks to find factors (of a non-negative matrix $\mathbf{X}$) under the constraint that both $\mathbf{A}$ and $\mathbf{Y}$ contain only elements with non-negative values. It has been proposed that this method is particularly suitable for finding the components of images, since from the physical properties of image formation it is known that image components are non-negative and that these components are combined additively (*i.e.,* are not subtracted) in order to generate images. Several different algorithms have been proposed for finding the factors $\mathbf{A}$ and $\mathbf{Y}$ under non-negativity constraints. Those tested here are listed in Table 1.

Algorithms `nmfdiv` and `nmfmse` impose non-negativity constraints solely, and differ only in the objective function that is minimised in order to find the factors. All the other algorithms extend non-negative matrix factorisation by imposing additional constraints on the factors. Algorithm `lnmf` imposes constraints that require the columns of $\mathbf{A}$ to contain as many non-zero elements as possible, and $\mathbf{Y}$ to contain as many zero elements as possible. This algorithm also requires that basis vectors be orthogonal. Both algorithms `snmf` and `nnsc` impose constraints on the sparseness of $\mathbf{Y}$. Algorithm `nmfsc` allows optional constraints to be imposed on the sparseness of either the basis vectors, the activations, or both. This algorithm was used with three combinations of sparseness constraints. For `nmfsc(A)` a constraint on the sparseness of the basis vectors was applied. This constraint required that each column of $\mathbf{A}$ had a sparseness of 0.5. Valid values for the parameter controlling sparseness could range from 0 (which would produce completely distributed basis vectors) to a value of 1 (which would produce completely sparse basis vectors). For `nmfsc(Y)` a constraint on the sparseness of the activations was applied. This constraint required that each row

	Acronym	Description	Reference
Non-negative Matrix Factorisation Algorithms	`nmfdiv`	NMF with divergence objective	(Lee and Seung, 2001)
	`nmfmse`	NMF with euclidean objective	(Lee and Seung, 2001)
	`lnmf`	Local NMF	(Li et al., 2001; Feng et al., 2002)
	`snmf`	Sparse NMF ($\alpha = 1$)	(Liu et al., 2003)
	`nnsc`	Non-negative sparse coding ($\lambda = 1$)	(Hoyer, 2002)
	`nmfsc(A)`	NMF with a sparseness constraint of 0.5 on the basis vectors	(Hoyer, 2004)
	`nmfsc(Y)`	NMF with a sparseness constraint of 0.7 on the activations	(Hoyer, 2004)
	`nmfsc(A&Y)`	NMF with sparseness constraints of 0.5 on the basis vectors and 0.7 on the activations	(Hoyer, 2004)
Neural Network Algorithms	`nndi`	Dendritic inhibition neural network with non-negative weights	
	`di`	Dendritic inhibition neural network	(Spratling and Johnson, 2002, 2003)

Table 1: The algorithms tested in this article. These include a number of different algorithms for finding matrix factorisations under non-negativity constraints and a neural network algorithm for performing competitive learning through dendritic inhibition.

of **Y** had a sparseness of 0.7 (where sparseness is in the range $[0,1]$, as for the constraint on **A**). For `nmfsc(A&Y)` these same constraints were imposed on both the sparseness of basis vectors and the activations. For the `nmfsc` algorithm, fixed values for the parameters controlling sparseness were used in order to provide a fairer comparison with the other algorithms all of which used constant parameter settings. Furthermore, since all the test cases studied are very similar, it is reasonable to expect an algorithm to work across them all without having to be tuned specifically to each individual task. The particular parameter values used were selected so as to provide the best overall results across all the tasks. The effect of changing the sparseness parameters is described in the discussion section.

2.2 Dendritic Inhibition Neural Network

In terms of neural networks, the factoring of **X** into **A** and **Y** constitutes the formation of a generative model (Hinton et al., 1995): neural activations **Y** reconstruct the input patterns **X** via a matrix of feedback (or generative) synaptic weights **A** (see Figure 1a). In contrast, traditional neural network algorithms have attempted to learn a set of feedforward (recognition) weights **W** (see Figure 1b),

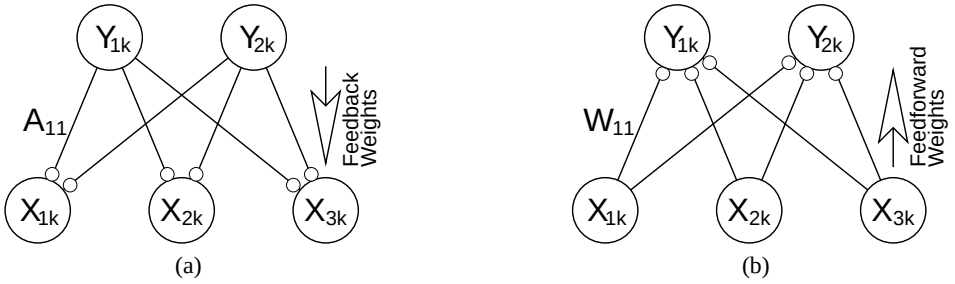


Figure 1: (a) A generative neural network model: feedback connections enable neural activations ($\vec{\mathbf{y}}_k$) to reconstruct an input pattern ($\vec{\mathbf{x}}_k$). (b) A recognition neural network model: feedforward connections cause neural activations ($\vec{\mathbf{y}}_k$) to be generated in response to an input pattern ($\vec{\mathbf{x}}_k$). Nodes are shown as large circles and excitatory synapses as small open circles.

such that

$$\mathbf{Y} = \mathrm{f}(\mathbf{W}, \mathbf{X}),$$

where $\mathbf{W}$ is an n by m matrix of synaptic weight values, $\mathbf{X} = [\vec{\mathbf{x}}_1, \ldots, \vec{\mathbf{x}}_p]$ is an m by p matrix of training images, and $\mathbf{Y} = [\vec{\mathbf{y}}_1, \ldots, \vec{\mathbf{y}}_p]$ is an n by p matrix containing the activation of each node in response to the corresponding image. Typically, the rows of $\mathbf{W}$ form templates that match image features, so that individual nodes become active in repose to the presence of specific components of the image. Nodes thus act as 'feature detectors' (Barlow, 1990, 1995).

Many different functions are possible for calculating neural activations and many different learning rules can be defined for finding the values of $\mathbf{W}$. For example, algorithms have been proposed for performing vector quantization (Kohonen, 1997; Ahalt et al., 1990), principal components analysis (Oja, 1992; Fyfe, 1997b; Földiák, 1989), and independent components analysis (Jutten and Herault, 1991; Charles and Fyfe, 1997; Fyfe, 1997a). Many of these algorithms impose non-negativity constraints on the elements of $\mathbf{Y}$ and $\mathbf{W}$ for reasons of biological plausibility, since neural firing rates are positive and since synapses can not change between being excitatory and being inhibitory. In general, such algorithms employ different forms of competitive learning, in which nodes compete to be active in response to each input pattern. Such competition can be implemented using a number of different forms of lateral inhibition. In this paper, a competitive learning algorithm in which nodes can inhibit the inputs to other nodes is used. This form of inhibition has been termed dendritic inhibition or pre-integration lateral inhibition. The full dendritic inhibition model (Spratling and Johnson, 2002, 2003) allows negative synaptic weight values. However, in order to make a fairer comparison with NMF, a version of the dendritic inhibition algorithm in which all synaptic weights are restricted to be non-negative (by clipping the weights at zero) is also used in the experiments described here. The full model will be referred to by the acronym `di` while the non-negative version will be referred to as `nndi` (see Table 1).

In a neural network model, an input image ($\vec{\mathbf{x}}_k$) generates activity ($\vec{\mathbf{y}}_k$) in the nodes of a neural network such that $\vec{\mathbf{y}}_k = \mathrm{f}(\mathbf{W}, \vec{\mathbf{x}}_k)$. In the dendritic inhibition model, the activation of each individual node is calculated as:

$$y_{jk} = \mathbf{W}\vec{\mathbf{x}}'_{kj}$$

where $\vec{\mathbf{x}}'_{kj}$ is an inhibited version of the input activations ($\vec{\mathbf{x}}_k$) that can differ for each node. The values of $\vec{\mathbf{x}}'_{kj}$ are calculated as:

$$x'_{ikj} = x_{ik}\left(1 - \alpha \max_{\substack{r=1 \\ (r\neq j)}}^{n}\left\{\frac{w_{ri}}{\max_{q=1}^{m}\{w_{rq}\}}\frac{y_{rk}}{\max_{q=1}^{n}\{y_{qk}\}}\right\}\right)^{+},$$

where α is a scale factor controlling the strength of lateral inhibition, and $(v)^+$ is the positive half-rectified value of v. The steady-state values of y_{jk} were found by iteratively applying the above two equations, increasing the value of α from 0 to 6 in steps of 0.25, while keeping the input image fixed.

The above activation function implements a form of lateral inhibition in which each neuron can inhibit the *inputs* to other neurons. The strength with which a node inhibits an input to another node is proportional to the strength of the afferent weight the inhibiting node receives from that particular input. Hence, if a node is strongly activated by the overall stimulus and it has a strong synaptic weight to a certain feature of that stimulus, then it will inhibit other nodes from responding to that feature. On the other hand, if an active node receives a weak weight from a feature then it will only weakly inhibit other nodes from responding to that feature. In this manner, each node can selectively 'block' its preferred inputs from activating other nodes, but does not inhibit other nodes from responding to distinct stimuli.

All weight values were initialised to random values chosen from a Gaussian distribution with a mean of $\frac{1}{m}$ and a standard deviation of $0.001\frac{n}{m}$. This small degree of noise on the initial weight values was sufficient to cause bifurcation of the activity values, and thus to cause differentiation of the receptive fields of different nodes through activity-dependent learning. Previous results with this algorithm have been produced using noise (with a similarly small magnitude) applied to the node activation values. Using noise applied to the node activations, rather than the weights, produces similar results to those reported in this article. However, noise was only applied to the initial weight values, and not to the node activations, here to provide a fairer comparison to the deterministic NMF algorithms. Note that a node which still has its initial weight values (*i.e.*, a node which has not had its weights modified by learning) is described as 'uncommitted'.

Synaptic weights were adjusted using the following learning rule (applied to weights with values greater than or equal to zero):

$$\Delta w_{ji} = \frac{(x_{ik} - \bar{x_k})}{\sum_{p=1}^{m} x_{pk}} \left(y_{jk} - \bar{y_k}\right)^{+}. \tag{1}$$

Here $\bar{x_k}$ is the mean value of the pixels in the training image (*i.e.*, $\bar{x_k} = \frac{1}{m}\sum_{i=1}^{m} x_{ik}$), and $\bar{y_k}$ is the mean of the output activations (*i.e.*, $\bar{y_k} = \frac{1}{n}\sum_{j=1}^{n} y_{jk}$). Following learning, synaptic weights were clipped at zero such that $w_{ji} = (w_{ji})^+$ and were normalised such that $\sum_{i=1}^{m}(w_{ji})^+ \leq 1$.

This learning rule encourages each node to learn weights selective for a set of coactive inputs. This is achieved since when a node is more active than average it increases its synaptic weights to active inputs and decreases its weights to inactive inputs. Hence, only sets of inputs which are consistently coactive will generate strong afferent weights. In addition, the learning rule is designed to ensure that different nodes can represent stimuli which share input features in common (*i.e.*, to allow the network to represent overlapping patterns). This is achieved by rectifying the

post-synaptic term of the rule so that no weight changes occur when the node is less active than average. If learning was not restricted in this way, whenever a pattern was presented all nodes which represented patterns with overlapping features would reduce their weights to these features.

For the algorithm `di`, but not for algorithm `nndi`, the following learning rule was applied to weights with values less than or equal to zero:

$$\Delta w_{ji} = \frac{\left(x'_{ikj} - 0.5x_{ik}\right)^-}{\sum_{p=1}^{n} y_{pk}} \left(y_{jk} - \bar{y_k}\right). \quad (2)$$

Here $(v)^-$ is the negative half-rectified value of v. Negative weights were clipped at zero such that $w_{ji} = (w_{ji})^-$ and were normalised such that $\sum_{i=1}^{m} (w_{ji})^- \geq -1$. Note that for programming convenience a single synapse is allowed to take either excitatory or inhibitory weight values. In a more biologically plausible implementation two separate sets of afferent connections could be used: the excitatory ones being trained using equation 1 and the inhibitory ones being trained using a rule similar to equation 2.

The negative weight learning rule has a different form from the positive weight learning rule as it serves a different purpose. The negative weights are used to ensure that each image component is represented by a distinct node rather than by the partial activation of multiple nodes each of which represents an overlapping image component. A full explanation of this learning rule is provided together with a concrete example of its function in section 3.3.

3. Results

In each of the experiments reported below, all the algorithms listed in Table 1 were applied to learning the components in a set of p training images. The average number of components that were correctly identified over the course of 25 trials was recorded. A new set of p randomly generated images were created for each trial. In each trial the NMF algorithms were trained until the total sum of the absolute difference in the objective function, between successive epochs, had not changed by more than 0.1% of its average value in 100 epochs, or until 2000 epochs had been completed. One epoch is when the method has been trained on all p training patterns. Hence, an epoch equals one update to the factors in an NMF algorithm, and p iterations of the neural network algorithm during which each individual image in the training set is presented once as input to the network. The neural network algorithms were trained for 10000 iterations. Hence, the NMF algorithms were each trained for at least 100 epochs, while the neural network algorithms were trained for at most 100 epochs (since the value of p was 100 or greater).

3.1 Standard Bars Problem

The bars problem (and its variations) is a benchmark task for the learning of independent image features (Földiák, 1990; Saund, 1995; Dayan and Zemel, 1995; Hinton et al., 1995; Harpur and Prager, 1996; Hinton and Ghahramani, 1997; Frey et al., 1997; Fyfe, 1997b; Charles and Fyfe, 1998; Hochreiter and Schmidhuber, 1999; Meila and Jordan, 2000; Plumbley, 2001; O'Reilly, 2001; Ge and Iwata, 2002; Lücke and von der Malsburg, 2004). In the standard version of the bars problem, as defined by Földiák (1990), training data consists of 8 by 8 pixel images in which each of the 16 possible (one-pixel wide) horizontal and vertical bars can be present with a probability of $\frac{1}{8}$. Typical examples of training images are shown in Figure 2.

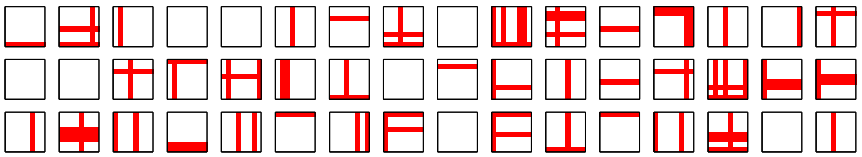


Figure 2: Typical training patterns for the standard bars problem. Horizontal and vertical bars in 8x8 pixel images are independently selected to be present with a probability of $\frac{1}{8}$. Dark pixels indicate active inputs.

In the first experiment the algorithms were trained on the standard bars problem. Twenty-five trials were performed with $p = 100$, and, with $p = 400$. The number of components that each algorithm could learn (n) was set to 32. Typical examples of the (generative) weights learnt by each NMF algorithm (*i.e.*, the columns of **A** reshaped as 8 by 8 pixel images) and of the (recognition) weights learnt by the neural network algorithms (*i.e.*, the rows of **W** reshaped as 8 by 8 pixel images) are shown in Figure 3. It can be seen that the NMF algorithms tend to learn a redundant representation, since the same bar can be represented multiple times. In contrast, the neural network algorithms learnt a much more compact code, with each bar being represented by a single node. It can also be seen that many of the NMF algorithms learnt representations of bars in which pixels were missing. Hence, these algorithms learnt random pieces of image components rather than the image components themselves. In contrast, the neural network algorithms (and algorithms `nnsc`, `nmfsc(A)`, and `nmfsc(A&Y)`) learnt to represent complete image components.

To quantify the results, the following procedure was used to determine the number of bars represented by each algorithm in each trial. For each node, the sum of the weights corresponding to each row and column of the input image was calculated. A node was considered to represent a particular bar if the total weight corresponding to that bar was twice that of the sum of the weights for any other row or column and if the minimum weight in the row or column corresponding to that bar was greater than the mean of all the (positive) weights for that node. The number of unique bars represented by at least one basis vector, or one node of the network, was calculated, and this value was averaged over the 25 trials. The mean number of bars (*i.e.*, image components) represented by each algorithm is shown in Figure 4a. Good performance requires both accuracy (finding all the image components) and reliability (doing so across all trials). Hence, the average number of bars represented needs to be close to 16 for an algorithm to be considered to have performed well. It can be seen that when $p = 100$, most of the NMF algorithms performed poorly on this task. However, algorithms `nmfsc(A)` and `nmfsc(A&Y)` produced good results, representing an average of 15.7 and 16 components respectively. This compares favourably to the neural network algorithms, with `nndi` representing 14.6 and `di` representing 15.9 of the 16 bars. When the number of images in the training set was increased to 400 this improved the performance of certain NMF algorithms, but lead to worse performance in others. For $p = 400$, every image component in every trial was found by algorithms `nnsc`, `nmfsc(A)`, `nmfsc(A&Y)` and `di`.

In the previous experiment there were significantly more nodes, or basis vectors, than was necessary to represent the 16 underlying image components. The poor performance of certain algorithms could thus potentially be due to over-fitting. The experiment was thus repeated using 16 nodes or

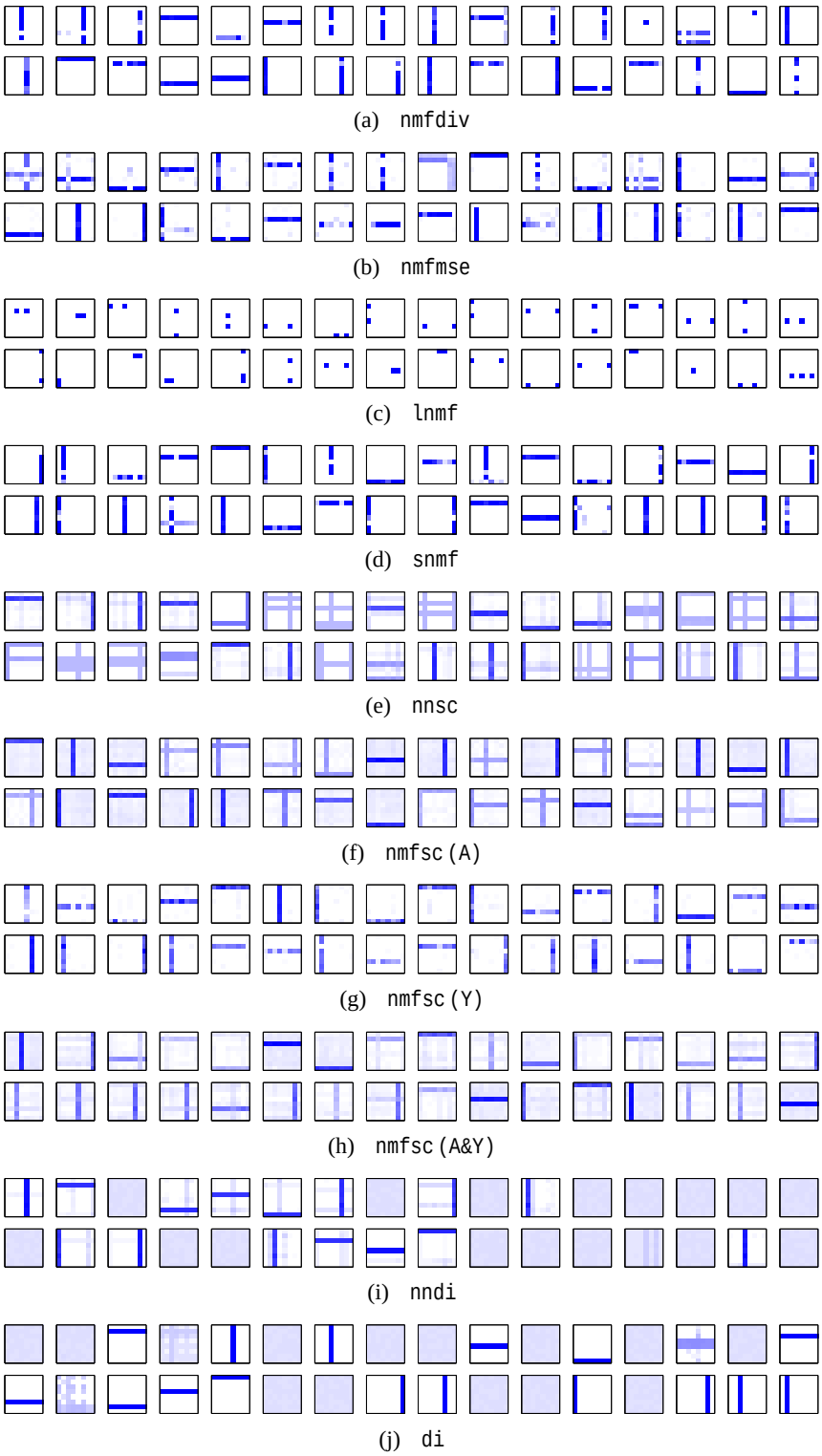
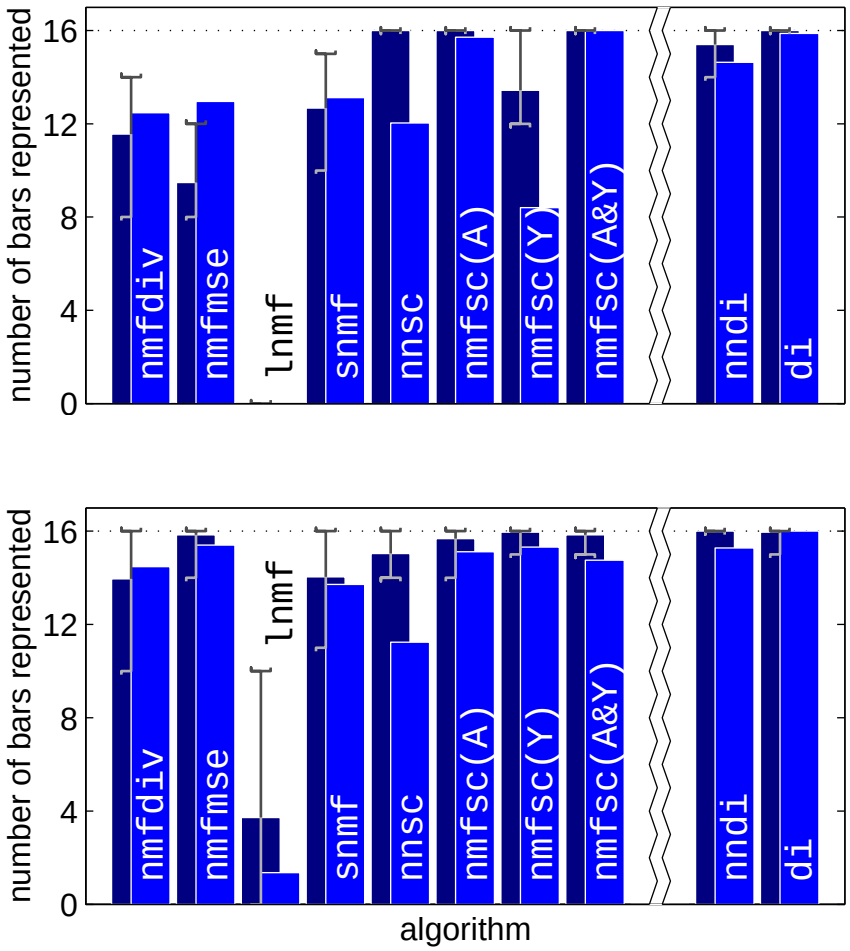


Figure 3: Typical generative or recognition weights learnt by 32 basis vectors or nodes when each algorithm was trained on the standard bars problem. Dark pixels indicate strong weights.



(b)

Figure 4: Performance of each algorithm when trained on the standard bars problem. For (a) 32, and (b) 16 basis vectors or nodes. Each bar shows the mean number of components correctly identified by each algorithm when tested over 25 trials. Results for different training set sizes are shown: the lighter, foreground, bars show results for $p = 100$, and the darker, background, bars show results for $p = 400$. Error bars show best and worst performance, across the 25 trials, when $p = 400$.

basis vectors. The results of this experiment are shown in Figure 4b. It can be seen that while the performance of some NMF algorithms is improved, the performance of others becomes slightly worse. Only NMF algorithms `nmfsc(A)` and `nmfsc(A&Y)` reliably find nearly all the bars with both 16 and 32 basis vectors. In contrast, the performance of the neural network algorithms is unaffected by changing the number of nodes. Such behaviour is desirable since it is generally not known in advance how many components there are. Hence, a robust algorithm needs to be able to correctly

learn image components with an excess of nodes. Across both these experiment the `di` version of the neural network algorithm performs as well as or better than any of the NMF algorithms.

3.2 Bars Problems Without Occlusion

To determine the effect of occlusion on the performance of each algorithm further experiments were performed using versions of the bars problem in which no occlusion occurs. Firstly, a linear version of the standard bars problem was used. Similar tasks have been used by Plumbley (2001) and Hoyer (2002). In this task, pixel values are combined additively at points of overlap between horizontal and vertical bars. As with the standard bars problem, training data consisted of 8 by 8 pixel images in which each of the 16 possible (one-pixel wide) horizontal and vertical bars could be present with a probability of $\frac{1}{8}$. All the algorithms were trained with $p = 100$ and with $p = 400$ using 32 nodes or basis vectors. The number of unique bars represented by at least one basis vector, or one node of the network, was calculated, and this value was averaged over 25 trials. The mean number of bars represented by each algorithm is shown in Figure 5a.

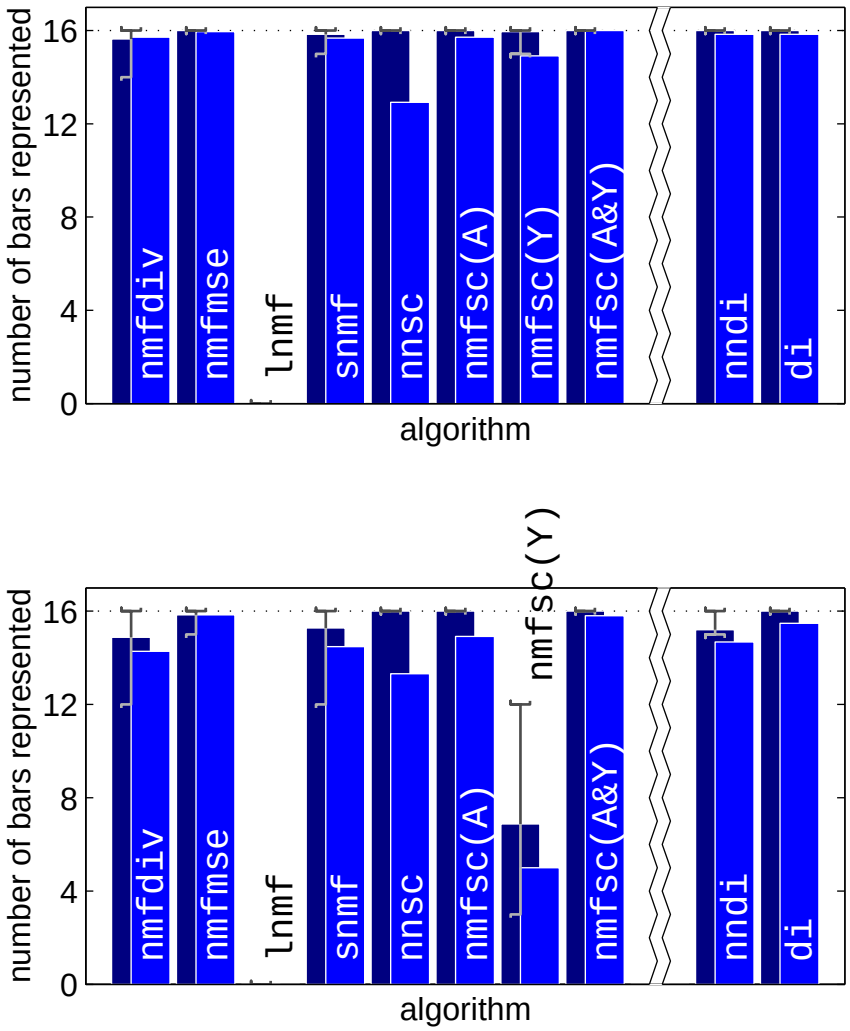
Another version of the bars problem in which occlusion is avoided is one in which horizontal and vertical bars do not co-occur. Similar tasks have been used by Hinton et al. (1995); Dayan and Zemel (1995); Frey et al. (1997); Hinton and Ghahramani (1997) and Meila and Jordan (2000). In this task, an orientation (either horizontal or vertical) was chosen with equal probability for each training image. The eight (one-pixel wide) bars of that orientation were then independently selected to be present with a probability of $\frac{1}{8}$. All the algorithms were trained with $p = 100$ and with $p = 400$ using 32 nodes or basis vectors. The number of unique bars represented by at least one basis vector, or one node of the network, was calculated, and this value was averaged over 25 trials. The mean number of bars represented by each algorithm is shown in Figure 5b.

For both experiments using training images in which occlusion does not occur, the performance of most of the NMF algorithms is improved considerably in comparison to the standard bars problem. The neural network algorithms also reliably learn the image components as with the standard bars problem.

3.3 Bars Problem with More Occlusion

To further explore the effects of occlusion on the learning of image components a version of the bars problem with double-width bars was used. Training data consisted of 9 by 9 pixel images in which each of the 16 possible (two-pixel wide) horizontal and vertical bars could be present with a probability $\frac{1}{8}$. The image size was increased by one pixel to keep the number of image components equal to 16 (as in the previous experiments). In this task, as in the standard bars problem, perpendicular bars overlap; however, the proportion of overlap is increased. Furthermore, neighbouring parallel bars also overlap (by 50%). Typical examples of training images are shown in Figure 6.

Each algorithm was trained on this double-width bars problem with $p = 400$. The number of components that each algorithm could learn (n) was set to 32. Typical examples of the (generative) weights learnt by each NMF algorithm and of the (recognition) weights learnt by the neural network algorithms are shown in Figure 7. It can be seen that the majority of the NMF algorithms learnt redundant encodings in which the basis vectors represent parts of image components rather than complete components. In contrast, the neural network algorithms (and the `nnsc` algorithm) learnt to represent complete image components.



(b)

Figure 5: Performance of each algorithm when trained on (a) the linear bars problem, and (b) the one-orientation bars problem, with 32 basis vectors or nodes. Each bar shows the mean number of components correctly identified by each algorithm when tested over 25 trials. Results for different training set sizes are shown: the lighter, foreground, bars show results for $p = 100$, and the darker, background, bars show results for $p = 400$. Error bars show best and worst performance, across the 25 trials, when $p = 400$.

The following procedure was used to determine the number of bars represented by each algorithm in each trial. For each node, the sum of the weights corresponding to every double-width bar was calculated. A node was considered to represent a particular bar if the total weight corresponding to that bar was 1.5 times that of the sum of the weights for any other bar and if the minimum weight of any pixel forming part of that bar was greater than the mean of all the (positive) weights

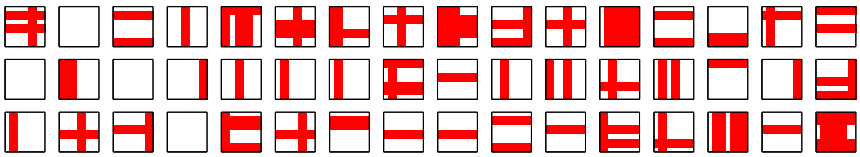


Figure 6: Typical training patterns for the double-width bars problem. Two pixel wide horizontal and vertical bars in a 9x9 pixel image are independently selected to be present with a probability of $\frac{1}{8}$. Dark pixels indicate active inputs.

for that node. The number of unique bars represented by at least one basis vector, or one node of the network, was calculated. The mean number of two-pixel-wide bars, over the 25 trials, represented by each algorithm is shown in Figure 8a. This set of training images could be validly, but less efficiently, represented by *18* one-pixel-wide bars. Hence, the mean number of one-pixel-wide bars, represented by each algorithm was also calculated and this data is shown in Figure 8b. The number of one-pixel-wide bars represented was calculated using the same procedure used to analyse the previous results (as stated in section 3.1).

It can be seen that the majority of the NMF algorithms perform very poorly on this task. Most fail to reliably represent either double- or single-width bars. However, algorithms `nnsc`, `nmfsc(A)` and `nmfsc(A&Y)` do succeed in learning the image components. The dendritic inhibition algorithm, with negative weights, also succeeds in identifying all the double-width bars in most trials. However, the non-negative version of this algorithm performs less well. This result illustrates the need to allow negative weight values in this algorithm in order to robustly learn image components. Negative weights are needed to disambiguate a real image component from the simultaneous presentation of partial components. For example, consider part of the neural network that is receiving input from four pixels that are part of four separate, neighbouring, columns of the input image (as illustrated in Figure 9). Assume that two nodes in the output layer (nodes 1 and 3) have learnt weights that are selective to two neighbouring, but non-overlapping, double-width bars (bars 1 and 3). When the double-width bar (bar 2) that overlaps these two represented bars is presented to the input, then the network will respond by partially activating nodes 1 and 3. Such a representation is reasonable since this input pattern could be the result of the co-activation of partially occluded versions of the two represented bars. However, if bar 2 recurs frequently then it is unlikely to be caused by the chance co-occurrence of multiple, partially occluded patterns, and is more likely to be an independent image component that should be represented in a similar way to the other components (*i.e.*, by the activation of a specific node tuned to that feature). One way to ensure that in such situations the network learns all image components is to employ negative synaptic weights. These negative weights are generated when a node is active and inputs, which are not part of the nodes' preferred input pattern, are inhibited. This can only occur when multiple nodes are co-active. If the pattern, to which this set of co-active nodes are responding, re-occurs then the negative weights will grow. When the negative weights are sufficiently large the response of these nodes to this particular pattern will be inhibited, enabling an uncommitted node to successfully compete to represent this pattern. On the other hand, if the pattern, to which this set of co-active nodes are responding, is just due to the

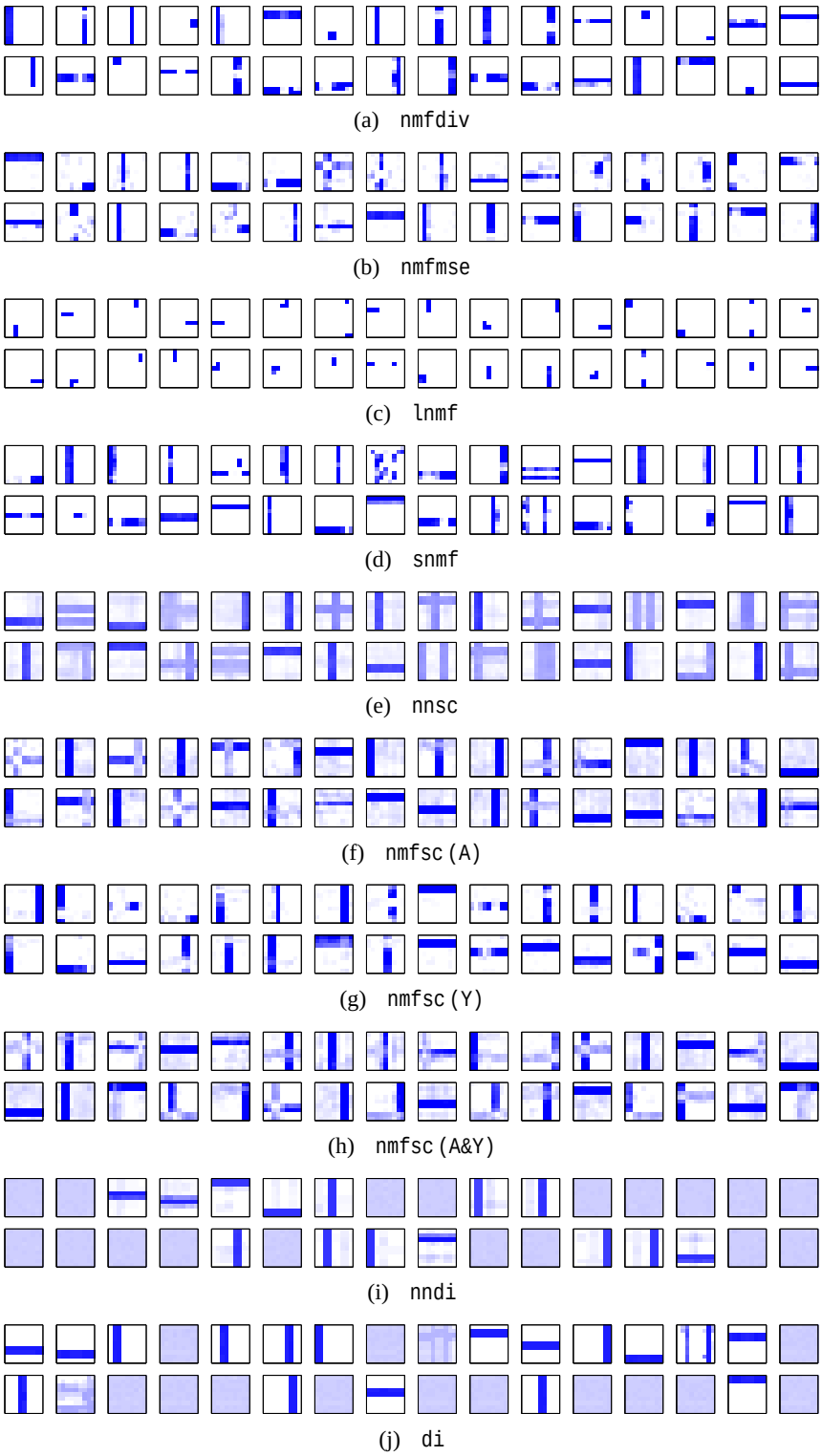


Figure 7: Typical generative or recognition weights learnt by 32 basis vectors or nodes when each algorithm was trained on the double-width bars problem. Dark pixels indicate strong weights.

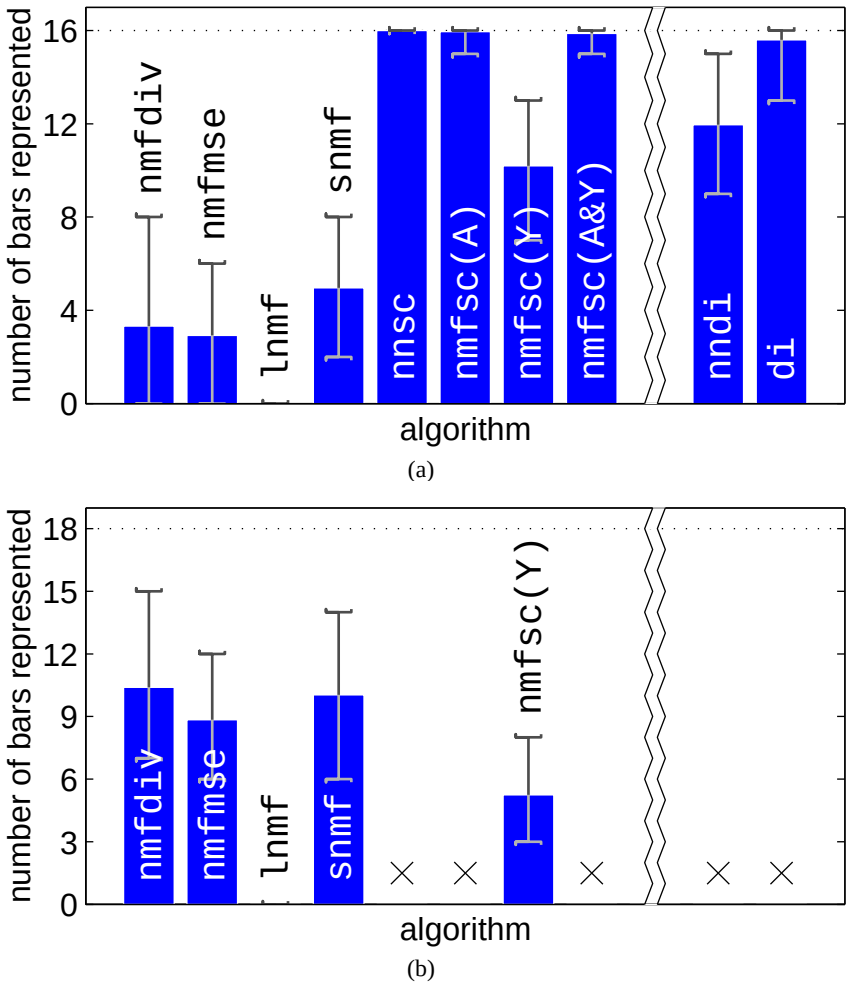


Figure 8: Performance of each algorithm when trained on the double-width bars problem, with 32 basis vectors or nodes, and $p = 400$. Each bar shows the mean number of components correctly identified by each algorithm when tested over 25 trials. (a) Number of double-width bars learnt. (b) Number of single-width bars learnt. Error bars show best and worst performance across the 25 trials. Note that algorithms that successfully learnt double-width bars—see (a)—do not appear in (b), but space is left for these algorithms in order to aid comparison between figures.

co-activation of independent input patterns then the weights will return toward zero on subsequent presentations of these patterns in isolation.

3.4 Bars Problem with Unequal Components

In each individual experiment reported in the previous sections, every component of the training images has been exactly equal in size to every other component and has occurred with exactly the

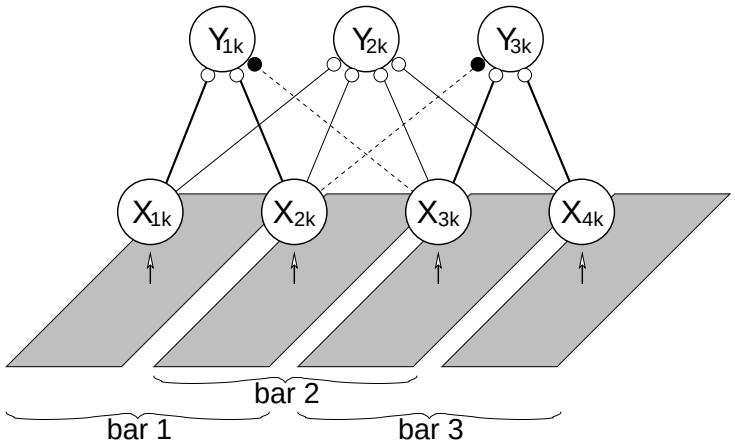


Figure 9: An illustration of a the role of negative weights in the dendritic inhibition algorithm. Nodes are shown as large circles, excitatory synapses as small open circles and inhibitory synapses as small filled circles. Nodes $\mathbf{Y}_{1k}$ and $\mathbf{Y}_{3k}$ are selective for the double-width bars 1 and 3 respectively. The occurrence in an image of the double-width bar 'bar 2' would be indicated by both nodes $\mathbf{Y}_{1k}$ and $\mathbf{Y}_{3k}$ being active at half-strength. However, if bar 2 occurs sufficiently frequently, the negative afferent weights indicated will develop which will suppress this response and enable the uncommitted node ($\mathbf{Y}_{2k}$) to compete to respond to this pattern. Note that each bar would activate two columns each containing eight input nodes, but only one input node per column is shown for clarity.

same probability. This is unlikely to be the case in real-world object recognition tasks. In the real-world, different image features can be radically different in size and can be encountered with very different frequency. For example, important features in face recognition include both the eyes and the hair (Sinha and Poggio, 1996; Davies et al., 1979) which can vary significantly in relative size. Furthermore, we effortlessly learn to recognise both immediate family members (who may be seen many times a day) and distant relatives (who may be seen only a few times a year).

To provide a more realistic test, a new variation on the bars problem was used. In this version, training data consisted of 16 by 16 pixel images. Image components consisted of seven one-pixel wide bars and one nine-pixel wide bar in both the horizontal and vertical directions. Hence, as in previous experiments, there were eight image components at each orientation and 16 in total. Parallel bars did not overlap, however, the proportion of overlap between the nine-pixel wide bars and all other perpendicular bars was large, while the proportion of overlap between perpendicular one-pixel wide bars was less than in the standard bars problem. Each horizontal bar was selected to be present in an image with a probability of $\frac{1}{8}$ while vertical bars occurred with a probability of $\frac{1}{32}$. Hence, in this test case half the underlying image components occurred at a different frequency to the other half and two of the components were a different size to the other 14.

Each algorithm was trained on this 'unequal' bars problem with $p = 400$. The number of components that each algorithm could learn (n) was set to 32. The mean number of bars, over 25 trials, represented by each algorithm is shown in Figure 10. It can be seen that none of the NMF algorithms succeeded in reliably identifying all the image components in this task. Learning to represent

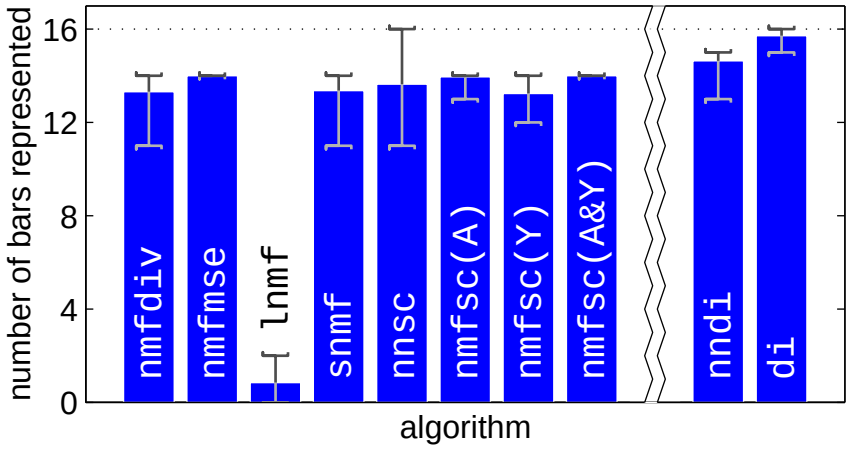


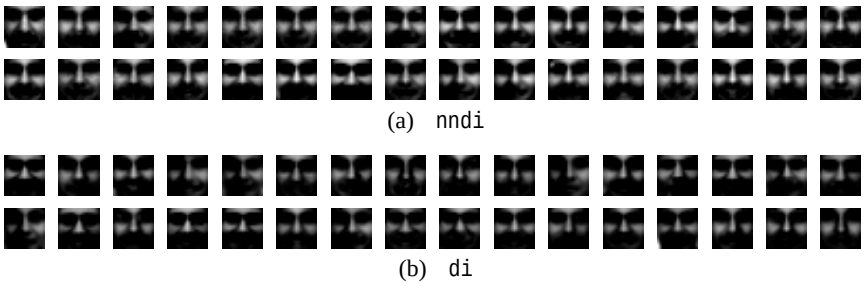
Figure 10: Performance of each algorithm when trained on the unequal bars problem, with 32 basis vectors or nodes, and $p = 400$. Each bar shows the mean number of components correctly identified by each algorithm when tested over 25 trials. Error bars show best and worst performance across the 25 trials.

the two large components appears to be a particular problem for all these algorithms, but the cause for this may differ between algorithms. For most NMF algorithms, the underlying linear model fails when occlusion is significant, as is the case for the two nine-pixel wide patterns. However, for the NMF algorithms that find non-negative factors with an additional constraint on the sparseness of the basis vectors (*i.e.*, `nmfsc(A)` and `nmfsc(A&Y)`) an alternative cause may be the imposition of a constraint that requires learnt image components to be a similar size, which is not the case for the components used in this task. The neural network algorithm with positive weights (`nndi`) produced results that are only marginally better than the NMF algorithms. This algorithm also fails to reliably learn the two large components due to the large overlap between them. In contrast, the dendritic inhibition neural network algorithm with negative weights (`di`), succeeded in identifying all the bars in most trials. As for the previous experiment with double-width bars, this result illustrates the need to allow negative weight values in this algorithm in order to robustly learn image components. Algorithm `di` learnt every image component in 18 of the 25 trials. In contrast, only one NMF algorithm (`nnsc`) managed to find all the image components, but it only did so in a single trial.

3.5 Face Images

Previously, NMF algorithms have been tested using a training set that consists of images of faces (Lee and Seung, 1999; Hoyer, 2004; Li et al., 2001; Feng et al., 2002). When these training images are taken from the CBCL face database,[1] algorithm `nmfdiv` learns basis vectors that correspond to localised image parts (Lee and Seung, 1999; Hoyer, 2004; Feng et al., 2002). However, when

1. CBCL Face Database #1, MIT Center For Biological and Computation Learning, `http://www.ai.mit.edu/projects/cbcl`.



(a) nndi

(b) di

Figure 11: (a) and (b) Typical recognition weights learnt by 32 nodes when both versions of the dendritic inhibition algorithm were trained on the CBCL face database. Light pixels indicate strong weights.

applied to the ORL face database,[2] algorithm nmfmse learns global, rather than local, image features (Li et al., 2001; Hoyer, 2004; Feng et al., 2002). In contrast, algorithm lnmf learns localised representations of the ORL face database (Feng et al., 2002; Li et al., 2001). Algorithm nmfsc can find either local or global representations of either set of face images with appropriate values for the constraints on the sparseness of the basis vectors and activations. Specifically, nmfsc learns localised image parts when constrained to produce highly sparse basis images, but learns global image features when constrained to produce basis images with low sparseness or if constrained to produce highly sparse activations (Hoyer, 2004). Hence, NMF algorithms can, in certain circumstances, learn localised image components, some of which appear to roughly correspond to parts of the face, but others of which are arbitrary, but localised, blobs. Essentially the NMF algorithms select a subset of the pixels which are simultaneously active across multiple images to be represented by a single basis vector. The same behaviour is observed in the bars problems, reported above, where a basis vector often corresponds to a random subset of pixels along a row or column of the image rather than representing an entire bar. Such arbitrary image components are not meaningful representations of the image data.

In contrast when the dendritic inhibition neural network is trained on face images, it learns global representations. Figure 11a and Figure 11b show the results of training nndi and di, for 10000 iterations, on the 2429 images in the CBCL face database. Parameters were identical to those used for the bars problems, except the weights were initialised to random values chosen from a Gaussian distribution with a larger standard deviation ($0.005\frac{n}{m}$) as this was found necessary to cause bifurcation of activity values. In both cases, each node has learnt to represent an average (or prototype) of a different subset of face images. When presented with highly overlapping training images the neural network algorithm will learn a prototype consisting of the common features between the different images (Spratling and Johnson, 2006). When presented with objects that have less overlap, the network will learn to represent the individual exemplars (Spratling and Johnson, 2006). These two forms of representation are believed to support perceptual categorisation and object recognition (Palmeri and Gauthier, 2004).

2. The ORL Database of Faces, AT&T Laboratories Cambridge, http://www.cl.cam.ac.uk/Research/DTG/attarchive/facedatabase.html.

4. Discussion

The NMF algorithm `lnmf` imposes the constraint that the basis vectors be orthogonal. This means that image components may not overlap and, hence, results in this algorithm's failure across all versions of the bars task. In all these tasks the underlying image components are overlapping bars patterns (even if they do not overlap in any single training image, as is the case with the one-orientation bars problem).

NMF algorithms which find non-negative factors without other constraints (*i.e.*, `nmfdiv` and `nmfmse`) generally succeed in identifying the underlying components of images when trained using images in which there is no occlusion (*i.e.*, on the linear and one-orientation bars problems). However, these algorithms fail when occlusion does occur between image components, and performance gets worse as the degree of occlusion increases. Hence, these algorithms fail to learn many image features in the standard bars problem and produce even worse performance when tested on the double-width bars problem.

NMF algorithms that find non-negative factors with an additional constraint on the sparseness of the activations (*i.e.*, `snmf`, `nnsc`, and `nmfsc(Y)`) require that the rows of $\mathbf{Y}$ have a particular sparseness. Such a constraint causes these algorithms to learn components that are present in a certain fraction of the training images (*i.e.*, each factor is required to appear with a similar frequency). Such a constraint can overcome the problems caused by occlusion and enable NMF to identify components in training images where occlusion occurs. For example, `nnsc` produced good results on the double-width bars problem. Given sufficient training data, `nnsc` also reliably finds nearly all the image components in all experiments except for the standard bars test when $n = 16$ and the unequal bars problem. However, `nmfsc(Y)` fails to produce consistently good results across experiments. This algorithm only reliably found all the image components for the standard bars problem when $n = 16$ and for the linear bars problem. Despite constraining the sparseness of the activations, algorithm `snmf` produced poor results in all experiments except for the linear bars problem.

The NMF algorithm that finds non-negative factors with an additional constraint on the sparseness of the basis vectors (*i.e.*, `nmfsc(A)`) requires that the columns of $\mathbf{A}$ have a particular sparseness. Such a constraint causes this algorithm to learn components that have a certain fraction of pixels with values greater than zero (*i.e.*, all factors are required to be a similar size). This algorithm produces good results across all the experiments except the unequal bars problem. Constraining the sparseness of the basis vectors thus appears to overcome the problems caused by occlusion and enable NMF to identify components in training images where occlusion occurs. However, this constraint may itself prevent the algorithm from identifying image components which are a different size from that specified by the sparseness parameter. The NMF algorithm that imposes constraints on both the sparseness of the activations and the sparseness of the basis vectors (*i.e.*, `nmfsc(A&Y)`) produces results similar to those produced by `nmfsc(A)`.

The performance of the `nmfsc` algorithm depends critically on the particular sparseness parameters that are chosen. As can be seen from figure 12, performance on a specific task can vary from finding every component in every trial, to failure to find even a single component across all trials. While appropriate values of sparseness constraint can enable the NMF algorithm to overcome inherent problems associated with the non-linear superposition of image components, inappropriate values of sparseness constraint will prevent the identification of factors that occur at a different frequency, or that are a different size, to that specified by the sparseness parameter chosen. This is particularly a problem when the frequency and size of different image components varies within a

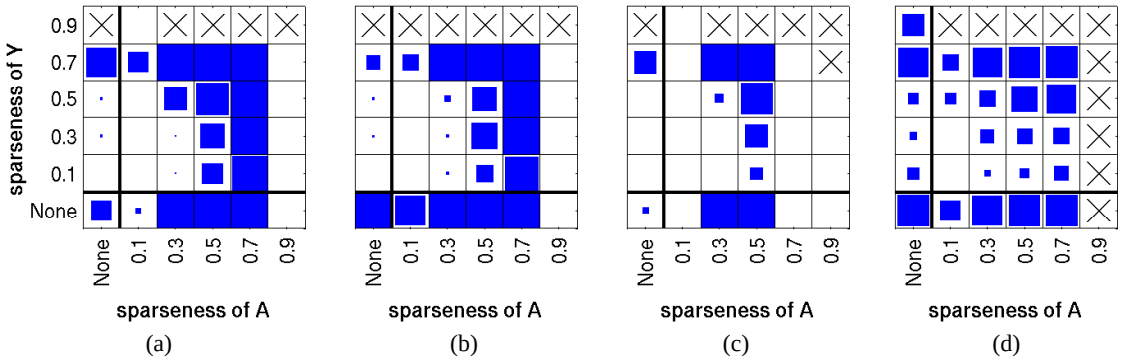


Figure 12: Performance of the `nmfsc` algorithm across the range of possible combinations of sparseness parameters. For (a) the standard bars problem, (b) the one-orientation bars problem, (c) the double-width bars problem, and (d) the unequal bars problem. In each case, $n = 32$ and $p = 400$. The sparseness of $\mathbf{Y}$ varies along the y-axis and the sparseness of $\mathbf{A}$ varies along the x-axis of each plot. Since the sparseness constraints are optional, 'None' indicates where no constraint was imposed. The length of the edge of each filled box is proportional to the mean number of bars learnt over 25 trials for that combination of parameters. Perfect performance would be indicated by a box completely filling the corresponding element of the array. 'X' marks combinations of parameter values for which the algorithm encountered a division-by-zero error and crashed.

single task. Hence, the `nmfsc` algorithm was unable to identify all the components in the unequal bars problem with any combination of sparseness parameters (figure 12d). In fact, no NMF algorithm succeeded in this task: either because the linear NMF model could not deal with occlusion or because the algorithm imposed sparseness constraints that could not be satisfied for all image components.

The parameter search shown in figure 12 was performed in order to select the parameter values that would produce the best overall results across all the tasks used in this paper for algorithms `nmfsc(A)`, `nmfsc(Y)`, and `nmfsc(A&Y)`. However, in many real-world tasks the user may not know what the image components should look like, and hence, it would be impossible to search for the appropriate parameter values. The `nmfsc` algorithm is thus best suited to tasks in which all components are either a similar size or occur at a similar frequency, and for which this size/frequency is either known *a priori* or the user knows what the components should look like and is prepared to search for parameters that enable these components to be 'discovered' by the algorithm.

Sparseness constraints are also often employed in neural network algorithms. However, in such recognition models, sparseness constraints usually limit the number of nodes that are simultaneously active in response to an input image (Földiák and Young, 1995; Olshausen and Field, 2004). This is equivalent to constraining the sparseness in the *columns* of $\mathbf{Y}$ (rather than the rows of $\mathbf{Y}$, or the columns of $\mathbf{A}$, as has been constrained in the NMF algorithms). Any constraints that impose restrictions on the number of active nodes will prevent a neural network from accurately and completely representing stimuli (Spratling and Johnson, 2004). Hence, such sparseness constraints

should be avoided. The dendritic inhibition model succeeds in learning representations of elementary image features without such constraints. However, to accurately represent image features that overlap, it is necessary for negative weight values to be allowed. This algorithm (`di`) produced the best overall performance across all the experiments performed here. This is achieved because this algorithm does not falsely assume that image composition is a linear process, nor does it impose constraints on the expected size or frequency of occurrence of image components. The dendritic inhibition algorithm thus provides an efficient, on-line, algorithm for finding image components.

When trained on images that are composed of elementary features, such as those used in the bars problems, algorithm `di` reliably and accurately learns representations of the underlying image features. However, when trained on images of faces, algorithm `di` learns holistic representations. In this case, large subsets of the training images contain virtually identical patterns of pixel values. These re-occurring, holistic patterns, are learnt by the dendritic inhibition algorithm. In contrast, the NMF algorithms (in certain circumstances) form distinct basis vectors to represent pieces of these recurring patterns. The separate representation of sub-patterns is due to constraints imposed by the algorithms and is not based on evidence contained in the training images. Hence, while these constraints make it appear that NMF algorithms have learnt face parts, these algorithms are representing arbitrary parts of larger image features. This is demonstrated by the results generated when the NMF algorithms are applied to the bars problems. In these cases, each basis vector often corresponds to a random subset of pixels along a row or column of the image rather than representing an entire bar. Such arbitrary image components are not meaningful representations of the image data. Rather than relying on a subjective assessment of the quality of the components that are learnt, the bars problems that are the main focus of this paper, provide a quantitative test of the accuracy and reliability with which elementary image features are discovered. Since the underlying image components are known, it is possible to compare the components learnt with the known features from which the training images were created. These results demonstrate that when the training images are actually composed of elementary features, NMF algorithms can fail to learn the underlying image components, whereas, the dendritic inhibition algorithm reliably and accurately does so.

Intuitively, the dendritic inhibition algorithm works because the learning rule causes nodes to learn re-occurring patterns of pre-synaptic activity. As an afferent weight to a node increases, so does the strength with which that node can inhibit the corresponding input activity received by all other nodes. This provides strong competition for specific patterns of inputs and forces different nodes to learn distinct image components. However, because the inhibition is specific to a particular set of inputs, nodes do not interfere with the learning of distinct image components by other nodes. Unfortunately, the operation of this algorithm has not so far been formulated in terms of the optimisation of an objective function. It is hoped that the empirical performance of this algorithm will prompt the development of such a mathematical analysis.

5. Conclusions

Non-negative matrix factorisation employs non-negativity constraints in order to model the physics of image formation, and it has been claimed that this makes NMF particularly suited to learning meaningful representations of image data (Lee and Seung, 1999; Feng et al., 2002; Liu et al., 2003; Liu and Zheng, 2004; Li et al., 2001). However, by employing a linear model, NMF fails to take into account another important factor of image composition, namely the presence of occlusion. Hence,

despite the claims, most NMF algorithms fail to reliably identify the underlying components of images, even in simple, artificial, tasks like those investigated here. These limitations can be overcome by imposing additional constraints on the sparseness of the factors that are found. However, to employ such constraints requires *a priori* knowledge, or trial-and-error, to find appropriate parameter values and can result in failure to identify components that violate the imposed constraint. In contrast, a neural network algorithm, employing a non-linear activation function, can reliably and accurately learn image components. This neural network algorithm is thus more likely to provide a robust method of learning image components suitable for object recognition.

Acknowledgments

This work was funded by the EPSRC through grant number GR/S81339/01. I would like to thank Patrik Hoyer for making available the MATLAB code that implements the NMF algorithms and which has been used to generate the results presented here.

References

S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3:277–90, 1990.

H. B. Barlow. Conditions for versatile learning, Helmholtz's unconscious inference, and the task of perception. *Vision Research*, 30:1561–71, 1990.

H. B. Barlow. The neuron doctrine in perception. In M. S. Gazzaniga, editor, *The Cognitive Neurosciences*, chapter 26. MIT Press, Cambridge, MA, 1995.

D. Charles and C. Fyfe. Discovering independent sources with an adapted PCA neural network. In D. W. Pearson, editor, *Proceedings of the 2nd International ICSC Symposium on Soft Computing (SOCO97)*. NAISO Academic Press, 1997.

D. Charles and C. Fyfe. Modelling multiple cause structure using rectification constraints. *Network: Computation in Neural Systems*, 9(2):167–82, 1998.

G. M. Davies, J. W. Shepherd, and H. D. Ellis. Similarity effects in face recognition. *American Journal of Psychology*, 92:507–23, 1979.

P. Dayan and R. S. Zemel. Competition and multiple cause models. *Neural Computation*, 7:565–79, 1995.

T. Feng, S. Z. Li, H.-Y. Shum, and H. Zhang. Local non-negative matrix factorization as a visual representation. In *Proceedings of the 2nd International Conference on Development and Learning (ICDL02)*, pages 178–86, 2002.

P. Földiák. Adaptive network for optimal linear feature extraction. In *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks*, volume 1, pages 401–5, New York, NY, 1989. IEEE Press.

P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–70, 1990.

P. Földiák and M. P. Young. Sparse coding in the primate cortex. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 895–8. MIT Press, Cambridge, MA, 1995.

B. J. Frey, P. Dayan, and G. E. Hinton. A simple algorithm that discovers efficient perceptual codes. In M. Jenkin and L. R. Harris, editors, *Computational and Psychophysical Mechanisms of Visual Coding*. Cambridge University Press, Cambridge, UK, 1997.

C. Fyfe. Independence seeking negative feedback networks. In D. W. Pearson, editor, *Proceedings of the 2nd International ICSC Symposium on Soft Computing (SOCO97)*. NAISO Academic Press, 1997a.

C. Fyfe. A neural net for PCA and beyond. *Neural Processing Letters*, 6(1-2):33–41, 1997b.

X. Ge and S. Iwata. Learning the parts of objects by auto-association. *Neural Networks*, 15(2): 285–95, 2002.

G. Harpur and R. Prager. Development of low entropy coding in a recurrent network. *Network: Computation in Neural Systems*, 7(2):277–84, 1996.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268(5214):1158–61, 1995.

G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London. Series B*, 352(1358):1177–90, 1997.

S. Hochreiter and J. Schmidhuber. Feature extraction through LOCOCODE. *Neural Computation*, 11:679–714, 1999.

P. O. Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing XII: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557–65, 2002.

P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–69, 2004.

C. Jutten and J. Herault. Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1997.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–91, 1999.

D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, Cambridge, MA, 2001. MIT Press.

S. Z. Li, X. Hou, H. Zhang, and Q. Cheng. Learning spatially localized, parts-based representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR01)*, volume 1, pages 207–12, 2001.

W. Liu and N. Zheng. Non-negative matrix factorization based methods for object recognition. *Pattern Recognition Letters*, 25(8):893–7, 2004.

W. Liu, N. Zheng, and X. Lu. Non-negative matrix factorization for visual coding. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP03)*, volume 3, pages 293–6, 2003.

J. Lücke and C. von der Malsburg. Rapid processing and unsupervised learning in a model of the cortical macrocolumn. *Neural Computation*, 16(3):501–33, 2004.

M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5: 927–35, 1992.

B. A. Olshausen and D. J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–7, 2004.

R. C. O'Reilly. Generalization in interactive networks: The benefits of inhibitory competition and Hebbian learning. *Neural Computation*, 13(6):1199–1242, 2001.

T. J. Palmeri and I. Gauthier. Visual object understanding. *Nature Reviews Neuroscience*, 5(4): 291–303, 2004.

M. D. Plumbley. Adaptive lateral inhibition for non-negative ICA. In *Proceedings of the international Conference on Independent Component Analysis and Blind Signal Separation (ICA2001)*, pages 516–21, 2001.

E. Saund. A multiple cause mixture model for unsupervised learning. *Neural Computation*, 7(1): 51–71, 1995.

P. Sinha and T. Poggio. I think I know that face...,. *Nature*, 384(6608):404, 1996.

M. W. Spratling and M. H. Johnson. Pre-integration lateral inhibition enhances unsupervised learning. *Neural Computation*, 14(9):2157–79, 2002.

M. W. Spratling and M. H. Johnson. Exploring the functional significance of dendritic inhibition in cortical pyramidal cells. *Neurocomputing*, 52-54:389–95, 2003.

M. W. Spratling and M. H. Johnson. Neural coding strategies and mechanisms of competition. *Cognitive Systems Research*, 5(2):93–117, 2004.

M. W. Spratling and M. H. Johnson. A feedback model of perceptual learning and categorisation. *Visual Cognition*, 13(2):129–65, 2006.

Journal of Machine Learning Research 7 (2006) 817–854 Submitted 9/05; Revised 1/06; Published 5/06

Consistency and Convergence Rates of One-Class SVMs and Related Algorithms

Régis Vert REGIS.VERT@LRI.FR
Laboratoire de Recherche en Informatique
Bâtiment 490, Université Paris-Sud
91405, Orsay Cedex, France, and

Masagroup
24 Boulevard de l'Hôpital
75005 Paris, France

Jean-Philippe Vert JEAN-PHILIPPE.VERT@ENSMP.FR
Center for Computational Biology
Ecole des Mines de Paris
35 rue Saint Honoré
77300, Fontainebleau, France

Editor: Bernhard Schölkopf

Abstract

We determine the asymptotic behaviour of the function computed by support vector machines (SVM) and related algorithms that minimize a regularized empirical convex loss function in the reproducing kernel Hilbert space of the Gaussian RBF kernel, in the situation where the number of examples tends to infinity, the bandwidth of the Gaussian kernel tends to 0, and the regularization parameter is held fixed. Non-asymptotic convergence bounds to this limit in the L_2 sense are provided, together with upper bounds on the classification error that is shown to converge to the Bayes risk, therefore proving the Bayes-consistency of a variety of methods although the regularization term does not vanish. These results are particularly relevant to the one-class SVM, for which the regularization can not vanish by construction, and which is shown for the first time to be a consistent density level set estimator.

Keywords: regularization, Gaussian kernel RKHS, one-class SVM, convex loss functions, kernel density estimation

1. Introduction

Given n independent and identically distributed (i.i.d.) copies $(X_1,Y_1),\ldots,(X_n,Y_n)$ of a random variable $(X,Y)\in\mathbb{R}^d\times\{-1,1\}$, we study in this paper the limit and consistency of learning algorithms that solve the following problem:

$$\arg\min_{f\in\mathcal{H}_\sigma}\left\{\frac{1}{n}\sum_{i=1}^{n}\phi\left(Y_i f(X_i)\right)+\lambda\| f\|^2_{\mathcal{H}_\sigma}\right\}, \tag{1}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a convex loss function and $\mathcal{H}_\sigma$ is the reproducing kernel Hilbert space (RKHS) of the normalized Gaussian radial basis function kernel (denoted simply Gaussian kernel below):

$$k_\sigma(x,x') := \frac{1}{\left(\sqrt{2\pi}\sigma\right)^d} \exp\left(\frac{-\|x-x'\|^2}{2\sigma^2}\right) , \ \sigma > 0 . \tag{2}$$

This framework encompasses in particular the classical support vector machine (SVM) (Boser et al., 1992) when $\phi(u) = \max(1-u,0)$ (Theorem 6). Recent years have witnessed important theoretical advances aimed at understanding the behavior of such regularized algorithms when n tends to infinity and λ decreases to 0. In particular the consistency and convergence rates of the two-class SVM (see, e.g., Steinwart, 2002; Zhang, 2004; Steinwart and Scovel, 2004, and references therein) have been studied in detail, as well as the shape of the asymptotic decision function (Steinwart, 2003; Bartlett and Tewari, 2004). The case of more general convex loss functions has also attracted a lot of attention recently (Zhang, 2004; Lugosi and Vayatis, 2004; Bartlett et al., 2006), and been shown to provide under general assumptions consistent procedure for the classification error.

All results published so far, however, study the case where λ decreases as the number of points tends to infinity (or, equivalently, where $\lambda\sigma^{-d}$ converges to 0 if one uses the classical non-normalized version of the Gaussian kernel instead of (2)). Although it seems natural to reduce regularization as more and more training data are available — even more than natural, it is the spirit of regularization (Tikhonov and Arsenin, 1977; Silverman, 1982) —, there is at least one important situation where λ is typically held fixed: the one-class SVM (Schölkopf et al., 2001). In that case, the goal is to estimate an α-quantile, that is, a subset of $\mathbb{R}^d$ of given probability α with minimum volume. The estimation is performed by thresholding the function output by the one-class SVM, that is, the SVM (1) with only positive examples; in that case λ is supposed to determine the quantile level.[1] Although it is known that the fraction of examples in the selected region converges to the desired quantile level α (Schölkopf et al., 2001), it is still an open question whether the region converges towards a quantile, that is, a region of minimum volume. Besides, most theoretical results about the consistency and convergence rates of two-class SVM with vanishing regularization constant do not translate to the one-class case, as we are precisely in the seldom situation where the SVM is used with a regularization term that does not vanish as the sample size increases.

The main contribution of this paper is to show that Bayes consistency for the classification error can be obtained for algorithms that solve (1) without decreasing λ, if instead the bandwidth σ of the Gaussian kernel decreases at a suitable rate. We prove upper bounds on the convergence rate of the classification error towards the Bayes risk for a variety of functions ϕ and of distributions P, in particular for SVM (Theorem 6). Moreover, we provide an explicit description of the function asymptotically output by the algorithms, and establish converge rates towards this limit for the L_2 norm (Theorem 7). In particular, we show that the decision function output by the one-class SVM converges towards the density to be estimated, truncated at the level 2λ (Theorem 8); we finally show (Theorem 9) that this implies the consistency of one-class SVM as a density level set estimator for the excess-mass functional (Hartigan, 1987).

This paper is organized as follows. In Section 2, we set the framework of this study and state the main results. The rest of the paper is devoted to the proofs and some extensions of these results. In Section 3, we provide a number of known and new properties of the Gaussian RKHS. Section 4

1. While the original formulation of the one-class SVM involves a parameter ν, there is asymptotically a one-to-one correspondence between λ and ν.

is devoted to the proof of the main theorem that describes the speed of convergence of the regularized ϕ-risk of its empirical minimizer towards its minimum. This proof involves in particular a control of the sample error in this particular setting that is dealt with in Section 5. Section 6 relates the minimization of the regularized ϕ-risk to more classical measures of performance, in particular classification error and L_2 distance to the limit. These results are discussed in more detail in Section 7 for the case of the 1- and 2-SVM. Finally the proof of the consistency of the one-class SVM as a density level set estimator is postponed to Section 8.

2. Notation and Main Results

Let (X,Y) be a pair of random variables taking values in $\mathbb{R}^d \times \{-1,1\}$, with distribution P. We assume throughout this paper that the marginal distribution of X has a density $\rho : \mathbb{R}^d \to \mathbb{R}$ with respect to the Lebesgue measure, and that its support is included in a compact set $\mathcal{X} \subset \mathbb{R}^d$. Let $\eta : \mathbb{R}^d \to [0,1]$ denote a measurable version of the conditional distribution of $Y = 1$ given X. The function $2\eta - 1$ then corresponds to the so-called *regression function*.

The normalized Gaussian radial basis function (RBF) kernel k_σ with bandwidth parameter $\sigma > 0$ is defined for any $(x,x') \in \mathbb{R}^d \times \mathbb{R}^d$ by:[2]

$$k_\sigma(x,x') := \frac{1}{\left(\sqrt{2\pi}\sigma\right)^d} \exp\left(\frac{-\|x-x'\|^2}{2\sigma^2}\right) ,$$

the corresponding reproducing kernel Hilbert space (RKHS) is denoted by $\mathcal{H}_\sigma$, with associated norm $\|.\|_{\mathcal{H}_\sigma}$. Moreover let

$$\kappa_\sigma := \|k_\sigma\|_{L_\infty} = 1/\left(\sqrt{2\pi}\sigma\right)^d . \tag{3}$$

Several useful properties of this kernel and its RKHS are gathered in Section 3.

Denoting by $\mathcal{M}$ the set of measurable real-valued functions on $\mathbb{R}^d$, we define several risks for functions $f \in \mathcal{M}$:

- The classification error rate, usually ref

  $$R(f) := P(\operatorname{sign}(f(X)) \neq Y) ,$$

  and the minimum achievable classification error rate over $\mathcal{M}$ is called the Bayes risk:

  $$R^* := \inf_{f \in \mathcal{M}} R(f).$$

- For a scalar $\lambda > 0$ fixed throughout this paper and a convex function $\phi : \mathbb{R} \to \mathbb{R}$, the ϕ-risk regularized by the RKHS norm is defined, for any $\sigma > 0$ and $f \in \mathcal{H}_\sigma$, by

  $$R_{\phi,\sigma}(f) := \mathbb{E}_P[\phi(Yf(X))] + \lambda\|f\|^2_{\mathcal{H}_\sigma},$$

  and the minimum achievable $R_{\phi,\sigma}$-risk over $\mathcal{H}_\sigma$ is denoted by

  $$R^*_{\phi,\sigma} := \inf_{f \in \mathcal{H}_\sigma} R_{\phi,\sigma}(f).$$

2. We refer the reader to Section 3.2 for a brief discussion on the relation between normalized/unnormalized Gaussian kernel.

Furthermore, for any real $r \geq 0$, we know that ϕ is Lipschitz on $[-r,r]$, and we denote by $L(r)$ the Lipschitz constant of the restriction of ϕ to the interval $[-r,r]$. For example, for the hinge loss $\phi(u) = \max(0, 1-u)$ one can take $L(r) = 1$, and for the squared hinge loss $\phi(u) = \max(0,1-u)^2$ one can take $L(r) = 2(r+1)$.

- Finally, the L_2-norm regularized ϕ-risk is, for any $f \in \mathcal{M}$:

$$R_{\phi,0}(f) := \mathbb{E}_P[\phi(Yf(X))] + \lambda \| f \|_{L_2}^2$$

where,

$$\| f \|_{L_2}^2 := \int_{\mathbb{R}^d} f(x)^2 dx \in [0, +\infty],$$

and the minimum achievable $R_{\phi,0}$-risk over $\mathcal{M}$ is denoted by

$$R_{\phi,0}^* := \inf_{f \in \mathcal{M}} R_{\phi,0}(f) < \infty$$

As we shall see in the sequel, the above notation is consistent with the fact that $R_{\phi,0}$ is the pointwise limit of $R_{\phi,\sigma}$ as σ tends to zero.

Each of these risks has an empirical counterpart where the expectation with respect to P is replaced by an average over an i.i.d. sample $T := \{(X_1,Y_1),\ldots,(X_n,Y_n)\}$. In particular, the following empirical version of $R_{\phi,\sigma}$ will be used

$$\forall \sigma > 0, f \in \mathcal{H}_\sigma, \quad \widehat{R}_{\phi,\sigma}(f) := \frac{1}{n}\sum_{i=1}^n \phi(Y_i f(X_i)) + \lambda \| f \|_{\mathcal{H}_\sigma}^2$$

Furthermore, $\hat{f}_{\phi,\sigma}$ denotes the minimizer of $\widehat{R}_{\phi,\sigma}$ over $\mathcal{H}_\sigma$ (see Steinwart, 2005a, for a proof of existence and uniqueness of $\hat{f}_{\phi,\sigma}$).

The main focus of this paper is the analysis of learning algorithms that minimize the empirical ϕ-risk regularized by the RKHS norm $\widehat{R}_{\phi,\sigma}$, and their limit as the number of points tends to infinity and the kernel width σ decreases to 0 at a suitable rate when n tends to ∞, λ being kept fixed. Roughly speaking, our main result shows that in this situation, the minimization of $\widehat{R}_{\phi,\sigma}$ asymptotically amounts to minimizing $R_{\phi,0}$. This stems from the fact that the empirical average term in the definition of $\widehat{R}_{\phi,\sigma}$ converges to its corresponding expectation, while the norm in $\mathcal{H}_\sigma$ of a function f decreases to its L_2 norm when σ decreases to zero. To turn this intuition into a rigorous statement, we need a few more assumptions about the minimizer of $R_{\phi,0}$ and about P. First, we observe that the minimizer of $R_{\phi,0}$ is indeed well-defined and can often be explicitly computed (the following lemma is part of Theorem 26 and is proved in Section 6.3):

Lemma 1 (Minimizer of $R_{\phi,0}$) *For any $x \in \mathbb{R}^d$, let*

$$f_{\phi,0}(x) := \operatorname*{arg\,min}_{\alpha \in \mathbb{R}} \left\{ \eta(x)\phi(\alpha) + (1-\eta(x))\phi(-\alpha) \right\} .$$

Then $f_{\phi,0}$ is measurable and satisfies:

$$R_{\phi,0}\left(f_{\phi,0}\right) = \inf_{f \in \mathcal{M}} R_{\phi,0}(f)$$

Second, let us recall the notion of modulus of continuity (DeVore and Lorentz, 1993, p.44):

Definition 2 (Modulus of Continuity) *Let f be a Lebesgue measurable function from $\mathbb{R}^d$ to $\mathbb{R}$. Then its modulus of continuity in the L_1-norm is defined for any $\delta \geq 0$ as follows*

$$\omega(f,\delta) := \sup_{0\leq\|t\|\leq\delta} \| f(.+t) - f(.) \|_{L_1} , \tag{4}$$

where $\|t\|$ is the Euclidean norm of $t \in \mathbb{R}^d$.

The main result of this paper, whose proof is postponed to Section 4, can now be stated as follows:

Theorem 3 (Main Result) *Let $\sigma_1 > \sigma > 0$, $0 < p < 2$, $\delta > 0$, and let $\hat{f}_{\phi,\sigma}$ denote the minimizer of the $\widehat{R}_{\phi,\sigma}$ risk over $\mathcal{H}_\sigma$, where ϕ is assumed to be convex. Assume that the marginal density ρ is bounded, and let $M := \sup_{x\in\mathbb{R}^d} \rho(x)$. Then there exist constants $(K_i)_{i=1\ldots4}$ (depending only on p, δ, λ, d, and M) such that the following holds with probability greater than $1-e^{-x}$ over the draw of the training data*

$$\begin{aligned} R_{\phi,0}(\hat{f}_{\phi,\sigma}) - R^*_{\phi,0} \leq\ & K_1 L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right)^{\frac{4}{2+p}} \left(\frac{1}{\sigma}\right)^{\frac{[2+(2-p)(1+\delta)]d}{2+p}} \left(\frac{1}{n}\right)^{\frac{2}{2+p}} \\ & + K_2 L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right)^{2} \left(\frac{1}{\sigma}\right)^{d} \frac{x}{n} \\ & + K_3 \frac{\sigma^2}{\sigma_1^2} \\ & + K_4 \omega(f_{\phi,0}, \sigma_1) , \end{aligned} \tag{5}$$

where $L(r)$ still denotes the Lipschitz constant of ϕ on the interval $[-r,r]$, for any $r > 0$.

The first two terms in r.h.s. of (5) bound the estimation error (also called sample error) associated with the Gaussian RKHS, which naturally tends to be small when the number of training data increases and when the RKHS is 'small', i.e., when σ is large. As is usually the case in such variance/bias splittings, the variance term here depends on the dimension d of the input space. Note that it is also parametrized by both p and δ. These two parameters come from the bound (36) on covering numbers that is used to derive the estimation error bound (31). Both constants K_1 and K_2 depend on them, although we do not know the explicit dependency. The third term measures the error due to penalizing the L_2-norm of a fixed function in $\mathcal{H}_{\sigma_1}$ by its $\|.\|_{\mathcal{H}_\sigma}$-norm, with $0 < \sigma < \sigma_1$. This is a price to pay to get a small estimation error. As for the fourth term, it is a bound on the approximation error of the Gaussian RKHS. Note that, once λ and σ have been fixed, σ_1 remains a free variable parameterizing the bound itself.

From (5), we can deduce the $R_{\phi,0}$-consistency of $\hat{f}_{\phi,\sigma}$ for Lipschitz loss functions, as soon as $f_{\phi,0}$ is integrable, $\sigma = o\left(n^{-1/(d+\varepsilon)}\right)$ for some $\varepsilon > 0$, and $\sigma_1 \to 0$ with $\sigma/\sigma_1 \to 0$. Now, in order to highlight the type of convergence rates one can obtain from Theorem 3, let us assume that the ϕ loss function is Lipschitz on $\mathbb{R}$ (e.g., take the hinge loss), and suppose that for some $0 \leq \beta \leq 1$, $c_1 > 0$,

and for any $h \geq 0$, $f_{\phi,0}$ satisfies the following inequality:[3]

$$\omega(f_{\phi,0}, \delta) \leq c_1 \delta^{\beta} . \tag{6}$$

Then the right-hand side of (5) can be optimized w.r.t. σ_1, σ, p and δ by balancing the first, third and fourth terms (the second term having always a better convergence rate than the first one). For any $\varepsilon > 0$, by choosing:

$$\begin{aligned}
\delta &= 1 , \\
p &= 2 - \frac{\varepsilon}{2d + d\beta - \beta} , \\
\sigma &= \left(\frac{1}{n}\right)^{\frac{2+\beta}{4\beta+(2+\beta)d+\varepsilon}} , \\
\sigma_1 &= \sigma^{\frac{2}{2+\beta}} = \left(\frac{1}{n}\right)^{\frac{2}{4\beta+(2+\beta)d+\varepsilon}} ,
\end{aligned}$$

the following rate of convergence is obtained:

$$R_{\phi,0}\left(\hat{f}_{\phi,\sigma}\right) - R^*_{\phi,0} = O_P\left(\left(\frac{1}{n}\right)^{\frac{2\beta}{4\beta+(2+\beta)d+\varepsilon}}\right) .$$

This shows in particular that, whatever the values of β and d, the convergence rate that can be derived from Theorem 3 is always slower than $1/\sqrt{n}$, and it gets slower and slower as the dimension d increases.

Theorem 3 shows that, when ϕ is convex, minimizing the $\widehat{R}_{\phi,\sigma}$ risk for well-chosen width σ is a an algorithm consistent for the $R_{\phi,0}$-risk. In order to relate this consistency with more traditional measures of performance of learning algorithms, the next theorem shows that under a simple additional condition on ϕ, $R_{\phi,0}$-risk-consistency implies Bayes consistency:

Theorem 4 (Relating $R_{\phi,0}$-Consistency with Bayes Consistency) *If ϕ is convex, differentiable at 0, with $\phi'(0) < 0$, then for every sequence of functions $(f_i)_{i\geq 1} \in \mathcal{M}$,*

$$\lim_{i\to+\infty} R_{\phi,0}(f_i) = R^*_{\phi,0} \implies \lim_{i\to+\infty} R(f_i) = R^*$$

This theorem results from a more general quantitative analysis of the relationship between the excess $R_{\phi,0}$-risk and the excess R-risk (Theorem 28), and is proved in Section 6.5. In order to state a refined version of it in the particular case of the support vector machine algorithm, we first need to introduce the notion of *low density exponent*:

Definition 5 *We say that a distribution P with marginal density ρ w.r.t. Lebesgue measure has a* low density exponent $\gamma \geq 0$ *if there exists $(c_2, \varepsilon_0) \in (0,\infty)^2$ such that*

$$\forall \varepsilon \in [0, \varepsilon_0], \quad P\left(\left\{x \in \mathbb{R}^d : \rho(x) \leq \varepsilon\right\}\right) \leq c_2 \varepsilon^{\gamma}.$$

3. For instance, it can be shown that the indicator function of the unit ball in $\mathbb{R}^d$, albeit not continuous, satisfies (6) with $\beta = 1$.

We are now in position to state a quantitative relationship between the excess $R_{\phi,0}$-risk and the excess R-risk in the case of support vector machines :

Theorem 6 (Consistency of SVM) *Let $\phi_1(\alpha) := \max(1-\alpha, 0)$ be the hinge loss function and let $\phi_2(\alpha) := \max(1-\alpha, 0)^2$ be the squared hinge loss function. Then for any distribution P with low density exponent γ, there exist constant $(K_1, K_2, r_1, r_2) \in (0,\infty)^4$ such that for any $f \in \mathcal{M}$ with an excess $R_{\phi_1,0}$-risk upper bounded by r_1 the following holds:*

$$R(f) - R^* \leq K_1 \left(R_{\phi_1,0}(f) - R^*_{\phi_1,0}\right)^{\frac{\gamma}{2\gamma+1}},$$

and if the excess regularized $R_{\phi_2,0}$-risk upper bounded by r_2 the following holds:

$$R(f) - R^* \leq K_2 \left(R_{\phi_2,2}(f) - R^*_{\phi_2,2}\right)^{\frac{\gamma}{2\gamma+1}},$$

This theorem is proved in Section 7.3. In combination with Theorem 3, it states the consistency of SVM, and gives upper bounds on the convergence rates, for the first time in a situation where the effect of regularization does not vanish asymptotically. In fact, Theorem 6 is a particular case of a more general result (Theorem 29) valid for a large class of convex loss functions. Section 6 is devoted to the analysis of the general case through the introduction of variational arguments, in the spirit of Bartlett et al. (2006).

Another consequence of the $R_{\phi,0}$-consistency of an algorithm is the L_2 convergence of the function output by the algorithm to the minimizer of the $R_{\phi,0}$-risk :

Lemma 7 (Relating $R_{\phi,0}$-Consistency with L_2-Consistency) *For any $f \in \mathcal{M}$, the following holds:*

$$\| f - f_{\phi,0} \|^2_{L_2} \leq \frac{1}{\lambda} \left(R_{\phi,0}(f) - R^*_{\phi,0}\right).$$

This result is the third statement of Theorem 26, proved in Section 6.3. It is particularly relevant to study algorithms whose objective is not binary classification. Consider for example the one-class SVM algorithm, which served as the initial motivation for this paper. Then we can state the following result, proved in Section 8.1 :

Theorem 8 (L_2-Consistency of One-Class SVM) *Let ρ_λ denote the function obtained after truncating the density:*

$$\rho_\lambda(x) := \begin{cases} \frac{\rho(x)}{2\lambda} & \text{if } \rho(x) \leq 2\lambda, \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

Let $\hat{f}_\sigma$ denote the function output by the one-class SVM:

$$\hat{f}_\sigma := \arg\min_{f \in \mathcal{H}_\sigma} \frac{1}{n} \sum_{i=1}^n \phi(f(X_i)) + \lambda \| f \|^2_{\mathcal{H}_\sigma}.$$

Then, under the general conditions of Theorem 3, and assuming that $\lim_{h\to 0} \omega(\rho_\lambda, h) = 0$,

$$\lim_{n\to+\infty} \| \hat{f}_\sigma - \rho_\lambda \|_{L_2} = 0, \quad \text{in probability,}$$

for a well-calibrated bandwidth σ.

In this and the next theorem, *well-calibrated* refers to any choice of bandwidth σ that ensures $R_{\phi,0}$-consistency, as discussed after Theorem 3. A very interesting by-product of this theorem is the consistency of the one-class SVM algorithm for density level set estimation, which to the best of our knowledge has not been stated so far (the proof being postponed to Section 8.2) :

Theorem 9 (Consistency of One-Class SVM for Density Level Set Estimation) *Let $0 < \mu < 2\lambda < M$, let C_μ be the level set of the density function ρ at level μ:*

$$C_\mu := \left\{ x \in \mathbb{R}^d \ : \ \rho(x) \geq \mu \right\} ,$$

and $\widehat{C}_\mu$ be the level set of $2\lambda \hat{f}_\sigma$ at level μ:

$$\widehat{C}_\mu := \left\{ x \in \mathbb{R}^d \ : \ \hat{f}_\sigma(x) \geq \frac{\mu}{2\lambda} \right\} ,$$

where $\hat{f}_\sigma$ is still the function output by the one-class SVM. For any distribution Q, for any subset C of $\mathbb{R}^d$, define the excess-mass of C with respect to Q as follows:

$$H_Q(C) := Q(C) - \mu Leb(C) \ .$$

Then, under the general assumptions of Theorem 3, and assuming that $\lim_{h \to 0} \omega(\rho_\lambda, h) = 0$, we have

$$\lim_{n \to +\infty} H_P(C_\mu) - H_P\left(\widehat{C}_\mu\right) = 0 \ , \quad \textit{in probability,}$$

for a well-calibrated bandwidth σ.

The excess-mass functional was first introduced by Hartigan (1987) to assess the quality of density level set estimators. It is maximized by the true density level set C_μ and acts as a risk functional in the one-class framework. The proof of Theorem 9 is based on the following general result: if $\hat{\rho}$ is a density estimator converging to the true density ρ in the L_2 sense, then for any fixed $0 < \mu < \sup_{\mathbb{R}^d} \{\rho\}$, the excess mass of the level set of $\hat{\rho}$ at level μ converges to the excess mass of C_μ. In other words, as is the case in the classification framework, plug-in rules built on L_2-consistent density estimators are consistent with respect to the excess mass.

3. Some Properties of the Gaussian Kernel and its RKHS

This section presents known and new results about the Gaussian kernel k_σ and its associated RKHS $\mathcal{H}_\sigma$, that are useful for the proofs of our results. They concern the explicit description of the RKHS norm in terms of Fourier transforms, its relation with the L_2-norm, and some approximation properties of convolutions with the Gaussian kernel. They make use of basic properties of Fourier transforms which we now recall (and which can be found in e.g. Folland, 1992, Chap. 7, p.205).
For any f in $L_1(\mathbb{R}^d)$, its Fourier transform $\mathcal{F}[f] : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\mathcal{F}[f](\omega) := \int_{\mathbb{R}^d} e^{-i<x,\omega>} f(x) dx \, .$$

If in addition $\mathcal{F}[f] \in L_1(\mathbb{R}^d)$, f can be recovered from $\mathcal{F}[f]$ by the inverse Fourier formula:

$$f(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \mathcal{F}[f](\omega) e^{i<x,\omega>} d\omega \, .$$

Finally Parseval's equality relates the L_2-norm of a function and its Fourier transform if $f \in L_1(\mathbb{R}^d) \cap L_2(\mathbb{R}^d)$ and $\mathcal{F}[f] \in L_2(\mathbb{R}^d)$:

$$\| f \|_{L_2}^2 = \frac{1}{(2\pi)^d} \| \mathcal{F}[f] \|_{L_2}^2 . \tag{8}$$

3.1 Fourier Representation of the Gaussian RKHS

For any $u \in \mathbb{R}^d$, the expression $k_\sigma(u)$ denotes $k_\sigma(0,u)$, with Fourier transform known to be:

$$\mathcal{F}[k_\sigma](\omega) = e^{\frac{-\sigma^2\|\omega\|^2}{2}} . \tag{9}$$

The general study of translation invariant kernels provides an accurate characterization of their associated RKHS in terms of the their Fourier transform (see, e.g., Matache and Matache, 2002). In the case of the Gaussian kernel, the following holds :

Lemma 10 (Characterization of $\mathcal{H}_\sigma$) *Let $C_0(\mathbb{R}^d)$ denote the set of continuous functions on $\mathbb{R}^d$ that vanish at infinity. The set*

$$\mathcal{H}_\sigma := \left\{ f \in C_0(\mathbb{R}^d) \ : \ f \in L_1(\mathbb{R}^d) \text{ and } \int_{\mathbb{R}^d} |\mathcal{F}[f](\omega)|^2 e^{\frac{\sigma^2\|\omega\|^2}{2}} d\omega < \infty \right\} \tag{10}$$

is the RKHS associated with the Gaussian kernel k_σ, and the associated dot product is given for any $f, g \in \mathcal{H}_\sigma$ by

$$\langle f, g \rangle_{\mathcal{H}_\sigma} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \mathcal{F}[f](\omega) \mathcal{F}[g](\omega)^* e^{\frac{\sigma^2\|\omega\|^2}{2}} d\omega , \tag{11}$$

where a^* denotes the conjugate of a complex number a. In particular the associated norm is given for any $f \in \mathcal{H}_\sigma$ by

$$\| f \|_{\mathcal{H}_\sigma}^2 = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}[f](\omega)|^2 e^{\frac{\sigma^2\|\omega\|^2}{2}} d\omega . \tag{12}$$

This lemma readily implies several basic facts about Gaussian RKHS and their associated norms summarized in the next lemma. In particular, it shows that the family $(\mathcal{H}_\sigma)_{\sigma>0}$ forms a nested collection of models, and that for any fixed function, the RKHS norm decreases to the L_2-norm as the kernel bandwidth decreases to 0.

Lemma 11 *The following statements hold:*

1. *For any $0 < \sigma < \tau$,*

$$\mathcal{H}_\tau \subset \mathcal{H}_\sigma \subset L_2(\mathbb{R}^d) . \tag{13}$$

Moreover, for any $f \in \mathcal{H}_\tau$,

$$\| f \|_{\mathcal{H}_\tau} \geq \| f \|_{\mathcal{H}_\sigma} \geq \| f \|_{L_2} \tag{14}$$

and

$$0 \leq \| f \|_{\mathcal{H}_\sigma}^2 - \| f \|_{L_2}^2 \leq \frac{\sigma^2}{\tau^2} \left(\| f \|_{\mathcal{H}_\tau}^2 - \| f \|_{L_2}^2 \right) . \tag{15}$$

2. *For any $\tau > 0$ and $f \in \mathcal{H}_\tau$,*

$$\lim_{\sigma \to 0} \| f \|_{\mathcal{H}_\sigma} = \| f \|_{L_2} . \tag{16}$$

3. For any $\sigma > 0$ and $f \in \mathcal{H}_\sigma$,

$$\| f \|_{L_\infty} \leq \sqrt{\kappa_\sigma} \| f \|_{\mathcal{H}_\sigma} . \tag{17}$$

Proof Equations 13 and 14 are direct consequences of the characterization of the Gaussian RKHS (12) and of the observation that

$$0 < \sigma < \tau \implies e^{\frac{\tau^2 \|\omega\|^2}{2}} \geq e^{\frac{\sigma^2 \|\omega\|^2}{2}} \geq 1.$$

In order to prove (15), we derive from (12) and Parseval's equality (8):

$$\| f \|^2_{\mathcal{H}_\sigma} - \| f \|^2_{L_2} = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}[f](\omega)|^2 \left[e^{\frac{\sigma^2 \|\omega\|^2}{2}} - 1 \right] d\omega . \tag{18}$$

For any $0 \leq u \leq v$, we have $(e^u - 1)/u \leq (e^v - 1)/v$ by convexity of e^u, and therefore:

$$\| f \|^2_{\mathcal{H}_\sigma} - \| f \|^2_{L_2} \leq \frac{1}{(2\pi)^d} \frac{\sigma^2}{\tau^2} \int_{\mathbb{R}^d} |\mathcal{F}[f](\omega)|^2 \left[e^{\frac{\tau^2 \|\omega\|^2}{2}} - 1 \right] d\omega, \tag{19}$$

which leads to (15). Equation 16 is now a direct consequence of (15). Finally, (17) is a classical bound derived from the observation that, for any $x \in \mathbb{R}^d$,

$$| f(x) | = \left| \langle f, k_\sigma \rangle_{\mathcal{H}_\sigma} \right| \leq \| f \|_{\mathcal{H}_\sigma} \| k_\sigma \|_{\mathcal{H}_\sigma} = \sqrt{\kappa_\sigma} \| f \|_{\mathcal{H}_\sigma} .$$

■

3.2 Links with the Non-Normalized Gaussian Kernel

It is common in the machine learning literature to work with a non-normalized version of the Gaussian RBF kernel, namely the kernel:

$$\tilde{k}_\sigma(x, x') := \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2} \right) . \tag{20}$$

From the relation $k_\sigma = \kappa_\sigma \tilde{k}_\sigma$ (remember that κ_σ is defined in Equation 3), we deduce from the general theory of RKHS that $\mathcal{H}_\sigma = \tilde{\mathcal{H}}_\sigma$ and

$$\forall f \in \mathcal{H}_\sigma, \quad \| f \|_{\tilde{\mathcal{H}}_\sigma} = \sqrt{\kappa_\sigma} \| f \|_{\mathcal{H}_\sigma} . \tag{21}$$

As a result, all statements about k_σ and its RKHS easily translate into statements about $\tilde{k}_\sigma$ and its RKHS. For example, (14) shows that, for any $0 < \sigma < \tau$ and $f \in \tilde{\mathcal{H}}_\tau$,

$$\| f \|_{\tilde{\mathcal{H}}_\tau} \geq \sqrt{\frac{\kappa_\tau}{\kappa_\sigma}} \| f \|_{\tilde{\mathcal{H}}_\sigma} = \left(\frac{\sigma}{\tau} \right)^{\frac{d}{2}} \| f \|_{\tilde{\mathcal{H}}_\sigma} ,$$

a result that was shown recently (Steinwart et al., 2004, Corollary 3.12).

3.3 Convolution with the Gaussian Kernel

Besides its positive definiteness, the Gaussian kernel is commonly used as a kernel for function approximation through convolution. Recall (Folland, 1992) that the convolution between two functions $f, g \in L_1\left(\mathbb{R}^d\right)$ is the function $f * g \in L_1\left(\mathbb{R}^d\right)$ defined by

$$f * g(x) := \int_{\mathbb{R}^d} f(x-u)\, g(u)\, du$$

and that it satisfies (see e.g. Folland, 1992, Chap. 7, p.207)

$$\mathcal{F}\left[f * g\right] = \mathcal{F}\left[f\right] \mathcal{F}\left[g\right] . \tag{22}$$

The convolution with a Gaussian RBF kernel is a technically convenient tool to map any square integrable function to a Gaussian RKHS. The following lemma (which can also be found in Steinwart et al., 2004) gives several interesting properties on the RKHS and L_2 norms of functions smoothed by convolution:

Lemma 12 *For any $\sigma > 0$ and any $f \in L_1\left(\mathbb{R}^d\right) \cap L_2\left(\mathbb{R}^d\right)$,*

$$k_\sigma * f \in \mathcal{H}_{\sqrt{2}\sigma}$$

and

$$\| k_\sigma * f \|_{\mathcal{H}_{\sqrt{2}\sigma}} = \| f \|_{L_2} . \tag{23}$$

For any $\sigma, \tau > 0$ that satisfy $0 < \sigma \leq \sqrt{2}\tau$, and for any $f \in L_1\left(\mathbb{R}^d\right) \cap L_2\left(\mathbb{R}^d\right)$,

$$k_\tau * f \in \mathcal{H}_\sigma$$

and

$$\| k_\tau * f \|^2_{\mathcal{H}_\sigma} - \| k_\tau * f \|^2_{L_2} \leq \frac{\sigma^2}{2\tau^2} \| f \|^2_{L_2} . \tag{24}$$

Proof Using (12), then (22) and (9), followed by Parseval's equality (8), we compute:

$$\begin{aligned}
\| k_\sigma * f \|^2_{\mathcal{H}_{\sqrt{2}\sigma}} &= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}\left[k_\sigma * f\right](\omega)|^2 e^{\sigma^2 \|\omega\|^2} d\omega \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}\left[f\right](\omega)|^2 e^{-\sigma^2 \|\omega\|^2} e^{\sigma^2 \|\omega\|^2} d\omega \\
&= \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\mathcal{F}\left[f\right](\omega)|^2 d\omega \\
&= \| f \|^2_{L_2} .
\end{aligned}$$

This proves the first two statements of the lemma.

Now, because $0 < \sigma \leq \sqrt{2}\tau$, previous first statement and (13) imply

$$k_\tau * f \in \mathcal{H}_{\sqrt{2}\tau} \subset \mathcal{H}_\sigma ,$$

and, using (15) and (23),

$$\begin{aligned}\| k_\tau * f \|^2_{\mathcal{H}_\sigma} - \| k_\tau * f \|^2_{L_2} &\leq \frac{\sigma^2}{2\tau^2}\left(\| k_\tau * f \|^2_{\mathcal{H}_{\sqrt{2}\tau}} - \| k_\tau * f \|^2_{L_2}\right) \\ &\leq \frac{\sigma^2}{2\tau^2}\| k_\tau * f \|^2_{\mathcal{H}_{\sqrt{2}\tau}} \\ &= \frac{\sigma^2}{2\tau^2}\| f \|^2_{L_2} .\end{aligned}$$

■

A final result we need is an estimate of the approximation properties of convolution with the Gaussian kernel. Convolving a function with a Gaussian kernel with decreasing bandwidth is known to provide an approximation of the original function under general conditions. For example, the assumption $f \in L_1(\mathbb{R}^d)$ is sufficient to show that $\| k_\sigma * f - f \|_{L_1}$ goes to zero when σ goes to zero (see, for example Devroye and Lugosi, 2000, page 79). We provide below a more quantitative estimate for the rate of this convergence under some assumption on the modulus of continuity of f (see Definition 2), using methods from DeVore and Lorentz (1993, Chap.7, par.2, p.202).

Lemma 13 *Let f be a bounded function in $L_1(\mathbb{R}^d)$. Then for all $\sigma > 0$, the following holds:*

$$\| k_\sigma * f - f \|_{L_1} \leq (1+\sqrt{d})\omega(f,\sigma) ,$$

where $\omega(f,.)$ denotes the modulus of continuity of f in the L_1 norm.

Proof Using the fact that k_σ is normalized, then Fubini's theorem and then the definition of ω, the following can be derived

$$\begin{aligned}\| k_\sigma * f - f \|_{L_1} &= \int_{\mathbb{R}^d}\left|\int_{\mathbb{R}^d} k_\sigma(t)[f(x+t)-f(x)]dt\right|dx \\ &\leq \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} k_\sigma(t)\,|f(x+t)-f(x)|\,dtdx \\ &= \int_{\mathbb{R}^d} k_\sigma(t)\left[\int_{\mathbb{R}^d}|f(x+t)-f(x)|\,dx\right]dt \\ &\leq \int_{\mathbb{R}^d} k_\sigma(t)\| f(.+t)-f(.)\|_{L_1}dt \\ &\leq \int_{\mathbb{R}^d} k_\sigma(t)\omega(f,\| t\|)dt .\end{aligned}$$

Now, using the following subadditivity property of ω (DeVore and Lorentz, 1993, Chap.2, par.7, p.44):

$$\omega(f,\delta_1+\delta_2) \leq \omega(f,\delta_1)+\omega(f,\delta_2) \ , \ \delta_1,\delta_2 > 0 ,$$

the following inequality can be derived for any positive λ and δ:

$$\omega(f,\lambda\delta) \leq (1+\lambda)\omega(f,\delta) .$$

Applying this and also Cauchy-Schwarz inequality leads to

$$\begin{aligned}
\|k_\sigma * f - f\|_{L_1} &\leq \int_{\mathbb{R}^d} \left(1 + \frac{\|t\|}{\sigma}\right) \omega(f,\sigma) k_\sigma(t)\,dt \\
&= \omega(f,\sigma)\left[1 + \frac{1}{\sigma}\int_{\mathbb{R}^d} \|t\| k_\sigma(t)\,dt\right] \\
&\leq \omega(f,\sigma)\left[1 + \frac{1}{\sigma}\left(\int_{\mathbb{R}^d} \|t\|^2 \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-\frac{\|t\|^2}{2\sigma^2}}\,dt\right)^{\frac{1}{2}}\right] \\
&= \omega(f,\sigma)\left[1 + \frac{1}{\sigma}\left(\sum_{i=1}^d \int_{\mathbb{R}^d} t_i^2 \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-\frac{\|t\|^2}{2\sigma^2}}\,dt\right)^{\frac{1}{2}}\right] \\
&= \omega(f,\sigma)\left[1 + \frac{1}{\sigma}\left(\sum_{i=1}^d \int_{\mathbb{R}^d} t_i^2 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t_i^2}{2\sigma^2}}\,dt_i\right)^{\frac{1}{2}}\right] \\
&= \omega(f,\sigma)\left[1 + \frac{1}{\sigma}\sqrt{d}\left(\int_{\mathbb{R}^d} u^2 \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{u^2}{2\sigma^2}}\,du\right)^{\frac{1}{2}}\right].
\end{aligned}$$

The integral term is exactly the variance of a Gaussian random variable, namely σ^2. Hence we end up with

$$\|k_\sigma * f - f\|_{L_1} \leq (1 + \sqrt{d})\omega(f,\sigma)\,.$$

■

4. Proof of Theorem 3

The proof of Theorem 3 is based on the following decomposition of the excess $R_{\phi,0}$-risk for the minimizer of the $\widehat{R}_{\phi,\sigma}$-risk:

Lemma 14 *For any $0 < \sigma < \sqrt{2}\sigma_1$ and any sample $(X_i, Y_i)_{i=1,\ldots,n}$, the minimizer $\hat{f}_{\phi,\sigma}$ of $\widehat{R}_{\phi,\sigma}$ satisfies:*

$$\begin{aligned}
R_{\phi,0}(\hat{f}_{\phi,\sigma}) - R^*_{\phi,0} \leq{}& \left[R_{\phi,\sigma}(\hat{f}_{\phi,\sigma}) - R^*_{\phi,\sigma}\right] \\
&+ \left[R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0}) - R_{\phi,0}(k_{\sigma_1} * f_{\phi,0})\right] \\
&+ \left[R_{\phi,0}(k_{\sigma_1} * f_{\phi,0}) - R^*_{\phi,0}\right]
\end{aligned} \tag{25}$$

Proof The excess $R_{\phi,0}$-risk decomposes as follows:

$$\begin{aligned}
R_{\phi,0}(\hat{f}_{\phi,\sigma}) - R^*_{\phi,0} &= \left[R_{\phi,0}\left(\hat{f}_{\phi,\sigma}\right) - R_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right)\right] \\
&\quad + \left[R_{\phi,\sigma}(\hat{f}_{\phi,\sigma}) - R^*_{\phi,\sigma}\right] \\
&\quad + \left[R^*_{\phi,\sigma} - R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0})\right] \\
&\quad + \left[R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0}) - R_{\phi,0}(k_{\sigma_1} * f_{\phi,0})\right] \\
&\quad + \left[R_{\phi,0}(k_{\sigma_1} * f_{\phi,0}) - R^*_{\phi,0}\right].
\end{aligned}$$

Note that by Lemma 12, $k_{\sigma_1} * f_{\phi,0} \in \mathcal{H}_{\sqrt{2}\sigma_1} \subset \mathcal{H}_\sigma \subset L_2\left(\mathbb{R}^d\right)$ which justifies the introduction of $R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0})$ and $R_{\phi,0}(k_{\sigma_1} * f_{\phi,0})$. Now, by definition of the different risks using (14), we have

$$R_{\phi,0}\left(\hat{f}_{\phi,\sigma}\right) - R_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) = \lambda\left(\| \hat{f}_{\phi,\sigma} \|_{L_2}^2 - \| \hat{f}_{\phi,\sigma} \|_{\mathcal{H}_\sigma}^2\right) \leq 0,$$

and

$$R^*_{\phi,\sigma} - R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0}) \leq 0.$$

■

Hence, controlling $R_{\phi,0}(\hat{f}_{\phi,\sigma}) - R^*_{\phi,0}$ to prove Theorem 3 boils down to controlling each of the three terms arising in (25), which can be done as follows:

- The first term in (25) is usually referred to as the sample error or estimation error. The control of such quantities has been the topic of much research recently, including for example Tsybakov (1997); Mammen and Tsybakov (1999); Massart (2000); Bartlett et al. (2005); Koltchinskii (2003); Steinwart and Scovel (2004). Using estimates of local Rademacher complexities through covering numbers for the Gaussian RKHS due to Steinwart and Scovel (2004), we prove below the following result

  Lemma 15 *For any $\sigma > 0$ small enough, let $\hat{f}_{\phi,\sigma}$ be the minimizer of the $\widehat{R}_{\phi,\sigma}$-risk on a sample of size n, where ϕ is a convex loss function. For any $0 < p < 2$, $\delta > 0$, and $x \geq 1$, the following holds with probability at least $1 - e^x$ over the draw of the sample:*

$$R_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) - R^*_{\phi,\sigma} \leq K_1 L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right)^{\frac{4}{2+p}} \left(\frac{1}{\sigma}\right)^{\frac{[2+(2-p)(1+\delta)]d}{2+p}} \left(\frac{1}{n}\right)^{\frac{2}{2+p}} + K_2 L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right)^2 \left(\frac{1}{\sigma}\right)^d \frac{x}{n},$$

  where K_1 and K_2 are positive constants depending neither on σ, nor on n.

- The second term in (25) can be upper bounded by

$$\frac{\phi(0)\sigma^2}{2\sigma_1^2}.$$

  Indeed, using Lemma 12, and the fact that $\sigma < \sqrt{2}\sigma_1$, we have

$$\begin{aligned} R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0}) - R_{\phi,0}(k_{\sigma_1} * f_{\phi,0}) &= \lambda\left[\| k_{\sigma_1} * f_{\phi,0} \|_{\mathcal{H}_\sigma}^2 - \| k_{\sigma_1} * f_{\phi,0} \|_{L_2}^2\right] \\ &\leq \frac{\lambda\sigma^2}{2\sigma_1^2} \| f_{\phi,0} \|_{L_2}^2. \end{aligned}$$

  Since $f_{\phi,0}$ minimizes $R_{\phi,0}$, we have $R_{\phi,0}\left(f_{\phi,0}\right) \leq R_{\phi,0}(0)$, which leads to $\| f_{\phi,0} \|_{L_2}^2 \leq \phi(0)/\lambda$. Therefore, we have

$$R_{\phi,\sigma}(k_{\sigma_1} * f_{\phi,0}) - R_{\phi,0}(k_{\sigma_1} * f_{\phi,0}) \leq \frac{\phi(0)\sigma^2}{2\sigma_1^2}.$$

- The third term in (25) can be upper bounded by

$$(2\lambda\| f_{\phi,0}\|_{L_\infty} + L\left(\| f_{\phi,0}\|_{L_\infty}\right) M)\left(1+\sqrt{d}\right)\omega\left(f_{\phi,0},\sigma_1\right) .$$

Indeed,

$$\begin{aligned} & R_{\phi,0}(k_{\sigma_1} * f_{\phi,0}) - R_{\phi,0}(f_{\phi,0}) \\ = \;& \lambda\left[\| k_{\sigma_1} * f_{\phi,0}\|_{L_2}^2 - \| f_{\phi,0}\|_{L_2}^2\right] + \left[\mathbb{E}_P\left[\phi\left(Y(k_{\sigma_1} * f_{\phi,0})(X)\right)\right] - \mathbb{E}_P\left[\phi\left(Y f_{\phi,0}(X)\right)\right]\right] \\ = \;& \lambda\left\langle k_{\sigma_1} * f_{\phi,0} - f_{\phi,0}, k_{\sigma_1} * f_{\phi,0} + f_{\phi,0}\right\rangle_{L_2} + \mathbb{E}_P\left[\phi\left(Y(k_{\sigma_1} * f_{\phi,0})(X)\right) - \phi\left(Y f_{\phi,0}(X)\right)\right] . \end{aligned}$$

Now, since $\| k_{\sigma_1} * f_{\phi,0}\|_{L_\infty} \leq \| f_{\phi,0}\|_{L_\infty}\| k_{\sigma_1}\|_{L_1} = \| f_{\phi,0}\|_{L_\infty}$, then using Lemma 13, we obtain:

$$\begin{aligned} R_{\phi,0}(k_{\sigma_1} * f_{\phi,0}) - R_{\phi,0}(f_{\phi,0}) \;\leq\; & 2\lambda\| f_{\phi,0}\|_{L_\infty}\| k_{\sigma_1} * f_{\phi,0} - f_{\phi,0}\|_{L_1} \\ & + L\left(\| f_{\phi,0}\|_{L_\infty}\right)\mathbb{E}_P\left[|(k_{\sigma_1} * f_{\phi,0})(X) - f_{\phi,0}(X)|\right] \\ \leq \;& (2\lambda\| f_{\phi,0}\|_{L_\infty} + L\left(\| f_{\phi,0}\|_{L_\infty}\right) M)\| k_{\sigma_1} * f_{\phi,0} - f_{\phi,0}\|_{L_1} \\ \leq \;& (2\lambda\| f_{\phi,0}\|_{L_\infty} + L\left(\| f_{\phi,0}\|_{L_\infty}\right) M)\left(1+\sqrt{d}\right)\omega\left(f_{\phi,0},\sigma_1\right) , \end{aligned}$$

where $M := \sup_{x\in\mathbb{R}^d}\rho(x)$ is supposed to be finite.

Now, Theorem 3 is proved by plugging the last three bounds in (25). ■

5. Proof of Lemma 15 (Sample Error)

The present section is divided into two subsections: the first one presents the proof of Lemma 15, and the auxiliary lemmas that are used in it are then proved in the second subsection.

5.1 Proof of Lemma 15

In order to upper bound the sample error, it is useful to work with a set of functions as "small" as possible, in a meaning made rigorous below. Although we study algorithms that work on the whole RKHS $\mathcal{H}_\sigma$ *a priori*, let us first show that we can drastically "downsize" it.

Indeed, recall that the marginal distribution of P in X is assumed to have a support included in a compact $\mathcal{X} \subset \mathbb{R}^d$. The restriction of k_σ to $\mathcal{X}$, denoted by $k_\sigma^{\mathcal{X}}$, is a positive definite kernel on $\mathcal{X}$ (Aronszajn, 1950) with RKHS defined by:

$$\mathcal{H}_\sigma^{\mathcal{X}} := \left\{ f_{|\mathcal{X}} : f \in \mathcal{H}_\sigma \right\} , \tag{26}$$

where $f_{|\mathcal{X}}$ denotes the restriction of f to $\mathcal{X}$, and RKHS norm:

$$\forall f^{\mathcal{X}} \in \mathcal{H}_\sigma^{\mathcal{X}}, \quad \| f^{\mathcal{X}}\|_{\mathcal{H}_\sigma^{\mathcal{X}}} := \inf\left\{\| f\|_{\mathcal{H}_\sigma} : f \in \mathcal{H}_\sigma \text{ and } f_{|\mathcal{X}} = f^{\mathcal{X}}\right\} . \tag{27}$$

For any $f^{\mathcal{X}} \in \mathcal{H}_\sigma^{\mathcal{X}}$ consider the following risks:

$$\begin{aligned} R_{\phi,\sigma}^{\mathcal{X}}(f^{\mathcal{X}}) &:= \mathbb{E}_{P_{|\mathcal{X}}}\left[\phi\left(Y f^{\mathcal{X}}(X)\right)\right] + \lambda\| f^{\mathcal{X}}\|_{\mathcal{H}_\sigma^{\mathcal{X}}}^2 , \\ \widehat{R}_{\phi,\sigma}^{\mathcal{X}}(f^{\mathcal{X}}) &:= \frac{1}{n}\sum_{i=1}^n \phi\left(Y_i f^{\mathcal{X}}(X_i)\right) + \lambda\| f^{\mathcal{X}}\|_{\mathcal{H}_\sigma^{\mathcal{X}}}^2 . \end{aligned}$$

We first show that the sample error is the same in $\mathcal{H}_\sigma$ and $\mathcal{H}_\sigma^{\mathcal{X}}$:

Lemma 16 *Let $f^X_{\phi,\sigma}$ and $\hat{f}^X_{\phi,\sigma}$ be respectively the minimizers of $R^X_{\phi,\sigma}$ and $\widehat{R}^X_{\phi,\sigma}$. Then it holds almost surely that*

$$R_{\phi,\sigma}\left(f_{\phi,\sigma}\right) = R^X_{\phi,\sigma}\left(f^X_{\phi,\sigma}\right) ,$$
$$R_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) = R^X_{\phi,\sigma}\left(\hat{f}^X_{\phi,\sigma}\right) .$$

From Lemma 16 we deduce that a.s.,

$$R_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) - R_{\phi,\sigma}\left(f_{\phi,\sigma}\right) = R^X_{\phi,\sigma}\left(\hat{f}^X_{\phi,\sigma}\right) - R^X_{\phi,\sigma}\left(f^X_{\phi,\sigma}\right) . \tag{28}$$

In order to upper bound this term, we use concentration inequalities based on local Rademacher complexities (Bartlett et al., 2006, 2005; Steinwart and Scovel, 2004). In this approach, a crucial role is played by the covering number of a functional class $\mathcal{F}$ under the empirical L_2-norm. Remember that for a given sample $T := \{(X_1,Y_1),\ldots,(X_n,Y_n)\}$ and $\varepsilon > 0$, an ε-cover for the empirical L_2 norm is a family of function $(f_i)_{i\in I}$ such that:

$$\forall f \in \mathcal{F}\,, \exists i \in I, \quad \left(\frac{1}{n}\sum_{j=1}^{n}\left(f\left(X_j\right) - f_i\left(X_j\right)\right)^2\right)^{\frac{1}{2}} \leq \varepsilon\,.$$

The covering number $\mathcal{N}\left(\mathcal{F}\,,\varepsilon,L_2(T)\right)$ is then defined as the smallest cardinal of an ε-cover.

We can now mention the following result, adapted to our notation and setting, that exactly fits our need.

Theorem 17 (see Steinwart and Scovel, 2004, Theorem 5.8.) *For $\sigma > 0$, let $\mathcal{F}_\sigma$ be a convex subset of $\mathcal{H}^X_\sigma$ and let ϕ be a convex loss function. Define $\mathcal{G}_\sigma$ as follows:*

$$\mathcal{G}_\sigma := \left\{ g_f(x,y) = \phi(yf(x)) + \lambda \| f \|^2_{\mathcal{H}^X_\sigma} - \phi(y f^X_{\phi,\sigma}(x)) - \lambda \| f^X_{\phi,\sigma} \|^2_{\mathcal{H}^X_\sigma} \;:\; f \in \mathcal{F}_\sigma \right\} . \tag{29}$$

where $f^X_{\phi,\sigma}$ minimizes $R^X_{\phi,\sigma}$ over $\mathcal{F}_\sigma$. Suppose that there are constants $c \geq 0$ and $B > 0$ such that, for all $g \in \mathcal{G}_\sigma$,

$$\mathbb{E}_P\left[g^2\right] \leq c\mathbb{E}_P\left[g\right] ,$$

and

$$\| g \|_{L_\infty} \leq B\,.$$

Furthermore, assume that there are constants $a \geq 1$ and $0 < p < 2$ with

$$\sup_{T\in Z^n} \log \mathcal{N}\left(B^{-1}\mathcal{G}_\sigma, \varepsilon, L_2(T)\right) \leq a\varepsilon^{-p} \tag{30}$$

for all $\varepsilon > 0$. Then there exists a constant $c_p > 0$ depending only on p such that for all $n \geq 1$ and all $x \geq 1$ we have

$$Pr^*\left(T \in Z^n \;:\; R^X_{\phi,\sigma}(\hat{f}^X_{\phi,\sigma}) > R^X_{\phi,\sigma}(f^X_{\phi,\sigma}) + c_p\varepsilon(n,a,B,c,x)\right) \leq e^{-x}\,, \tag{31}$$

where

$$\varepsilon(n,a,B,c,p,x) := \left(B + B^{\frac{2p}{2+p}}c^{\frac{2-p}{2+p}}\right)\left(\frac{a}{n}\right)^{\frac{2}{2+p}} + (B+c)\frac{x}{n}\,.$$

Note that we use the outer probability $\Pr^*$ in (31) because the argument is not necessarily measurable. From the inequalities $\| f^X_{\phi,\sigma} \|^2_{\mathcal{H}^X_\sigma} \leq \phi(0)/\lambda$ and $\| \hat{f}^X_{\phi,\sigma} \|^2_{\mathcal{H}^X_\sigma} \leq \phi(0)/\lambda$, we see that it is enough to take

$$\mathcal{F}_\sigma = \sqrt{\frac{\phi(0)}{\lambda}} \mathcal{B}^X_\sigma \ , \tag{32}$$

where $\mathcal{B}^X_\sigma$ is the unit ball of $\mathcal{H}^X_\sigma$, to derive a control of (28) from Theorem 17. In order to apply this theorem we now provide uniform upper bounds over $\mathcal{G}_\sigma$ for the variance of g and its uniform norm, as well as an upper bound on the covering number of $\mathcal{G}_\sigma$.

Lemma 18 *For all $\sigma > 0$, for all $g \in \mathcal{G}_\sigma$,*

$$\mathbb{E}_P \left[g^2 \right] \leq \left(L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} \right) \sqrt{\kappa_\sigma} + 2\sqrt{\lambda \phi(0)} \right)^2 \frac{2}{\lambda} \mathbb{E}_P \left[g \right] \ . \tag{33}$$

Let us fix

$$B = 2L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} \right) \sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} + \phi(0) \ . \tag{34}$$

Then, the following two lemmas can be derived:

Lemma 19 *For all $\sigma > 0$, for all $g \in \mathcal{G}_\sigma$,*

$$\| g \|_{L_\infty} \leq B. \tag{35}$$

Lemma 20 *For all $\sigma > 0$, $0 < p \leq 2$, $\delta > 0$, $\varepsilon > 0$, the following holds:*

$$\log \mathcal{N} \left(B^{-1} \mathcal{G}_\sigma, \varepsilon, L_2(T) \right) \leq c_2 \sigma^{-((1-p/2)(1+\delta))d} \varepsilon^{-p} \ , \tag{36}$$

where c_1 and c_2 are constants that depend neither on σ, nor on ε (but they depend on p, δ, d and λ).

Combining now the results of Lemmas 18, 19 and 20 allows to apply Theorem 17 with $\mathcal{F}_\sigma$ defined by (32), any $p \in [0,2]$, and the following parameters:

$$\begin{aligned} c &= \left(L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} \right) \sqrt{\kappa_\sigma} + 2\sqrt{\lambda \phi(0)} \right)^2 \frac{2}{\lambda} \ , \\ B &= 2L \left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} \right) \sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} + \phi(0) \ , \\ a &= c_2 \sigma^{-((1-p/2)(1+\delta))d} \ , \end{aligned}$$

from which we deduce Lemma 15. ■

5.2 Proofs of Auxiliary Lemmas

Proof of Lemma 16 Because the support of P is included in $\mathcal{X}$, the following trivial equality holds:

$$\forall f \in \mathcal{H}_\sigma, \quad \mathbb{E}_P\left[\phi\left(Yf\left(X\right)\right)\right] = \mathbb{E}_{P_{|\mathcal{X}}}\left[\phi\left(Yf_{|\mathcal{X}}\left(X\right)\right)\right] . \tag{37}$$

Using first the definition of the restricted RKHS (26), then (37) and (27), we obtain

$$\begin{aligned} R^{\mathcal{X}}_{\phi,\sigma}\left(f^{\mathcal{X}}_{\phi,\sigma}\right) &= \inf_{f^{\mathcal{X}} \in \mathcal{H}^{\mathcal{X}}_\sigma} \mathbb{E}_{P_{|\mathcal{X}}}\left[\phi\left(Yf^{\mathcal{X}}\left(X\right)\right)\right] + \lambda \| f^{\mathcal{X}} \|_{\mathcal{H}^{\mathcal{X}}_\sigma} \\ &= \inf_{f \in \mathcal{H}_\sigma} \mathbb{E}_{P_{|\mathcal{X}}}\left[\phi\left(Yf_{|\mathcal{X}}\left(X\right)\right)\right] + \lambda \| f_{|\mathcal{X}} \|_{\mathcal{H}^{\mathcal{X}}_\sigma} \\ &= \inf_{f \in \mathcal{H}_\sigma} \mathbb{E}_P\left[\phi\left(Yf\left(X\right)\right)\right] + \lambda \| f \|_{\mathcal{H}_\sigma} \\ &= R_{\phi,\sigma}\left(f_{\phi,\sigma}\right) , \end{aligned} \tag{38}$$

which proves the first statement.

In order to prove the second statement, let us first observe that with probability 1, $X_i \in \mathcal{X}$ for $i = 1,\ldots,n$, and therefore:

$$\forall f \in \mathcal{H}_\sigma, \quad \frac{1}{n}\sum_{i=1}^n \phi\left(Y_i f\left(X_i\right)\right) = \frac{1}{n}\sum_{i=1}^n \phi\left(Y_i f_{|\mathcal{X}}\left(X_i\right)\right) , \tag{39}$$

from which we can conclude, using the same line of proof as (38), that the following holds a.s.:

$$\widehat{R}_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) = \widehat{R}^{\mathcal{X}}_{\phi,\sigma}\left(\hat{f}^{\mathcal{X}}_{\phi,\sigma}\right) .$$

Let us now show that this implies the following equality:

$$\hat{f}^{\mathcal{X}}_{\phi,\sigma} = \hat{f}_{\phi,\sigma|\mathcal{X}} . \tag{40}$$

Indeed, on the one hand, $\| \hat{f}_{\phi,\sigma|\mathcal{X}} \|_{\mathcal{H}^{\mathcal{X}}_\sigma} \leq \| \hat{f}_{\phi,\sigma} \|_{\mathcal{H}_\sigma}$ by (27). On the other hand, (39) implies that

$$\frac{1}{n}\sum_{i=1}^n \phi\left(Y_i \hat{f}_{\phi,\sigma}(X_i)\right) = \frac{1}{n}\sum_{i=1}^n \phi\left(Y_i \hat{f}_{\phi,\sigma|\mathcal{X}}\left(X_i\right)\right) .$$

As a result, we get $\widehat{R}^{\mathcal{X}}_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma|\mathcal{X}}\right) \leq \widehat{R}_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) = \widehat{R}^{\mathcal{X}}_{\phi,\sigma}\left(\hat{f}^{\mathcal{X}}_{\phi,\sigma}\right)$, from which we deduce that $\hat{f}_{\phi,\sigma|\mathcal{X}}$ and $\hat{f}^{\mathcal{X}}_{\phi,\sigma}$ both minimize the strictly convex functional $\widehat{R}^{\mathcal{X}}_{\phi,\sigma}$ on $\mathcal{H}^{\mathcal{X}}_\sigma$, proving (40). We also deduce from $\widehat{R}^{\mathcal{X}}_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma|\mathcal{X}}\right) = \widehat{R}_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right)$ and from (39) that

$$\| \hat{f}_{\phi,\sigma|\mathcal{X}} \|_{\mathcal{H}^{\mathcal{X}}_\sigma} = \| \hat{f}_{\phi,\sigma} \|_{\mathcal{H}_\sigma} . \tag{41}$$

Now, using (40), (37), then (41), we can conclude the proof of the second statement as follows:

$$\begin{aligned} R^{\mathcal{X}}_{\phi,\sigma}\left(\hat{f}^{\mathcal{X}}_{\phi,\sigma}\right) &= R^{\mathcal{X}}_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma|\mathcal{X}}\right) \\ &= \mathbb{E}_{P_{|\mathcal{X}}}\left[\phi\left(Y\hat{f}_{\phi,\sigma|\mathcal{X}}\left(X\right)\right)\right] + \lambda \| \hat{f}_{\phi,\sigma|\mathcal{X}} \|_{\mathcal{H}^{\mathcal{X}}_\sigma} \\ &= \mathbb{E}_P\left[\phi\left(Y\hat{f}_{\phi,\sigma}\left(X\right)\right)\right] + \lambda \| \hat{f}_{\phi,\sigma} \|_{\mathcal{H}^{\mathcal{X}}_\sigma} \\ &= R_{\phi,\sigma}\left(\hat{f}_{\phi,\sigma}\right) , \end{aligned}$$

concluding the proof of Lemma 16. ■

Proof of Lemma 18 We prove the uniform upper bound on the variances of the excess-loss functions in terms of their expectation, using an approach similar to but slightly simpler than Bartlett et al. (2006, Lemma 14) and Steinwart and Scovel (2004, Proposition 6.1). First we observe, using (17) and the fact that $\mathcal{F}_\sigma \subset \sqrt{\phi(0)/\lambda}B_\sigma$, that for any $f \in \mathcal{F}_\sigma$,

$$\begin{aligned} \| f \|_{L_\infty} &\leq \sqrt{\kappa_\sigma} \| f \|_{\mathcal{H}_\sigma} \\ &\leq \sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}. \end{aligned}$$

As a result, for any $(x,y) \in \mathcal{X} \times \{-1,+1\}$,

$$\begin{aligned} \left| g_f(x,y) \right| &\leq \left| \phi\left(y f(x)\right) - \phi\left(y f_{\phi,\sigma}(x)\right) \right| + \lambda \left| \| f \|_{\mathcal{H}_\sigma}^2 - \| f_{\phi,\sigma} \|_{\mathcal{H}_\sigma}^2 \right| \\ &\leq L\left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right) \left| f(x) - f_{\phi,\sigma}(x) \right| + \lambda \| f - f_{\phi,\sigma} \|_{\mathcal{H}_\sigma} \| f + f_{\phi,\sigma} \|_{\mathcal{H}_\sigma} \\ &\leq \left(L\left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right) \sqrt{\kappa_\sigma} + 2\sqrt{\lambda \phi(0)} \right) \| f - f_{\phi,\sigma} \|_{\mathcal{H}_\sigma}. \end{aligned} \tag{42}$$

Taking the square on both sides of this inequality and averaging with respect to P leads to:

$$\forall f \in \mathcal{F}_\sigma, \quad \mathbb{E}_P\left[g_f^2\right] \leq \left(L\left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right) \sqrt{\kappa_\sigma} + 2\sqrt{\lambda \phi(0)} \right)^2 \| f - f_{\phi,\sigma} \|_{\mathcal{H}_\sigma}^2. \tag{43}$$

On the other hand, we deduce from the convexity of ϕ that for any $(x,y) \in \mathcal{X} \times \{-1,+1\}$ and any $f \in \mathcal{F}_\sigma$:

$$\begin{aligned} &\frac{\phi(y f(x)) + \lambda \| f \|_{\mathcal{H}_\sigma}^2 + \phi(y f_{\phi,\sigma}(x)) + \lambda \| f_{\phi,\sigma} \|_{\mathcal{H}_\sigma}^2}{2} \\ &\qquad \geq \phi\left(\frac{y f(x) + y f_{\phi,\sigma}(x)}{2}\right) + \lambda \frac{\| f \|_{\mathcal{H}_\sigma}^2 + \| f_{\phi,\sigma} \|_{\mathcal{H}_\sigma}^2}{2} \\ &\qquad = \phi\left(y \frac{f + f_{\phi,\sigma}}{2}(x)\right) + \lambda \| \frac{f + f_{\phi,\sigma}}{2} \|_{\mathcal{H}_\sigma}^2 + \lambda \| \frac{f - f_{\phi,\sigma}}{2} \|_{\mathcal{H}_\sigma}^2. \end{aligned}$$

Averaging this inequality with respect to P rewrites:

$$\begin{aligned} \frac{R_{\phi,\sigma}(f) + R_{\phi,\sigma}\left(f_{\phi,\sigma}\right)}{2} &\geq R_{\phi,\sigma}\left(\frac{f + f_{\phi,\sigma}}{2}\right) + \lambda \| \frac{f - f_{\phi,\sigma}}{2} \|_{\mathcal{H}_\sigma}^2 \\ &\geq R_{\phi,\sigma}\left(f_{\phi,\sigma}\right) + \lambda \| \frac{f - f_{\phi,\sigma}}{2} \|_{\mathcal{H}_\sigma}^2, \end{aligned}$$

where the second inequality is due to the definition of $f_{\phi,\sigma}$ as a minimizer of $R_{\phi,\sigma}$. Therefore we get, for any $f \in \mathcal{F}_\sigma$,

$$\begin{aligned} \mathbb{E}_P\left[g_f\right] &= R_{\phi,\sigma}(f) - R_{\phi,\sigma}\left(f_{\phi,\sigma}\right) \\ &\geq \frac{\lambda}{2} \| f - f_{\phi,\sigma} \|_{\mathcal{H}_\sigma}^2. \end{aligned} \tag{44}$$

Combining (43) and (44) finishes the proof of Lemma 18 ■

Proof of Lemma 19 Following a path similar to (42), we can write for any $f \in \mathcal{F}_\sigma$ and any $(x,y) \in \mathcal{X} \times \{-1,+1\}$:

$$\begin{aligned} \left| g_f(x,y) \right| &\leq \left| \phi(yf(x)) - \phi\left(yf_{\phi,\sigma}(x)\right) \right| + \lambda \left| \| f \|^2_{\mathcal{H}_\sigma} - \| f_{\phi,\sigma} \|^2_{\mathcal{H}_\sigma} \right| \\ &\leq L\left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right) \left| f(x) - f_{\phi,\sigma}(x) \right| + \lambda \frac{\phi(0)}{\lambda} \\ &\leq 2L\left(\sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}}\right) \sqrt{\frac{\kappa_\sigma \phi(0)}{\lambda}} + \phi(0) \, . \end{aligned}$$

■

Proof of Lemma 20 Let us introduce the notation $l_\phi \circ f(x,y) := \phi(y(f(x))$ and $L_{\phi,\sigma} \circ f := l_\phi \circ f + \lambda \| f \|^2_{\mathcal{H}^{\mathcal{X}}_\sigma}$, for $f \in \mathcal{H}^{\mathcal{X}}_\sigma$ and $(x,y) \in \mathcal{X} \times \{-1,1\}$. We can then rewrite (29) as:

$$\mathcal{G}_\sigma = \left\{ L_{\phi,\sigma} \circ f - L_{\phi,\sigma} \circ f^{\mathcal{X}}_{\phi,\sigma} \, : \, f \in \mathcal{F}_\sigma \right\} \, .$$

The covering number of a set does not change when the set is translated by a single function, therefore:

$$\mathcal{N}\left(B^{-1}\mathcal{G}_\sigma, \varepsilon, L_2(T)\right) = \mathcal{N}\left(B^{-1} L_{\phi,\sigma} \circ \mathcal{F}_\sigma, \varepsilon, L_2(T)\right) \, .$$

Denoting now $[a,b]$ the set of constant functions with values between a and b, we deduce, from the fact that $\lambda \| f \|^2_{\mathcal{H}^{\mathcal{X}}_\sigma} \leq \phi(0)$ for $f \in \mathcal{F}_\sigma$, that

$$B^{-1} L_{\phi,\sigma} \circ \mathcal{F}_\sigma \subset B^{-1} l_\phi \circ \mathcal{F}_\sigma + \left[0, B^{-1}\phi(0)\right] \, .$$

Using the sub-additivity of the entropy we therefore get:

$$\log \mathcal{N}\left(B^{-1}\mathcal{G}_\sigma, 2\varepsilon, L_2(T)\right) \leq \log \mathcal{N}\left(B^{-1} l_\phi \circ \mathcal{F}_\sigma, \varepsilon, L_2(T)\right) + \log \mathcal{N}\left(\left[0, B^{-1}\phi(0)\right], \varepsilon, L_2(T)\right) \, . \quad (45)$$

In order to upper bound the first term in the r.h.s. of (45), we observe, using (17), that for any $f \in \mathcal{F}_\sigma$ and $x \in \mathcal{X}$,

$$|f(x)| \leq \sqrt{\kappa_\sigma} \| f \|_{\mathcal{H}^{\mathcal{X}}_\sigma} \leq \sqrt{\frac{\phi(0)\kappa_\sigma}{\lambda}} \, ,$$

and therefore a simple computation shows that, if $u(x,y) := B^{-1}\phi(yf(x))$ and $u'(x,y) := B^{-1}\phi(yf'(x))$ are two elements of $B^{-1} l_\phi \circ \mathcal{F}_\sigma$ (with $f, f' \in \mathcal{F}_\sigma$), then for any sample T:

$$\| u - u' \|_{L_2(T)} \leq B^{-1} L\left(\sqrt{\frac{\phi(0)\kappa_\sigma}{\lambda}}\right) \| f - f' \|_{L_2(T)} \, .$$

and therefore

$$\begin{aligned} \log \mathcal{N}\left(B^{-1} l_\phi \circ \mathcal{F}_\sigma, \varepsilon, L_2(T)\right) &\leq \log \mathcal{N}\left(\mathcal{F}_\sigma, B\varepsilon L\left(\sqrt{\frac{\phi(0)\kappa_\sigma}{\lambda}}\right)^{-1}, L_2(T)\right) \\ &\leq \log \mathcal{N}\left(\mathcal{B}^{\mathcal{X}}_\sigma, B\varepsilon L\left(\sqrt{\frac{\phi(0)\kappa_\sigma}{\lambda}}\right)^{-1} \sqrt{\frac{\lambda}{\phi(0)}}, L_2(T)\right) \, . \end{aligned} \quad (46)$$

Recalling the definition of B in (34), we obtain:

$$B\varepsilon L\left(\sqrt{\frac{\phi(0)\kappa_\sigma}{\lambda}}\right)^{-1}\sqrt{\frac{\lambda}{\phi(0)}} \geq 2\varepsilon\sqrt{\kappa_\sigma}\,,$$

and therefore

$$\log\mathcal{N}\left(B^{-1}l_\phi\circ\mathcal{F}_\sigma,\varepsilon,L_2(T)\right) \leq \log\mathcal{N}\left(\mathcal{B}_\sigma^X,2\varepsilon\sqrt{\kappa_\sigma},L_2(T)\right)\,.$$

The second term in the r.h.s. of (45) is easily upper bounded by:

$$\log\mathcal{N}\left(\left[0,B^{-1}\phi(0)\right],\varepsilon,L_2(T)\right) \leq \log\left(\frac{\phi(0)}{B\varepsilon}\right)\,,$$

and we finally get:

$$\log\mathcal{N}\left(B^{-1}\mathcal{G}_\sigma,2\varepsilon,L_2(T)\right) \leq \log\mathcal{N}\left(\mathcal{B}_\sigma^X,2\varepsilon\sqrt{\kappa_\sigma},L_2(T)\right)+\log\left(\frac{\phi(0)}{B\varepsilon}\right)\,. \tag{47}$$

We now need to upper bound the covering number of the unit ball in the RKHS. We make use of the following result, proved by Steinwart and Scovel (2004, Theorem 2.1, page 5): if $\tilde{\mathcal{B}_\sigma^X}$ denotes the unit ball of the RKHS associated with the non-normalized Gaussian kernel (20) on a compact set, then for all $0<p\leq 2$ and all $\delta>0$ there exists a constant $c_{p,\delta,d}$ independent of σ such that for all $\tilde{\varepsilon}>0$ we have:

$$\log\mathcal{N}\left(\tilde{\mathcal{B}_\sigma^X},\tilde{\varepsilon},L_2(T)\right) \leq c_{p,\delta,d}\sigma^{(1-p/2)(1+\delta)d}\varepsilon^{-p}\,. \tag{48}$$

Now, using (21), we observe that

$$\mathcal{B}_\sigma^X = \sqrt{\kappa_\sigma}\tilde{\mathcal{B}_\sigma^X}\,,$$

and therefore:

$$\begin{aligned}\log\mathcal{N}\left(\mathcal{B}_\sigma^X,2\varepsilon\sqrt{\kappa_\sigma},L_2(T)\right) &= \log\mathcal{N}\left(\sqrt{\kappa_\sigma}\tilde{\mathcal{B}_\sigma^X},2\varepsilon\sqrt{\kappa_\sigma},L_2(T)\right)\\ &= \log\mathcal{N}\left(\tilde{\mathcal{B}_\sigma^X},2\varepsilon,L_2(T)\right)\,.\end{aligned} \tag{49}$$

Plugging (48) into (49), and (49) into (47) finally leads to the announced result, after observing that the second term in the r.h.s. of (47) becomes negligible compared to the first one and can therefore be hidden in the constant for ε small enough. ■

6. Some Properties of the L_2-Norm-Regularized ϕ-Risk

In this section we investigate the conditions on the loss function ϕ under which the Bayes consistency of the minimization of the regularized ϕ-risk holds. In the spirit of Bartlett et al. (2006), we introduce a notion of classification-calibration for regularized loss functions ϕ, and upper bound the excess risk of any classifier f in terms of its excess of regularized ϕ-risk. We also upper-bound the L_2-distance between f and $f_{\phi,0}$ in terms of the excess of regularized ϕ-risk of f, which is useful to proove Bayes consistency in the one-class setting.

6.1 Classification Calibration

In the classical setting, Bartlett et al. (2006, Definition 1, page 7) introduce the following notion of classification-calibrated loss functions:

Definition 21 *For any* $(\eta,\alpha)\in[0,1]\times\mathbb{R}$*, let the generic conditional* ϕ*-risk be defined by:*

$$C_\eta(\alpha) := \eta\phi(\alpha)+(1-\eta)\,\phi(-\alpha).$$

The loss function ϕ *is said to be classification-calibrated if, for any* $\eta\in[0,1]\backslash\{1/2\}$*:*

$$\inf_{\alpha\in\mathbb{R}:\alpha(2\eta-1)\leq 0} C_\eta(\alpha) > \inf_{\alpha\in\mathbb{R}} C_\eta(\alpha)$$

The importance here is in the *strict* inequality, which implies in particular that if the global infimum of C_η is reached at some point α, then $\alpha>0$ (resp. $\alpha<0$) if $\eta>1/2$ (resp. $\eta<1/2$). This condition, that generalizes the requirement that the minimizer of $C_\eta(\alpha)$ has the correct sign, is a minimal condition that can be viewed as a pointwise form of Fisher consistency for classification. In our case, noting that for any $f\in\mathcal{M}$, the L_2-regularized ϕ-risk can be rewritten as follows:

$$R_{\phi,0}(f)=\int_{\mathbb{R}^d}\left\{[\eta(x)\phi(f(x))+(1-\eta(x))\,\phi(-f(x))]\,\rho(x)+\lambda f(x)^2\right\}dx\,,$$

we introduce the regularized generic conditional ϕ-risk:

$$\forall(\eta,\rho,\alpha)\in[0,1]\times(0,+\infty)\times\mathbb{R},\quad C_{\eta,\rho}(\alpha):=C_\eta(\alpha)+\frac{\lambda\alpha^2}{\rho}\,,$$

as well as the related weighted regularized generic conditional ϕ-risk:

$$\forall(\eta,\rho,\alpha)\in[0,1]\times[0,+\infty)\times\mathbb{R},\quad G_{\eta,\rho}(\alpha):=\rho C_\eta(\alpha)+\lambda\alpha^2\,.$$

This leads to the following notion of classification-calibration:

Definition 22 *We say that* ϕ *is classification calibrated for the regularized risk, or* R-classification-calibrated*, if for any* $(\eta,\rho)\in[0,1]\backslash\{1/2\}\times(0,+\infty)$

$$\inf_{\alpha\in\mathbb{R}:\alpha(2\eta-1)\leq 0} C_{\eta,\rho}(\alpha) > \inf_{\alpha\in\mathbb{R}} C_{\eta,\rho}(\alpha)$$

The following result clarifies the relationship between the properties of classification-calibration and R-classification-calibration.

Lemma 23 *For any function* $\phi:\mathbb{R}\to[0,+\infty)$*,* $\phi(x)$ *is R-classification-calibrated if and only if for any* $t>0$*,* $\phi(x)+tx^2$ *is classification-calibrated.*

Proof For any $\phi:\mathbb{R}\to[0,+\infty)$ and $\rho>0$, let $\widetilde{\phi}(x):=\phi(x)+\lambda x^2/\rho$ and $\widetilde{C}_\eta$ the corresponding generic conditional $\widetilde{\phi}$-risk. Then one easily gets, for any $\alpha\in\mathbb{R}$

$$\widetilde{C}_\eta(\alpha)=C_{\eta,\rho}(\alpha)\,.$$

As a result, ϕ is R-classification-calibrated if and only if, for any ρ, $\widetilde{\phi}$ is classification-calibrated, which proves the lemma. ■

Classification-calibration and R-classification-calibration are two different properties related to each other by Lemma 23, but none of them implies the other one for general non-convex functions ϕ, as illustrated by the following two examples.

Example 1 ***: A classification-calibrated, but not R-classification-calibrated function.*** *Let* $\phi(x) = 1$ *on* $(-\infty,-2]$, $\phi(x) = 2$ *on* $[-1,1]$, $\phi(x) = 0$ *on* $[2,+\infty)$, *and* ϕ *continous linear on* $[-2,-1]$ *and* $[1,2]$. *Then* $C_\eta(\alpha)$ *is also continuous and piecewise linear on the intervals delimited by the points* $-2,-1,1,2$, *with values* η *on* $(-\infty,-2]$, $1-\eta$ *on* $[2,+\infty)$, *and* 2 *on* $[-1,1]$. *As a result,* $\inf_{\alpha\in\mathbb{R}} C_\eta(\alpha) = \min(\eta, 1-\eta)$ *and* $\inf_{\alpha\in\mathbb{R}:\alpha(2\eta-1)\leq 0} C_\eta(\alpha) = \max(\eta, 1-\eta)$. *This shows that* ϕ is classification-calibrated. *However, as soon as* $\rho < 2\lambda$, *the global minimum of* $C_{\eta,\rho}(\alpha)$ *is reached for* $\alpha = 0$ *and therefore:*

$$\inf_{\alpha\in\mathbb{R}:\alpha(2\eta-1)\leq 0} C_{\eta,\rho}(\alpha) = \inf_{\alpha\in\mathbb{R}} C_{\eta,\rho}(\alpha) = 2,$$

which shows that ϕ is not R-classification-calibrated *in this case.*

Example 2 ***: A R-classification-calibrated, but not classification-calibrated function.*** *Let* $\phi : \mathbb{R} \to [0,+\infty)$ *be any function with negative right-hand and positive left-hand derivatives at* 0, *satisfying*

$$\lim_{\alpha\to-\infty} \phi(\alpha) = \lim_{\alpha\to\infty} \phi(\alpha) = 0\,, \tag{50}$$

and

$$\forall \alpha > 0, \quad \phi(\alpha) < \phi(-\alpha)\,. \tag{51}$$

An example of a function that satisfies these conditions is $\phi(\alpha) = e^{\alpha}$ *for* $\alpha \leq 0$, $\phi(\alpha) = e^{-2\alpha}$ *for* $\alpha \geq 0$. *Because of (50), it is clear that such functions satisfy*

$$\inf_{\alpha\leq 0} C_\eta(\alpha) = \inf_{\alpha\geq 0} C_\eta(\alpha) = \inf_{\alpha\in\mathbb{R}} C_\eta(\alpha) = 0\,,$$

which shows that they are not classification-calibrated. In order to show that they are R-classification-calibrated, it suffices to show by Lemma 23 that for any $t > 0$, $\phi(x) + tx^2$ *is classification-calibrated.* ϕ *being nonnegative, the corresponding generic conditional risk*

$$\widetilde{C}_\eta(\alpha) = \eta\phi(\alpha) + (1-\eta)\phi(-\alpha) + t\alpha^2$$

satisfies:

$$\lim_{\alpha\to-\infty} \widetilde{C}_\eta(\alpha) = \lim_{\alpha\to\infty} \widetilde{C}_\eta(\alpha) = +\infty\,.$$

As a result, for any $\eta \neq 1/2$, *the infimum of* $\widetilde{C}_\eta$ *over* $\{\alpha \in \mathbb{R} : \alpha(2\eta-1) \leq 0\}$ *is reached at some finite point* $\tilde{\alpha}_\eta$. *Moreover,* $\widetilde{C}_\eta$ *has negative right-hand and positive left-hand derivatives at* 0, *ensuring that the minimum is not reached at* 0: $\tilde{\alpha}_\eta(2\eta-1) < 0$. *This implies by (51) that*

$$(2\eta-1)\left(\phi(\tilde{\alpha}_\eta) - \phi(-\tilde{\alpha}_\eta)\right) > 0.$$

Combining this with the following equality holding for any $\alpha \in \mathbb{R}$*:*

$$\widetilde{C}_\eta(\alpha) - \widetilde{C}_\eta(-\alpha) = (2\eta-1)\left(\phi(\alpha) - \phi(-\alpha)\right)\,,$$

we obtain

$$\widetilde{C}_\eta(\tilde{\alpha}_\eta) > \widetilde{C}_\eta(-\tilde{\alpha}_\eta)\,.$$

As a result,

$$\inf_{\alpha\in\mathbb{R}:\alpha(2\eta-1)\leq 0} C_{\eta,\rho}(\alpha) = \widetilde{C}_\eta(\tilde{\alpha}_\eta) > \widetilde{C}_\eta(-\tilde{\alpha}_\eta) \geq \inf_{\alpha\in\mathbb{R}} C_{\eta,\rho}(\alpha)\,,$$

showing that $\phi(x) + tx^2$ *is classification-calibrated, and therefore that* ϕ *is R-classification-calibrated.*

6.2 Classification-Calibration of Convex Loss Functions

The following lemma states the equivalence between classification calibration and R-classification calibration for convex loss functions, and it gives a simple characterization of this property.

Lemma 24 *For a convex function $\phi : \mathbb{R} \to [0, +\infty)$, the following properties are equivalent:*

1. *ϕ is classification-calibrated,*
2. *ϕ is R-classification-calibrated,*
3. *ϕ is differentiable at 0 and $\phi'(0) < 0$.*

Proof The equivalence of the first and the third properties is shown in Bartlett et al. (2006, Theorem 4). From this and lemma 23, we deduce that ϕ is R-classification-calibrated iff $\phi(x) + tx^2$ is classification-calibrated for any $t > 0$, iff $\phi(x) + tx^2$ is differentiable at 0 with negative derivative (for any $t > 0$), iff $\phi(x)$ is differentiable at 0 with negative derivative. This proves the equivalence between the second and third properties. ■

6.3 Some Properties of the Minimizer of the $R_{\phi,0}$-Risk

When ϕ is convex, the function $C_\eta(\alpha)$ is a convex function (as a convex combination of convex functions), and therefore $G_{\eta,\rho}(\alpha)$ is strictly convex and diverges to $+\infty$ in $-\infty$ and $+\infty$; as a result, for any $(\eta, \rho) \in [0,1] \times [0, +\infty)$, there exists a unique $\alpha(\eta, \rho)$ that minimizes $G_{\eta,\rho}$ on $\mathbb{R}$. It satisfies the following inequality:

Lemma 25 *If $\phi : \mathbb{R} \to [0, +\infty)$ is a convex function, then for any $(\eta, \rho) \in [0,1] \times [0, +\infty)$ and any $\alpha \in \mathbb{R}$,*

$$G_{\eta,\rho}(\alpha) - G_{\eta,\rho}(\alpha(\eta,\rho)) \geq \lambda (\alpha - \alpha(\eta,\rho))^2 \ . \tag{52}$$

Proof For any $(\eta, \rho) \in [0,1] \times [0, +\infty)$, the function $G_{\eta,\rho}(\alpha)$ is the sum of the convex function $\rho C_\eta(\alpha)$ and of the strictly convex function $\lambda\alpha^2$. Let us denote by $C_\eta^+(\alpha)$ the right-hand derivative of C_η at the point α (which is well defined for convex functions). The right-hand derivative of a convex function being non-negative at a minimum, we have (denoting $\alpha_* := \alpha(\eta, \rho)$):

$$\rho C_\eta^+(\alpha_*) + 2\lambda\alpha_* \geq 0 \ . \tag{53}$$

Now, for any $\alpha > \alpha_*$, we have by convexity of C_η:

$$C_\eta(\alpha) \geq C_\eta(\alpha_*) + (\alpha - \alpha_*) C_\eta^+(\alpha_*) \ . \tag{54}$$

Moreover, by direct calculation we get:

$$\lambda\alpha^2 = \lambda\alpha_*^2 + 2\lambda\alpha_*(\alpha - \alpha_*) + \lambda(\alpha - \alpha_*)^2 \ . \tag{55}$$

Mutliplying (54) by ρ, adding (55) and plugging (53) into the result leads to:

$$G_{\eta,\rho}(\alpha) - G_{\eta,\rho}(\alpha_*) \geq \lambda(\alpha - \alpha_*)^2 .$$

This inequality is also valid for $\alpha \leq \alpha_*$: starting this time from

$$\rho C_\eta^-(\alpha_*) + 2\lambda\alpha_* \leq 0 \,,$$

where C_η^- denotes the left-hand derivative of C_η, and from

$$C_\eta(\alpha) \geq C_\eta(\alpha_*) + (\alpha - \alpha_*) C_\eta^-(\alpha_*) \,,$$

which holds for any $\alpha < \alpha_*$ by convexity of C_η, we can draw exactly the same lines of reasoning as for the case $\alpha > \alpha^*$. ■

From this result we obtain the following characterization and properties of the minimizer of the $R_{\phi,0}$-risk:

Theorem 26 *If $\phi : \mathbb{R} \to [0,+\infty)$ is a convex function, then the function $f_{\phi,0} : \mathbb{R}^d \to \mathbb{R}$ defined for any $x \in \mathbb{R}^d$ by*

$$f_{\phi,0}(x) := \alpha(\eta(x), \rho(x))$$

satisfies:

1. *$f_{\phi,0}$ is measurable.*
2. *$f_{\phi,0}$ minimizes the $R_{\phi,0}$-risk:*

$$R_{\phi,0}\left(f_{\phi,0}\right) = \inf_{f \in \mathcal{M}} R_{\phi,0}(f) \,.$$

3. *For any $f \in \mathcal{M}$, the following holds:*

$$\| f - f_{\phi,0} \|_{L_2}^2 \leq \frac{1}{\lambda} \left(R_{\phi,0}(f) - R_{\phi,0}^* \right).$$

Proof To show that $f_{\phi,0}$ is measurable, it suffices to show that the mapping $(\eta,\rho) \in [0,1] \times [0,+\infty) \mapsto \alpha(\eta,\rho)$ is continuous. Indeed, if this is true, then $f_{\phi,0}$ is measurable as a continuous function of two measurable functions η and ρ.

In order to show the continuity of $(\eta,\rho) \mapsto \alpha(\eta,\rho)$, fix $(\eta_0,\rho_0) \in [0,1] \times [0,+\infty)$ and the corresponding $\alpha_0 := \alpha(\eta_0,\rho_0)$. Then, for any $\varepsilon > 0$, there exists a neighborhood B_ε of (η_0,ρ_0) such that for any $(\eta,\rho) \in B_\varepsilon$, for any $\alpha \in [\alpha_0 - \varepsilon, \alpha_0 + \varepsilon]$,

$$\left| G_{\eta,\rho}(\alpha) - G_{\eta_0,\rho_0}(\alpha) \right| < \frac{\lambda\varepsilon^2}{3} \,. \tag{56}$$

To see that, first note that the function ϕ is continuous and thus bounded by some constant A on $[\alpha_0 - \varepsilon, \alpha_0 + \varepsilon]$, and therefore, for any α in $[\alpha_0 - \varepsilon, \alpha_0 + \varepsilon]$, we have

$$\begin{aligned} \left| G_{\eta,\rho}(\alpha) - G_{\eta_0,\rho_0}(\alpha) \right| &= \left| (\eta\rho - \eta_0\rho_0)(\phi(\alpha) - \phi(-\alpha)) + (\rho - \rho_0)\phi(-\alpha) \right| \\ &\leq 2A \left(|\eta\rho - \eta_0\rho_0| + |\rho - \rho_0| \right) . \end{aligned}$$

Hence, (56) holds by taking, for instance,

$$B_\varepsilon := \left\{ (\eta,\rho) \in \mathbb{R}^2 \,:\, |\eta\rho - \eta_0\rho_0| < \lambda\varepsilon^2/12A \,,\, |\rho - \rho_0| < \lambda\varepsilon^2/12A \right\} .$$

Now, applying (56) successively to $\alpha_0+\varepsilon$ and to α_0, then using (52), we easily obtain that for any $(\eta,\rho)\in B_\varepsilon$,

$$G_{\eta,\rho}(\alpha_0+\varepsilon) > G_{\eta,\rho}(\alpha_0)+\frac{\lambda\varepsilon^2}{3}\,.$$

In the same way, applying (56) successively to $\alpha_0-\varepsilon$ and to α_0, then using (52), we obtain that for any $(\eta,\rho)\in B_\varepsilon$,

$$G_{\eta,\rho}(\alpha_0-\varepsilon) > G_{\eta,\rho}(\alpha_0)+\frac{\lambda\varepsilon^2}{3}\,.$$

This reveals the existence of two points around α_0, namely $\alpha_0-\varepsilon$ and $\alpha_0+\varepsilon$, at which the function $G_{\eta,\rho}$ takes values larger than $G_{\eta,\rho}(\alpha_0)$. By convexity of $G_{\eta,\rho}$, this implies that its minimizer, namely $\alpha(\eta,\rho)$, is in the interval $[\alpha_0-\varepsilon,\alpha_0+\varepsilon]$, as soon as $(\eta,\rho)\in B_\varepsilon$, which concludes the proof of the continuity of $(\eta,\rho)\to\alpha(\eta,\rho)$, and therefore of the measurability of $f_{\phi,0}$.

Now, we have by construction, for any $f\in\mathcal{M}$:

$$\forall x\in\mathbb{R}^d,\quad G_{\eta(x),\rho(x)}\left(f_{\phi,0}(x)\right)\le G_{\eta(x),\rho(x)}\left(f(x)\right)$$

which after integration leads to:

$$R_{\phi,0}(f_{\phi,0})\le R_{\phi,0}(f)\,,$$

proving the second statement of the theorem.

Finally, for any $f\in\mathcal{M}$, rewriting (52) with $\alpha=f(x)$, $\rho=\rho(x)$ and $\eta=\eta(x)$ shows that:

$$\forall x\in\mathbb{R}^d,\quad G_{\eta(x),\rho(x)}\left(f(x)\right)-G_{\eta(x),\rho(x)}\left(f_{\phi,0}(x)\right)\ge\lambda\left(f(x)-f_{\phi,0}(x)\right)^2\,,$$

which proves the third statement of Theorem 26. ■

6.4 Relating the $R_{\phi,0}$-Risk with the Classification Error Rate

In the "classical" setting (with a regularization parameter converging to 0), the idea of relating the convexified risk to the true risk (more simply called risk) has recently gained a lot of interest. Zhang (2004) and Lugosi and Vayatis (2004) upper bound the excess-risk by some function of the excess ϕ-risk to prove consistency of various algorithms (and obtain upper bounds for the rates of convergence of the risk to the Bayes risk). These ideas were then generalized by Bartlett et al. (2006), which we now adapt to our framework.

Let us define, for any $(\eta,\rho)\in[0,1]\times(0,+\infty)$,

$$M(\eta,\rho):=\min_{\alpha\in\mathbb{R}}C_{\eta,\rho}(\alpha)=C_{\eta,\rho}\left(\alpha(\eta,\rho)\right),$$

and for any $\rho>0$ the function ψ_ρ defined for all θ in $[0,1]$ by

$$\psi_\rho(\theta):=\phi(0)-M\left(\frac{1+\theta}{2},\rho\right).$$

The following lemma summarizes a few properties of M and ψ_ρ. Explicit computations for some standard loss functions are performed in Section 7.

Lemma 27 *If $\phi : \mathbb{R} \to [0,+\infty)$ is a convex function, then for any $\rho > 0$, the following properties hold:*

1. *the function $\eta \mapsto M(\eta,\rho)$ is symmetric around $1/2$, concave, and continuous on $[0,1]$;*
2. *ψ_ρ is convex, continuous, nonnegative, and nondecreasing on $[0,1]$, and $\psi(0) = 0$;*
3. *if $0 < \rho < \tau$, then $\psi_\rho \leq \psi_\tau$ on $[0,1]$;*
4. *ϕ is R-classification-calibrated if and only if $\psi_\rho(\theta) > 0$ for $\theta \in (0,1]$.*

Proof For any $\rho > 0$, let

$$\phi_\rho(x) := \phi(x) + \frac{\lambda x^2}{\rho}.$$

As already observed in the proof of Lemma 23, the corresponding generic conditional ϕ_ρ-risk $\widetilde{C}_\eta$ satisfies

$$\widetilde{C}'_\eta(\alpha) = C_{\eta,\rho}(\alpha)\ .$$

ϕ_ρ being convex, the first two points are direct consequences of Bartlett et al. (2006, Theorem 4 & Lemma 6). In particular,the function ψ_ρ is nondecreasing due to the fact that it is minimal at 0 and convex on $[0,1]$.

To prove the third point, it suffices to observe that for $0 < \rho \leq \tau$ we have for any $(\eta,\alpha) \in [0,1] \times \mathbb{R}$:

$$C_{\eta,\rho}(\alpha) - C_{\eta,\tau}(\alpha) = \lambda\alpha^2 \left(\frac{1}{\rho} - \frac{1}{\tau}\right) \geq 0,$$

which implies, by taking the minimum in α:

$$M(\eta,\rho) \geq M(\eta,\tau),$$

and therefore, for $\theta \in [0,1]$

$$\psi_\rho(\theta) \leq \psi_\tau(\theta).$$

Finally, by lemma 24, ϕ is R-classification-calibrated iff ϕ_ρ is classification-calibrated (because both properties are equivalent to saying that ϕ is differentiable at 0 and $\phi'(0) < 0$), iff $\psi_\rho(\theta) > 0$ for $\theta \in (0,1]$ by Bartlett et al. (2006, Theorem 6). ■

We are now in position to state a first result to relate the excess $R_{\phi,0}$-risk to the excess-risk. The dependence on $\rho(x)$ generates difficulties compared with the "classical" setting, which forces us to separate the low density regions from the rest in the analysis.

Theorem 28 *Suppose ϕ is a convex classification-calibrated function, and for any $\varepsilon > 0$, let*

$$A_\varepsilon := \left\{x \in \mathbb{R}^d : \rho(x) \leq \varepsilon\right\}.$$

For any $f \in \mathcal{M}$ the following holds:

$$R(f) - R^* \leq \inf_{\varepsilon > 0} \left\{P(A_\varepsilon) + \psi_\varepsilon^{-1}\left(R_{\phi,0}(f) - R^*_{\phi,0}\right)\right\} \tag{57}$$

Proof First note that for convex classification-calibrated functions, ψ_ε is strictly increasing on $[0,1]$. Indeed, by Lemma 27, it is convex and reaches its unique minimum at 0 in this case. Since ψ_ε is also continuous on $[0,1]$, it is therefore invertible, which justifies the use of ψ_ε^{-1} in (57).

Fix now a function $f \in \mathcal{M}$, and let $U(x) := 1$ if $f(x)(2\eta(x)-1) < 0$, 0 otherwise (U is the indicator function of the set where f and the optimal classifier disagree). For any $\varepsilon > 0$, if we define $B_\varepsilon := \mathbb{R}^d \backslash A_\varepsilon$, we can compute:

$$
\begin{aligned}
R_{\phi,0}(f) - R^*_{\phi,0} &= \int_{\mathbb{R}^d} \left[C_{\eta(x),\rho(x)}(f(x)) - M(\eta(x),\rho(x)) \right] \rho(x) dx \\
&\geq \int_{\mathbb{R}^d} \left[C_{\eta(x),\rho(x)}(f(x)) - M(\eta(x),\rho(x)) \right] U(x)\rho(x) dx \\
&\geq \int_{\mathbb{R}^d} \left[\phi(0) - M(\eta(x),\rho(x)) \right] U(x)\rho(x) dx \\
&= \int_{\mathbb{R}^d} \psi_{\rho(x)}(|2\eta(x)-1|) U(x)\rho(x) dx \\
&\geq \int_{B_\varepsilon} \psi_{\rho(x)}(|2\eta(x)-1|) U(x)\rho(x) dx \\
&\geq \int_{B_\varepsilon} \psi_{\varepsilon}(|2\eta(x)-1|) U(x)\rho(x) dx \\
&= \int_{B_\varepsilon} \psi_{\varepsilon}(U(x)|2\eta(x)-1|) \rho(x) dx \\
&= P(B_\varepsilon) \int_{B_\varepsilon} \psi_{\varepsilon}(U(x)|2\eta(x)-1|) \frac{\rho(x)}{P(B_\varepsilon)} dx \\
&\geq P(B_\varepsilon) \psi_\varepsilon \left(\frac{1}{P(B_\varepsilon)} \int_{B_\varepsilon} |2\eta(x)-1| U(x)\rho(x) dx \right) \\
&\geq \psi_\varepsilon \left(\int_{B_\varepsilon} |2\eta(x)-1| U(x)\rho(x) dx \right) \\
&= \psi_\varepsilon \left(\int_{\mathbb{R}^d} |2\eta(x)-1| U(x)\rho(x) dx - \int_{A_\varepsilon} |2\eta(x)-1| U(x)\rho(x) dx \right) \\
&\geq \psi_\varepsilon \left(\int_{\mathbb{R}^d} |2\eta(x)-1| U(x)\rho(x) dx - P(A_\varepsilon) \right) \\
&= \psi_\varepsilon \left(R(f) - R^* - P(A_\varepsilon) \right) ,
\end{aligned}
$$

where the successive (in)equalities are respectively justified by: (i) the definition of $R_{\phi,0}$ and the second point of Theorem 26; (ii) the fact that $U \leq 1$; (iii) the fact that when f and $2\eta - 1$ have different signs, then $C_{\eta,\rho}(f) \geq C_{\eta,\rho}(0) = \phi(0)$; (iv) the definition of ψ_ρ; (v) the obvious fact that $B_\varepsilon \subset \mathbb{R}^d$; (vi) the observation that, by definition, ρ is larger than ε on B_ε, and the third point of Lemma 27; (vii) the fact that $\psi_\varepsilon(0) = 0$ and $U(x) \in \{0,1\}$; (viii) a simple division and multiplication by $P(B_\varepsilon) > 0$; (ix) Jensen's inequality; (x) the convexity of ψ_ε and the facts that $\psi(0) = 0$ and $P(B_\varepsilon) < 1$; (xi) the fact that $B_\varepsilon = \mathbb{R}^d \backslash A_\varepsilon$; (xii) the upper bound $|2\eta(x)-1| U(x) \leq 1$ and the fact that ψ_ε is increasing; and (xiii) a classical inequality that can be found, e.g., in Devroye et al. (1996, Theorem 2.2, page 16). Composing each side by the strictly increasing function ψ_ε^{-1} leads to the announced result. ■

6.5 Proof of Theorem 4

Theorem 4 is a direct corollary of Theorem 28.[4] Indeed, keeping the notation of the previous section, let us choose for any $\delta > 0$ an ε small enough to ensure $P(A_\varepsilon) < \delta/2$, and $N \in \mathbb{N}$ such that for any $n > N$,

$$R_{\phi,0}(f_n) - R^*_{\phi,0} < \psi_\varepsilon\left(\frac{\delta}{2}\right).$$

Then a direct application of Theorem 28 in this case shows that for any $n > N$, $R(f_n) - R* < \delta$, concluding the proof of Theorem 4. ■

This important result shows that any consistency result for the regularized ϕ-risk implies consistency for the true risk, that is, convergence to the Bayes risk. Besides, convergence rates for the regularized ϕ-risk towards its minimum translate into convergence rates for the risk towards the Bayes risk thanks to (57), as we will show in the next subsection.

6.6 Refinements under a Low Noise Assumption

When the distribution P satisfies a low noise assumption as defined in section 2, we have the following result:

Theorem 29 *Let ϕ be a convex loss function such that there exist $(\kappa, \beta, \nu) \in (0, +\infty)^3$ satisfying:*

$$\forall (\varepsilon, u) \in (0, +\infty) \times \mathbb{R}, \quad \psi_\varepsilon^{-1}(u) \leq \kappa u^\beta \varepsilon^{-\nu}.$$

Then for any distribution P with low density exponent γ, there exist constant $(K, r) \in (0, +\infty)$ such that for any $f \in \mathcal{M}$ with an excess regularized ϕ-risk upper bounded by r the following holds:

$$R(f) - R^* \leq K\left(R_{\phi,0}(f) - R^*_{\phi,0}\right)^{\frac{\beta\gamma}{\gamma+\nu}}.$$

Proof Let $(c_2, \varepsilon_0) \in (0, +\infty)^2$ such that

$$\forall \varepsilon \in [0, \varepsilon_0], \quad P(A_\varepsilon) \leq c_2 \varepsilon^\gamma, \tag{58}$$

and define

$$r := \varepsilon_0^{\frac{\gamma+\nu}{\beta}} \kappa^{-\frac{1}{\beta}} c_2^{\frac{1}{\beta}}. \tag{59}$$

Given any function $f \in \mathcal{M}$ such that $\delta = R_{\phi,0}(f) - R^*_{\phi,0} \leq r$, let

$$\varepsilon := \kappa^{\frac{1}{\gamma+\nu}} c_2^{-\frac{1}{\gamma+\nu}} \delta^{\frac{\beta}{\gamma+\nu}}. \tag{60}$$

Because $\delta \leq r$, we can upper bound ε by:

$$\begin{aligned} \varepsilon &\leq \kappa^{\frac{1}{\gamma+\nu}} c_2^{-\frac{1}{\gamma+\nu}} r^{\frac{\beta}{\gamma+\nu}} \\ &= \varepsilon_0. \end{aligned} \tag{61}$$

4. We note that after this work was submitted, a related analysis has been proposed in Steinwart (2005b). The latter provides a very general framework, which in particular allows to derive Theorem 4 without the use of Theorem 28.

Combining (61) and (58), we obtain:

$$\begin{aligned} P(A_\varepsilon) &\leq c_2 \varepsilon^\gamma \\ &\leq \kappa^{\frac{\gamma}{\gamma+\nu}} c_2^{\frac{\nu}{\gamma+\nu}} \delta^{\frac{\beta\gamma}{\gamma+\nu}}. \end{aligned} \tag{62}$$

On the other hand,

$$\begin{aligned} \psi_\varepsilon^{-1}(\delta) &\leq \kappa \delta^\beta \varepsilon^{-\nu} \\ &= \kappa^{\frac{\gamma}{\gamma+\nu}} c_2^{\frac{\nu}{\gamma+\nu}} \delta^{\frac{\beta\gamma}{\gamma+\nu}}. \end{aligned} \tag{63}$$

Combining Theorem 28 with (62) and (63) leads to the result claimed with the constant r defined in (59) and

$$K := 2\kappa^{\frac{\gamma}{\gamma+\nu}} c_2^{\frac{\nu}{\gamma+\nu}}.$$

■

7. Consistency of SVMs

In this section we illustrate the results obtained throughout Section 6 for a general loss function ϕ, in particular the control of the excess R-risk by the excess $R_{\phi,0}$-risk of Theorem 29, to the specific cases of the loss functions used in 1- and 2-SVM. This leads to the proof of Theorem 6 in Section 7.3.

7.1 The Case of 1-SVM

Let $\phi(\alpha) = \max(1-\alpha, 0)$. Then we easily obtain, for any $(\eta, \rho) \in [-1,1] \times (0, +\infty)$:

$$C_{\eta,\rho}(\alpha) = \begin{cases} \eta(1-\alpha) + \lambda\alpha^2/\rho & \text{if } \alpha \in (-\infty, -1] \\ \eta(1-\alpha) + (1-\eta)(1+\alpha) + \lambda\alpha^2/\rho & \text{if } \alpha \in [-1, 1] \\ (1-\eta)(1+\alpha) + \lambda\alpha^2/\rho & \text{if } \alpha \in [1, +\infty). \end{cases}$$

This shows that $C_{\eta,\rho}$ is strictly decreasing on $(-\infty, -1]$ and strictly increasing on $[1, +\infty)$; as a result it reaches its minimum on $[-1, 1]$. Its derivative on this interval is equal to:

$$\forall \alpha \in (-1, 1), \quad C'_{\eta,\rho}(\alpha) = \frac{2\lambda\alpha}{\rho} + 1 - 2\eta.$$

This shows that $C_{\eta,\rho}$ reaches its minimum at the point:

$$\alpha(\eta, \rho) = \begin{cases} -1 & \text{if } \eta \leq 1/2 - \lambda/\rho \\ (\eta - 1/2)\rho/\lambda & \text{if } \eta \in [1/2 - \lambda/\rho, 1/2 + \lambda/\rho] \\ 1 & \text{if } \eta \geq 1/2 + \lambda/\rho \end{cases} \tag{64}$$

and that the value of this minimum is equal to:

$$M(\eta, \rho) = \begin{cases} 2\eta + \lambda/\rho & \text{if } \eta \leq 1/2 - \lambda/\rho \\ 1 - \rho(\eta - 1/2)^2/\lambda & \text{if } \eta \in [1/2 - \lambda/\rho, 1/2 + \lambda/\rho] \\ 2(1-\eta) + \lambda/\rho & \text{if } \eta \geq 1/2 + \lambda/\rho \end{cases}$$

From this we deduce that for all $(\rho,\theta) \in (0,+\infty) \times [-1,1]$:

$$\psi_\rho(\theta) = \begin{cases} \rho\theta^2/(4\lambda) & \text{if } 0 \leq \theta \leq 2\lambda/\rho, \\ \theta - \lambda/\rho & \text{if } 2\lambda/\rho \leq \theta \leq 1 \end{cases}$$

whose inverse function is

$$\psi_\rho^{-1}(u) = \begin{cases} \sqrt{4\lambda u/\rho} & \text{if } 0 \leq u \leq \lambda/\rho, \\ u + \lambda/\rho & \text{if } u \geq \lambda/\rho. \end{cases} \tag{65}$$

7.2 The Case of 2-SVM

Let $\phi(\alpha) = \max(1-\alpha, 0)^2$. Then we obtain, for any $(\eta,\rho) \in [-1,1] \times (0,+\infty)$:

$$C_{\eta,\rho}(\alpha) = \begin{cases} \eta(1-\alpha)^2 + \lambda\alpha^2/\rho & \text{if } \alpha \in (-\infty,-1] \\ \eta(1-\alpha)^2 + (1-\eta)(1+\alpha)^2 + \lambda\alpha^2/\rho & \text{if } \alpha \in [-1,1] \\ (1-\eta)(1+\alpha)^2 + \lambda\alpha^2/\rho & \text{if } \alpha \in [1,+\infty). \end{cases}$$

This shows that $C_{\eta,\rho}$ is strictly decreasing on $(-\infty,-1]$ and strictly increasing on $[1,+\infty)$; as a result it reaches its minimum on $[-1,1]$. Its derivative on this interval is equal to:

$$\forall \alpha \in (-1,1), \quad C'_{\eta,\rho}(\alpha) = 2\left(1 + \frac{\lambda}{\rho}\right)\alpha + 1 - 2\eta.$$

This shows that $C_{\eta,\rho}$ reaches its minimum at the point:

$$\alpha(\eta,\rho) = (2\eta - 1)\frac{\rho}{\lambda+\rho}. \tag{66}$$

and that the value of this minimum is equal to:

$$M(\eta,\rho) = 1 - (2\eta-1)^2\frac{\rho}{\lambda+\rho}.$$

From this we deduce that for all $(\rho,\theta) \in (0,+\infty) \times [-1,1]$:

$$\psi_\rho(\theta) = \frac{\rho}{\lambda+\rho}\theta^2$$

whose inverse function is

$$\psi_\rho^{-1}(u) = \sqrt{\left(1 + \frac{\lambda}{\rho}\right)u}. \tag{67}$$

Remark 30 *The minimum of $C_{\eta,\rho}$ being reached on $(-1,1)$ for any $(\eta,\rho) \in [0,1] \times (0,+\infty)$, the result would be identical for any convex loss function ϕ_0 that is equal to $(1-\alpha)^2$ on $(-\infty,1)$. Indeed, the corresponding regularized generic conditional ϕ_0-risk would coincide with $C_{\eta,\rho}$ on $(-1,1)$ and would be no smaller than $C_{\eta,\rho}$ outside of this interval; it would therefore have the same minimal value reached at the same point, and consequently the same function M and ψ. This is for example the case with the loss function used in LS-SVM, $\phi_0(\alpha) = (1-\alpha)^2$.*

7.3 Proof of Theorem 6

Starting with $\phi_1(\alpha) = \max(1-\alpha, 0)$, let us follow the proof of Theorem 29 by taking $\beta = \nu = 1/2$ and $\kappa = 2\sqrt{\lambda}$. For r defined as in (59), let us choose

$$r_1 = \min\left(r, \left(\frac{c_2 \lambda^{\gamma+\nu}}{\kappa 2^{\frac{\gamma+\nu}{\beta}}}\right)^{\frac{1}{\beta+\gamma+\nu}}\right).$$

For a function $f \in \mathcal{M}$, choosing ε as in (60), $\delta \leq r_1$ implies

$$\begin{aligned}\delta &\leq \left(\frac{c_2 \lambda^{\gamma+\nu}}{\kappa 2^{\frac{\gamma+\nu}{\beta}}}\right)^{\frac{1}{\beta+\gamma+\nu}} \\ &= \left(\varepsilon^{-(\gamma+\nu)} 2^{-\frac{\gamma+\nu}{\beta}} \lambda^{\gamma+\nu} \delta^{\beta}\right)^{\frac{1}{\beta+\gamma+\nu}}\end{aligned}$$

and therefore:

$$\delta 2^{-\frac{1}{\beta}} \leq \frac{\lambda}{\varepsilon}.$$

This ensures by (65) that for $u = \delta 2^{-\frac{1}{\beta}}$, one indeed has

$$\psi_\rho^{-1}(u) = \kappa u^{\beta} \varepsilon^{-\nu},$$

which allows the rest of the proof, in particular (59), to be valid. This proves the result for ϕ_1, with

$$K_1 = 2 \times 2^{\frac{2\gamma}{2\gamma+1}} \lambda^{\frac{\gamma}{2\gamma+1}} c_2^{\frac{1}{2\gamma+1}}.$$

For $\phi_2(\alpha) = \max(1-\alpha, 0)^2$ we can observe from (67) that, for any $\varepsilon \in (0, \varepsilon_0]$,

$$\psi_\varepsilon^{-1}(u) \leq \sqrt{(\lambda + \varepsilon_0)\frac{u}{\rho}}.$$

and the proof of Theorem 29 leads to the claimed result with $r_2 = r$ defined in (59), and

$$K_2 = 2 \times (\lambda + \varepsilon_0)^{\frac{\gamma}{2\gamma+1}} c_2^{\frac{1}{2\gamma+1}}.$$

■

Remark 31 *We note here that ε can be chosen as small as possible in order to move the constant K_2 as close as possible to its lower bound:*

$$\bar{K_2} = 2 \times \lambda^{\frac{\gamma}{2\gamma+1}} c_2^{\frac{1}{2\gamma+1}}.$$

but the counterpart of decreasing K_2 is to decrease r_2 too, by (59). We also notice the constant corresponding to the 1-SVM loss function is larger than that of the 2-SVM loss function, by a factor of up to $2^{\frac{2\gamma}{2\gamma+1}}$

8. Consistency of One-class SVMs for Density Level Set Estimation

In this section we focus on the one-class case: η is identically equal to 1, and P is just considered as a distribution on $\mathbb{R}^d$, with density ρ with respect to the Lebesgue measure. The first subsection is devoted to the proof of Theorem 8, and the second subsection to the proof of Theorem 9.

8.1 Proof of Theorem 8

Theorem 8 easily follows from combining some results given in this paper. First, it follows from (64) that, in the one-class case where $\eta = 1$ on its domain, the asymptotic function $f_{\phi,0}$ equals the truncated density ρ_λ. Then, using Lemma 7, we get the following bound:

$$\| \hat{f}_\sigma - \rho_\lambda \|_{L_2}^2 \leq \frac{1}{\lambda} \left(R_{\phi,0}(\hat{f}_\sigma) - R_{\phi,0}^* \right).$$

Finally, under the assumption $\lim_{\delta \to 0} \omega(\rho_\lambda, \delta) = 0$, using Theorem 3 with, for instance, $p = 1$, $\delta = 1$, $\sigma = (1/n)^{1/4d}$, $\sigma_1 = (1/n)^{1/2d}$, and $x = \log(n)$, we deduce that for any $\varepsilon > 0$,

$$P\left\{ \| \hat{f}_\sigma - \rho_\lambda \|_{L_2} \geq \varepsilon \right\} \to 0$$

as $n \to \infty$.

■

8.2 Proof of Theorem 9

Theorem 9 directly follows from combining Theorem 8 with Theorem 33, which is stated and proved at the end of this section. To prove Theorem 33, it will be useful to first state Lemma 32.

Before going to this point, let us recall some specific notation in the context of density level set estimation. The aim is to estimate a density level set of level μ, for some $\mu > 0$:

$$C_\mu := \left\{ x \in \mathbb{R}^d \ : \ \rho(x) \geq \mu \right\}. \tag{68}$$

The estimator that is considered here is the plug-in density level set estimator associated with $\hat{f}_\sigma$, denoted by $\widehat{C}_\mu$:

$$\widehat{C}_\mu := \left\{ x \in \mathbb{R}^d \ : \ 2\lambda \hat{f}_\sigma(x) \geq \mu \right\}. \tag{69}$$

Recall that the asymptotic behaviour of $\hat{f}_\sigma$ in the one-class case is given in Theorem 8: $\hat{f}_\sigma$ converge to ρ_λ, which is proportional to the density ρ truncated at level 2λ. Taking into account the behaviour of ρ_λ, we only consider the situation where $0 < \mu < 2\lambda < \sup_{\mathbb{R}^d}(\rho) = M$. The density ρ is still assumed to have a compact support $S \subset \mathcal{X}$. To assess the quality of $\widehat{C}_\mu$, we use the so-called *excess mass* functional, first introduced by Hartigan (1987), which is defined for any measurable subset C of $\mathbb{R}^d$ as follows:

$$H_P(C) := P(C) - \mu \mathrm{Leb}(C), \tag{70}$$

where Leb is the Lebesgue measure. Note that H_P is defined with respect to both P and μ, and that it is maximized by C_μ. Hence, the quality of an estimate $\widehat{C}$ depends here on how its excess mass is close to this of C_μ.

The following lemma relates the L_2 convergence of a density estimator to the consistency of the associated plug-in density level set estimator, with respect to the excess mass criterion:

Lemma 32 *Let P be a probability distribution on $\mathbb{R}^d$ with compact support $S \subset \mathcal{X}$. Assume that P is absolutely continuous with respect to the Lebesgue's measure, and let ρ denote its associated density function. Assume furthermore that ρ is bounded on S. Consider a non-negative density estimate $\hat{\rho}$ defined on $\mathbb{R}^d$. Then the following holds*

$$H_P(C_\mu) - H_P(\widehat{C}) \leq K_5 \| \hat{\rho} - \rho \|_{L_2} \,, \tag{71}$$

where $\widehat{C}$ is the level set of $\hat{\rho}$ at level μ, and

$$K_5 = \frac{2\sqrt{m\| \rho \|_{L_\infty} + \frac{1-m}{Leb(S)}}}{m\mu} \,.$$

Proof To prove the lemma, it is convenient to first build an artificial classification problem using the density function ρ and the desired level μ, then to relate the excess-risk involved in this classification problem to the excess-mass involved in the original one-class problem. Note that this technique has already been used in Steinwart et al. (2005). Let us consider the following joint distribution Q defined by its marginal density function

$$q(x) := \begin{cases} m\rho(x) + (1-m)\frac{1}{\mathrm{Leb}(S)} & \text{if } x \in S \,, \\ 0 & \text{otherwise} \,, \end{cases} \tag{72}$$

and by its regression function

$$\eta_0(x) := \frac{m\rho(x)}{m\rho(x) + (1-m)\frac{1}{\mathrm{Leb}(S)}} \,, \quad x \in S \,, \tag{73}$$

where m is chosen such that

$$\eta_0(x) = \frac{\rho(x)}{\rho(x) + \mu} \,, \tag{74}$$

that is

$$m := \frac{1}{1 + \mu \mathrm{Leb}(S)} \,. \tag{75}$$

In words, in the above artificial classification problem, the initial distribution P stands for the marginal distribution of the positive class, and the negative class is generated by the uniform distribution over the support of P. The mixture coefficient m is determined by the initially desired density level μ. The corresponding Bayes classifier, which is the plug-in rule associated with η_0, is denoted by h^*.

Furthermore let us define $\hat{\eta}_0 := \hat{\rho} / (\hat{\rho} + \mu)$, which stands for an estimate of η_0 in our artificial classification problem, and $\hat{h}$ as the plug-in classifier associated with $\hat{\eta}_0$: $\hat{h} := \mathrm{sign}\,(2\hat{\eta}_0 - 1)$. Then it is straightforward that h^* is the indicator function of C_μ, and that $\hat{h}$ is the indicator function of $\widehat{C}$. Moreover

$$R(\hat{h}) - R(h^*) = m \left(H_P(C^*) - H_P(\widehat{C}) \right) \,.$$

Indeed,

$$\begin{aligned} R(\hat{h}) &= Q(\hat{h}(X) \neq Y) \\ &= Q(Y=-1)Q(\hat{h}(X)=1|Y=-1) + Q(Y=1)Q(\hat{h}(X)=-1|Y=1) \\ &= (1-m)\frac{\mathrm{Leb}\left(\widehat{C}\right)}{\mathrm{Leb}\,(S)} + m(1 - P(\widehat{C})) \,, \end{aligned}$$

and, similarly,

$$R(h^*) = (1-m)\frac{\mathrm{Leb}\,(C_\mu)}{\mathrm{Leb}\,(S)} + m(1-P(C_\mu))\,, \tag{76}$$

which proves the claim.
Now, the following can be derived, starting from an equality that can be found in Devroye et al. (1996, page 16):

$$\begin{aligned}
R(\hat{h}) - R(h^*) &= 2\mathbb{E}_Q\left[\left|\eta_0 - \frac{1}{2}\right| 1_{\hat{h}\neq h^*}\right] \\
&\leq 2\mathbb{E}_Q\left[|\,\eta_0 - \hat{\eta}_0\,|^2\right]^{1/2} \\
&= 2\mu\left(\int_{\mathbb{R}^d}\left(\frac{|\hat{\rho}(x)-\rho(x)|}{(\hat{\rho}(x)+\mu)(\rho(x)+\mu)}\right)^2 q(x)dx\right)^{1/2} \\
&\leq 2\mu\sqrt{A}\left(\int_{\mathbb{R}^d}\left(\frac{|\hat{\rho}(x)-\rho(x)|}{(\hat{\rho}(x)+\mu)(\rho(x)+\mu)}\right)^2 dx\right)^{1/2} \\
&\leq 2\frac{\sqrt{A}}{\mu}\|\,\hat{\rho}-\rho\,\|_{L_2}\,,
\end{aligned}$$

where A is a positive uniform upper bound on $q(x)$, for instance $A = m\|\,\rho\,\|_{L_\infty} + (1-m)\,/\mathrm{Leb}(S)$. Combining the previous equality with the last inequality concludes the proof. ■

We could just directly apply this lemma to $\hat{f}_\sigma$, ρ_λ and the distribution P_λ defined[5] through ρ_λ, but this would prove the consistency of $\hat{f}_\sigma$ with respect to the excess mass H_{P_λ}, which is different from the criterion H_P of interest. The following lemma implies that the plug-in density level set estimator at level $0 < \mu < 2\lambda$ based on the one-class SVM estimator is indeed consistent with respect to the excess mass H_P.

Theorem 33 *Let $\hat{f}$ be a non-negative squared integrable function that estimates ρ_λ (as defined in Equation 7). Let $0 < \mu < 2\lambda$. Let $\widehat{C}$ denote the level set of $2\lambda\hat{f}$ at level μ. Then*

$$H_P(C_\mu) - H_P(\widehat{C}) \leq K_6\|\,\hat{f} - \rho_\lambda\,\|_{L_2} \tag{77}$$

where $K_6 > 0$ depends neither on σ, nor on n.

Proof Let us introduce the following estimator:

$$\tilde{\rho} := 2\lambda\hat{f} + \tilde{\rho}_\lambda\,, \tag{78}$$

where the function $\tilde{\rho}_\lambda$ is defined as follows:

$$\tilde{\rho}_\lambda := \begin{cases} \rho(x) - 2\lambda & \text{if } \rho(x) \geq 2\lambda\,, \\ 0 & \text{otherwise,} \end{cases} \tag{79}$$

5. Note that P_λ is not a probability distribution, since the function ρ_λ does not integrate to 1. Still, the excess-mass remains well-defined.

and let $\tilde{C}$ denote the level set of $\tilde{\rho}$ at level μ. It can be checked that $\tilde{\rho} - \rho = 2\lambda\left(\hat{f} - \rho_\lambda\right)$, implying that

$$\| \tilde{\rho} - \rho \|_{L_2} = 2\lambda \| \hat{f} - \rho_\lambda \|_{L_2} . \tag{80}$$

Hence, using Lemma 32, we have

$$H_P(C_\mu) - H_P\left(\tilde{C}\right) \leq K_5 \| \tilde{\rho} - \rho \|_{L_2} = 2\lambda K_5 \| \hat{f} - \rho_\lambda \|_{L_2} , \tag{81}$$

leading to

$$H_P(C_\mu) - H_P(\widehat{C}) \leq 2\lambda c \| \hat{f} - \rho_\lambda \|_{L_2} + \left| H_P(\widehat{C}) - H_P(\tilde{C}) \right| . \tag{82}$$

The last thing to do is to bound $\left| H_P(\widehat{C}) - H_P(\tilde{C}) \right|$. Since P has a bounded density w.r.t. the Lebesgue's measure,

$$\left| H_P(\widehat{C}) - H_P(\tilde{C}) \right| \leq (\mu + M) \mathrm{Leb}\left(\widehat{C} \Delta \tilde{C}\right) . \tag{83}$$

By construction of $\tilde{\rho}$, if $C_{2\lambda}$ denotes the level set of ρ at level 2λ, and $\overline{C_{2\lambda}}$ its complementary in $\mathbb{R}^d$, then we have $\widehat{C} \cap \overline{C_{2\lambda}} = \tilde{C} \cap \overline{C_{2\lambda}}$ and $2\lambda \hat{f} \geq \mu \implies \tilde{\rho} \geq \mu$. Hence

$$\begin{aligned}
\mathrm{Leb}\left(\widehat{C} \Delta \tilde{C}\right) &= \int_{C_{2\lambda}} 1_{\left\{2\lambda \hat{f} < \mu \wedge \tilde{\rho} \geq \mu\right\}} \\
&\leq \int_{C_{2\lambda}} 1_{\left\{2\lambda \hat{f} < \mu\right\}} \\
&\leq \int_{C_{2\lambda}} \frac{2\lambda - 2\lambda \hat{f}}{2\lambda - \mu} 1_{\left\{2\lambda \hat{f} < \mu\right\}} \\
&\leq \frac{1}{2\lambda - \mu} \left(\int_{C_{2\lambda}} \left(2\lambda \rho_\lambda - 2\lambda \hat{f}\right)^2 \right)^{1/2} \\
&\leq \frac{2\lambda}{2\lambda - \mu} \| \hat{f} - \rho_\lambda \|_{L_2} .
\end{aligned}$$

This concludes the proof. ■

Acknowledgments

The authors are grateful to Stéphane Boucheron, Pascal Massart and Ingo Steinwart for fruitful discussions and advices. JPV is supported by NHGRI NIH award R33 HG003070 and by the grant ACI NIM 2003-72 of the French Ministry for Research and New Technologies. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, 68:337 – 404, 1950.

P. I. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification and risk bounds. *J. Amer. Statist. Assoc.*, 101(473):138–156, 2006.

P. L. Bartlett and A. Tewari. Sparseness vs estimating conditional probabilities: Some asymptotic results. In *Lecture Notes in Computer Science*, volume 3120, pages 564–578. Springer, 2004.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *Ann. Stat.*, 33(4): 1497–1537, 2005.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

R. A. DeVore and G. G. Lorentz. *Constructive Approximation*. Springer Grundlehren der Mathematischen Wissenschaften. Springer Verlag, 1993.

L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer Series in Statistics. Springer, 2000.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Applications of Mathematics*. Springer, 1996.

G. B. Folland. *Fourier analysis and its applications*. The Wadsworth & Brooks/Cole Mathematics Series. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1992.

J. A. Hartigan. Estimation of a convex density contour in two dimensions. *J. Amer. Statist. Assoc.*, 82(397):267–270, 1987.

V. Koltchinskii. Localized Rademacher complexities. Manuscript, September 2003.

G. Lugosi and N. Vayatis. On the Bayes-risk consistency of regularized boosting methods. *Ann. Stat.*, 32:30–55, 2004.

E. Mammen and A. Tsybakov. Smooth discrimination analysis. *Ann. Stat.*, 27(6):1808–1829, 1999.

P. Massart. Some applications of concentration inequalities to statistics. *Ann. Fac. Sc. Toulouse*, IX (2):245–303, 2000.

M. T. Matache and V. Matache. Hilbert spaces induced by Toeplitz covariance kernels. In *Lecture Notes in Control and Information Sciences*, volume 280, pages 319–334. Springer, Jan 2002.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13:1443–1471, 2001.

B. W. Silverman. On the estimation of a probability density function by the maximum penalized likelihood method. *Ann. Stat.*, 10:795–810, 1982.

I. Steinwart. Support vector machines are universally consistent. *J. Complexity*, 18:768–791, 2002.

I. Steinwart. Sparseness of support vector machines. *J. Mach. Learn. Res.*, 4:1071–1105, 2003.

I. Steinwart. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Trans. Inform. Theory*, 51(1):128–142, 2005a.

I. Steinwart. How to compare loss functions and their risks. Technical report, Los Alamos National Laboratory, 2005b.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. Technical Report LA-UR 04-8796, Los Alamos National Laboratory, 2004.

I. Steinwart, D. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. Technical Report LA-UR 04-8274, Los Alamos National Laboratory, 2004.

I. Steinwart, D. Hush, and Scovel C. A classification framework for anomaly detection. *J. Mach. Learn. Res.*, 6:211–232, 2005.

A.N. Tikhonov and V.Y. Arsenin. *Solutions of ill-posed problems*. W.H. Winston, 1977.

A. B. Tsybakov. On nonparametric estimation of density level sets. *Ann. Stat.*, 25:948–969, June 1997.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Ann. Stat.*, 32:56–134, 2004.

Journal of Machine Learning Research 7 (2006) 855–876 Submitted 11/05; Published 5/06

Infinite-σ Limits For Tikhonov Regularization

Ross A. Lippert LIPPERT@MATH.MIT.EDU
Department of Mathematics
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139-4307, USA

Ryan M. Rifkin RRIFKIN@HONDA-RI.COM
Honda Research Institute USA, Inc.
145 Tremont Street
Boston, MA 02111, USA

Editor: Gabor Lugosi

Abstract

We consider the problem of Tikhonov regularization with a general convex loss function: this formalism includes support vector machines and regularized least squares. For a family of kernels that includes the Gaussian, parameterized by a "bandwidth" parameter σ, we characterize the limiting solution as $\sigma \to \infty$. In particular, we show that if we set the regularization parameter $\lambda = \tilde{\lambda}\sigma^{-2p}$, the regularization term of the Tikhonov problem tends to an indicator function on polynomials of degree $\lfloor p \rfloor$ (with residual regularization in the case where $p \in \mathbb{Z}$). The proof rests on two key ideas: *epi-convergence*, a notion of functional convergence under which limits of minimizers converge to minimizers of limits, and a *value-based formulation of learning*, where we work with regularization on the function output values (y) as opposed to the function expansion coefficients in the RKHS. Our result generalizes and unifies previous results in this area.

Keywords: Tikhonov regularization, Gaussian kernel, theory, kernel machines

1. Introduction

Given a data set $(x_1, \hat{y}_1), \ldots, (x_n, \hat{y}_n) \in \mathbb{R}^d \times \mathbb{R}$, the supervised learning task is to construct a function $f(x)$ that, given a new point, x, will predict the associated y value. A number of methods for this problem have been studied. One popular family of techniques is Tikhonov regularization in a Reproducing Kernel Hilbert Space (RKHS) (Evgeniou et al., 2000):

$$\inf_{f \in \mathcal{H}} \left\{ n\lambda ||f||_\kappa^2 + \sum_{i=1}^{n} v(f(x_i), \hat{y}_i) \right\}.$$

Here, $v : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a *loss function* indicating the price we pay when we see x_i, predict $f(x_i)$, and the true value is $\hat{y}_i$. The squared norm, $||f||_\kappa^2$, in the RKHS $\mathcal{H}$ involves the kernel function κ (Aronszajn, 1950). The regularization constant, $\lambda > 0$, controls the trade-off between fitting the training set accurately (minimizing the penalties) and forcing f to be smooth in $\mathcal{H}$. The Representer Theorem (Wahba, 1990; Girosi et al., 1995; Schölkopf et al., 2001) guarantees that the solution to

the Tikhonov regularization can be written in the form

$$f(x) = \sum_{i=1}^{n} c_i \kappa(x_i, x).$$

In practice, solving a Tikhonov regularization problem is equivalent to finding the expansion coefficients c_i.

One popular choice for κ is the *Gaussian* kernel $\kappa_\sigma(x,x') = e^{-\frac{||x-x'||^2}{2\sigma^2}}$, where σ is the bandwidth of the Gaussian. Common choices for v include the *square* loss, $v(y,\hat{y}) = (y-\hat{y})^2$, and the *hinge* loss, $v(y,\hat{y}) = \max\{0, 1-y\hat{y}\}$, which lead to regularized least squares and support vector machines, respectively.

Our work was originally motivated by the empirical observation that on a range of tasks, regularized least squares achieved very good performance with very large σ. (For example, we could often choose σ so large that every kernel product between pairs of training points was between .99999 and 1.) To get good results with large σ, it was necessary to make λ small. We decided to study this relationship.

Regularized least squares (RLS) is an especially simple Tikhonov regularization algorithm: "training" RLS simply involves solving a system of linear equations. In particular, defining the matrix K via $K_{ij} = \kappa(x_i, x_j)$, the RLS expansion coefficients c are given by $(K + n\lambda I)c = \hat{y}$, or $c = (K + n\lambda I)^{-1}\hat{y}$. Given a test point x_0, we define the n-vector k via $k_i = \kappa(x_0, x_i)$, and we have, for RLS with a fixed bandwidth,

$$f(x_0) = k^t (K + n\lambda I)^{-1} \hat{y}.$$

In Lippert and Rifkin (2006), we studied the limit of this expression as $\sigma \to \infty$, showing that if we set $\lambda = \tilde{\lambda}\sigma^{-2p-1}$ for p a positive integer, the infinite-σ limit converges (pointwise) to the degree p polynomial with minimal empirical risk on the training set. The asymptotic predictions are equivalent to those we would get if we simply fit an (unregularized) degree p polynomial to our training data.

In Keerthi and Lin (2003), a similar phenomenon was also noticed for support vector machines (SVM) with Gaussian kernels, where it was observed that the SVM function could be made to converge (in the infinite-σ limit) to a linear function which minimized the hinge loss plus a residual regularization (discussed further below). In that work, only a linear result was obtained; no results were given for general polynomial approximation limits.

In the current work, we unify and generalize these results, showing that the occurrence of these polynomial approximation limits is a general phenomenon, which holds across all convex loss functions and a wide variety of kernels taking the form $\kappa_\sigma(x,x') = \kappa(x/\sigma, x'/\sigma)$. Our main result is that for a convex loss function and a valid kernel, if we take $\sigma \to \infty$ and $\lambda = \tilde{\lambda}\sigma^{-2p}$, the regularization term of the Tikhonov problem tends to an indicator function on polynomials of degree $\lfloor p \rfloor$. In the case where $p \in \mathbb{Z}$, there is residual regularization on the degree-p coefficients of the limiting polynomial.

Our proof relies on two key ideas. The first is the notion of *epi-convergence*, a functional convergence under which limits of minimizers converge to minimizers of limits. This notion allows us to characterize the limiting Tikhonov regularization problem in a mathematically precise way. The second notion is a *value-based formulation of learning*. The idea is that instead of working with the expansion coefficients in the RKHS (c_i), we can write the regularization problem directly in terms of the predicted values (y_i). This allows us to avoid combining and canceling terms whose limits are individually undefined.

2. Notation

In this section, we describe the notation we use throughout the paper. Some of our choices are non-standard, and we try to indicate these.

2.1 Data Sets and Regularization

We refer to a general d dimensional vector with the symbol x (plus superscripts or subscripts). We assume a fixed set of n training points $x_i \in \mathbb{R}^d$ $(1 \leq i \leq n)$ and refer to the totality of these data points by X, such that for any $f : \mathbb{R}^d \to \mathbb{R}$, $f(X) = (\, f(x_1) \quad \cdots \quad f(x_n)\,)^t$ is the vector of values over the points and for any $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $g(X,X)$ is the matrix of values over pairs of points, i.e. $[g(X,X)]_{ij} = g(x_i,x_j)$. We let x_0 represent an arbitrary "test point" not in the training set. We assume we are given "true" $\hat{y}$ values at the training points: $\hat{y}_i \in \mathbb{R}, 1 \leq i \leq n$. While it is more common to use $\hat{y}$ to refer to the "predicted" output values and y to refer to the "true" output values, we find this choice much more notationally convenient, because our value-based formulation of learning (3) requires us to work with the predicted values very frequently.

Tikhonov regularization is given by

$$\inf_{f \in \mathcal{H}} \left\{ n\lambda ||f||_\kappa^2 + \sum_{i=1}^{n} v(f(x_i), \hat{y}_i) \right\}. \tag{1}$$

Tikhonov regularization can be used for both classification and regression tasks, but we refer to the function f as the *regularized solution* in all cases. We call the left-hand portion the *regularization term*, and the right-hand portion the *loss term*. We assume a loss function $v(y,\hat{y})$ that is convex in its first argument and minimized at $y = \hat{y}$ (thereby ruling out, for example, the 0/1 "misclassification rate"). We call such a loss function *valid*. Aside from convexity, we will be unconcerned with the form of the loss function and often take the loss term in the optimization in (1) to be some convex function $V : \mathbb{R}^n \to \mathbb{R}$ which is minimized by the vector $\hat{y}$ of $\hat{y}_i$'s.

To avoid confusion, when subscripting over d dimensional indices, we use letters from the beginning of the alphabet $(a,b,c,\ldots)$, while using letters from the middle $(i,j,\ldots)$ for n dimensional indices.

When referring to optimization problems, we use an over-dot (e.g. $\dot{y}$) to denote optimizing quantities. We are not time-differentiating anything in this work, so this should not cause confusion.

2.2 Polynomials

By a *multi-index* we refer to $I \in \mathbb{Z}^d$ such that $I_a \geq 0$. Given $x \in \mathbb{R}^d$ we write $x^I = \prod_{a=1}^d x_a^{I_a}$. We also write X^I to denote $(\, x_1^I \quad \cdots \quad x_n^I\,)^t$. The *degree* of a multi-index is $|I| = \sum_{a=1}^d I_a$. We use the "choose" notation

$$\binom{|I|}{I} = \frac{|I|!}{\prod_{a=1}^d I_a!}.$$

Let $\{I_i\}_{i=0}^\infty$ be an ordering of multi-indices which is non-decreasing by degree (in particular, $I_0 = (\,0 \quad \cdots \quad 0\,)$). We consider this fixed for the remainder of the work. Define $d_c = |\{I : |I| \leq c\}|$ and note that $d_c = \binom{d+c+1}{c}$. Put differently, $\{I : |I| = c\} = \{I_i : d_{c-1} \leq i < d_c\}$.

Given a data set, while the monomials x^{I_i} are linearly independent as functions, no more than n of the X^{I_i} can be linearly independent. We say that a data set is *generic* if the X^{I_i}, for $i < n$, are

linearly independent. This is equivalent to requiring that the data not reside on the zero-set of a low degree system polynomials. This is not an unreasonable assumption for data which is presumed to have been generated by distributions supported on $\mathbb{R}^n$ or some sphere in $\mathbb{R}^n$. Throughout this paper, we assume that our data is generic. It is possible to carry out the subsequent derivations without it, but the modifications which result are tedious. In particular, parts of Theorem 12, which treat the first n monomials X^{I_i} as linearly independent (e.g. assuming $v_\alpha(X)$ is non-singular in the proof), would need to be replaced with analogous statements about the first n monomials $X^{I_{i_j}}$ which are linearly independent, and various power-series expansion coefficients would have to be adjusted accordingly. Additionally, our main result requires not only genericity of the data, but also that $n > d_p$ where p is the degree of the asymptotic regularized solution.

2.3 Kernels

It is convenient to use infinite matrices and vectors to express certain infinite series. Where used, the convergence of the underlying series implies the convergence of any infinite sums that arise from the matrix products we form, and we will not attempt to define any inverses of non-diagonal infinite matrices. This is merely a notational device to avoid excessive explicit indexing and summing in the formulas ahead. Additionally, since many of our vectors come from power series expansions, we adopt the convention of indexing such vectors and matrices starting from 0.

A Reproducing Kernel Hilbert Space (RKHS) is characterized by a kernel function κ. If κ has a power series expansion, we may write

$$\begin{aligned}\kappa(x,x') &= \sum_{i,j\geq 0} M_{ij} x^{I_i} x'^{I_j} \\ &= v(x) M v(x')^t\end{aligned}$$

where $M \in \mathbb{R}^{\infty\times\infty}$ is an infinite matrix and $v(x) = \begin{pmatrix}1 & x^{I_1} & x^{I_2} & \cdots\end{pmatrix} \in \mathbb{R}^{1\times\infty}$ is an infinite row-vector valued function of x. We emphasize that M is an infinite matrix induced by the kernel function κ and the ordering of multi-indices; it has nothing to do with our data set.

We say that a kernel is *valid* if every finite upper-left submatrix of M is symmetric and positive definite; in this case, we also say that the infinite matrix M is symmetric positive definite. This condition is the one we use in our main proof; however, it can be difficult to check. It is independent of the Mercer property (which states that the kernel matrix $\kappa(X,X)$ for a set X is positive semidefinite), since $\kappa(x,x') = \frac{1}{1-xx'}$ is valid but not Mercer, and $\exp(-(x^3-x'^3)^2)$ is Mercer but not valid. This notion is, basically, that the feature space of κ can approximate any polynomial function near the origin to arbitrary accuracy. We are not aware of any mention of this property in the literature. The following lemma gives a stronger condition that implies validity.

Lemma 1 *If $\kappa(x,x') = \sum_{c\geq 0}(x\cdot x')^c g_c(x) g_c(x')$ for some analytic functions $g_c(x)$ such that $g_c(0) \neq 0$, then κ is a valid kernel.*

Proof By the multinomial theorem,

$$(x\cdot x')^c = \left(\sum_{i=1}^d x_i x'_i\right)^c = \sum_{|I|=c} \binom{|I|}{I} x^I x'^I.$$

Let $g_c(x) = \sum_I G_{cI} x^I$, and note $g_c(0) = G_{c0}$, thus

$$\begin{aligned}
\kappa(x,x') &= \sum_{I,J,c\geq 0} \sum_{|E|=c} \binom{|E|}{E} x^{I+E} G_{cI} G_{cJ} x'^{J+E} \\
&= \sum_{I,J,E} \binom{|I_k|}{I_k} x^{I+E} G_{|E|I} G_{|E|J} x'^{J+E} \\
&= \sum_{I,J\geq E} x^I G_{|E|(I-E)} \binom{|E|}{E} G_{|E|(J-E)} x'^J \\
&= v(x) L L^t v(x')
\end{aligned}$$

where $L_{ij} = \binom{|I_j|}{I_j}^{\frac{1}{2}} G_{|I_j|(I_i-I_j)}$ when $I_i \geq I_j$ and 0 otherwise. In other words, L is an infinite lower triangular matrix with non-vanishing diagonal elements (since $G_{c0} \neq 0$ for all c).

Since $M = LL^t$, the upper-left submatrices of L are the Cholesky factors of the corresponding upper-left submatrix of M, and thus M is positive definite. ■

We note that $\kappa(x,x') = \exp(-\frac{1}{2}||x-x'||^2)$ can be written in the form of Lemma 1:

$$\begin{aligned}
\kappa(x,x') &= \exp\left(-\frac{1}{2}||x||^2\right) \exp(x\cdot x') \exp\left(-\frac{1}{2}||x'||^2\right) \\
&= \sum_{c=0}^{\infty} \frac{(x\cdot x')^c}{c!} \exp\left(-\frac{1}{2}||x||^2\right) \exp\left(-\frac{1}{2}||x'||^2\right) \\
&= \sum_{c=0}^{\infty} (x\cdot x')^c g_c(x) g_c(x'),
\end{aligned}$$

where $g_c(x) = \frac{1}{\sqrt{c!}} \exp(-\frac{1}{2}||x||^2)$.

We will consider kernel functions κ_σ which are parametrized by a bandwidth parameter σ (or $s = \frac{1}{\sigma}$). We will occasionally use K_σ to refer to the matrix whose i,jth entry $\kappa_\sigma(x_i,x_j)$, for $1 \leq i \leq n, 1 \leq j \leq n$. We will also use k_σ to denote the n-vector whose ith entry is $\kappa_\sigma(x_i,x_0)$ — the kernel product between the ith training point x_i and the test point x_0.

3. Value-Based Learning Formulation

In this section, we discuss our value-based learning formulation. Using the representer theorem and basic facts about RKHS, the standard Tikhonov regularization problem (1) can be written in terms of the expansion coefficients c and the kernel matrix K:

$$\inf_{c\in\mathbb{R}^n} \left\{ n\lambda c^t K c + \sum_{i=1}^n v([Kc]_i, \hat{y}_i) \right\}. \tag{2}$$

The predicted values on the training set are $y = f(X) = Kc$. If the kernel matrix K is invertible (which is the case for a Gaussian kernel and a generic data set), then $c = K^{-1}y$, and we can rewrite the minimization as

$$\inf_{y\in\mathbb{R}^n} \left\{ n\lambda y^t K^{-1} y + V(y) \right\}. \tag{3}$$

where V is convex ($V(y) = \sum_{i=1}^{n} v_i(y_i, \hat{y}_i)$).

While problem 2 is explicit in the coefficients of the expansion of the regularized solution, problem 3 is explicit in the predicted values y_i. The purpose behind our choice of formulation is to avoid the unnecessary complexities which result from replacing κ with κ_σ and taking limits as both c_i and $\kappa_\sigma(x, x_i)$ change separately with σ: note that in problem 3, *only the regularization term is varying with* σ.

In this section, we will show how our formulation achieves this, by allowing us to state a single optimization problem which simultaneously solves the Tikhonov regularization problem on the training data and evaluates the resulting function on the test data.

Theorem 2 *Let* $y = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \in \mathbb{R}^{m+n}$ *be a block vector with* $y_0 \in \mathbb{R}^m, y_1 \in \mathbb{R}^n$ *and* $K = \begin{pmatrix} K_{00} & K_{01} \\ K_{10} & K_{11} \end{pmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$ *be a positive definite matrix. For any* $V : \mathbb{R}^n \to \mathbb{R}$*, if* $\dot{y}$ *minimizes*

$$y^t K^{-1} y + V(y_1) \tag{4}$$

then $\dot{y}_1$ *minimizes*

$$y_1^t K_{11}^{-1} y_1 + V(y_1) \tag{5}$$

and $\dot{y}_0 = K_{01} K_{11}^{-1} \dot{y}_1$.

Proof

$$\inf_{y_0, y_1} \{y^t K^{-1} y + V(y_1)\} = \inf_{y_1} \{\inf_{y_0} \{y^t K^{-1} y\} + V(y_1)\}. \tag{6}$$

Let $K^{-1} = \bar{K} = \begin{pmatrix} \bar{K}_{00} & \bar{K}_{01} \\ \bar{K}_{10} & \bar{K}_{11} \end{pmatrix}$. Consider minimizing $y^t K^{-1} y = y_0^t \bar{K}_{00} y_0 + 2 y_0^t \bar{K}_{01} y_1 + y_1^t \bar{K}_{11} y_1$. For fixed y_1, $\dot{y}_0 = -\bar{K}_{00}^{-1} \bar{K}_{01} y_1 = K_{01} K_{11}^{-1} y_1$ by (17) of Lemma 15. Thus,

$$\inf_{y_0} \left\{ y^t K^{-1} y \right\} = y_1^t (\bar{K}_{11} - \bar{K}_{10} \bar{K}_{00}^{-1} \bar{K}_{01}) y_1 = y_1^t K_{11}^{-1} y_1$$

by (19) of Lemma 15. ■

We contextualize this result in terms of the Tikhonov learning problem with the following corollary.

Corollary 3 *Let* X *be a given set of data points* $x_1, \ldots, x_n$ *with* x_0 *a test point. Let* κ *be a valid kernel function and* $V : \mathbb{R}^n \to \mathbb{R}$ *be arbitrary. If* $\dot{y} = (\dot{y}_0 \quad \dot{y}_1 \quad \cdots \quad \dot{y}_n)^t$ *is the minimizer of*

$$n\lambda y^t \kappa \left(\begin{pmatrix} x_0 \\ X \end{pmatrix}, \begin{pmatrix} x_0 \\ X \end{pmatrix} \right)^{-1} y + V(y_1, \ldots, y_n)$$

then $(\dot{y}_1 \quad \cdots \quad \dot{y}_n)^t$ *minimizes* $n\lambda y^t \kappa(X, X)^{-1} y + V(y)$ *and*
$\dot{y}_0 = \sum_{i=1}^{n} c_i \kappa(x_0, x_i)$*, for* $c = \kappa(X, X)^{-1} (\dot{y}_1 \quad \cdots \quad \dot{y}_n)^t$.

Thus, when solving for y instead of c, we can evaluate the function at a test point x_0 by including the additional point in a larger minimization problem where the test point contributes to the regularization, but not the loss. When taking limits, we are going to work directly with the y_i, and we are going to avoid dealing with the (divergent) limits of the c_i.

4. Epi-limits, Convex Functions, and Quadratic Forms

The relationship between the limit of a function and the limit of its minimizer(s) is subtle, and it is very easy to make incorrect statements. For convex functions there are substantial results on this subject, which we review; we essentially follow the development of Rockafellar and Wets (2004, chap. 7). Since the component of our objective which depends on the limiting parameter is a quadratic form, we will eventually specialize the results to quadratic forms.

Definition 4 (epigraphs) *Given a function $f : \mathbb{R}^n \to (-\infty, \infty]$, its* epigraph, *epi f is the subset of $\mathbb{R} \times \mathbb{R}^n$ given by*

$$epi\ f = \{(z,x) : z \geq f(x)\}.$$

We call f closed, convex, or proper if those statements are true of epi f (proper referring to epi f being neither $\emptyset$ nor $\mathbb{R}^{n+1}$).

The functions we will be interested in are closed, convex, and proper. We will therefore adopt the abbreviation *ccp* for these conditions. Additionally, since we will be studying parameterized functions, f_s, for $0 < s$ as $s \to 0$, we say that such a family of functions is *eventually* convex (or closed, or proper) when there exists some $s_0 > 0$ such that f_s is convex (or closed, or proper) for all $0 < s < s_0$.

We review the definition of lim inf and lim sup for functions of a single variable. Given $h : (0,\infty) \to (-\infty,\infty]$, it is clear that the functions $\inf_{s' \in (0,s)}\{h(s')\}$ and $\sup_{s' \in (0,s)}\{h(s')\}$ are non-increasing and non-decreasing functions of (increasing) s respectively.

Definition 5 *For $h : (0,\infty) \to (-\infty,\infty]$,*

$$\begin{aligned} \liminf_{s \to 0} h(s) &= \sup_{s>0} \left\{ \inf_{s' \in (0,s)} \{h(s')\} \right\} \\ \limsup_{s \to 0} h(s) &= \inf_{s>0} \left\{ \sup_{s' \in (0,s)} \{h(s')\} \right\}. \end{aligned}$$

As defined, either of the limits may take the value ∞. A useful alternate characterization, which is immediate from the definition, is $\liminf_{s\to 0} h(s) = h_0$ iff $\forall \varepsilon > 0, \exists s_0, \forall s \in (0,s_0) : h(s) \geq h_0 - \varepsilon$, and $\limsup_{s\to 0} h(s) = h_0$ iff $\forall \varepsilon > 0, \exists s_0, \forall s \in (0,s_0) : h(s) \leq h_0 + \varepsilon$, where either inequality can be strict if $h_0 < \infty$.

Definition 6 (epi-limits) *We say $\lim_{s\to 0} f_s = f$ if for all $x_0 \in \mathbb{R}^n$, both the following properties hold:*
Property 1: *$\forall x : [0,\infty) \to \mathbb{R}^n$ continuous at $x(0) = x_0$ satisfies*

$$\liminf_{s \to 0} f_s(x(s)) \geq f(x_0) \tag{7}$$

Property 2: *$\exists x : [0,\infty) \to \mathbb{R}^n$ continuous at $x(0) = x_0$ satisfying*

$$\limsup_{s \to 0} f_s(x(s)) \leq f(x_0). \tag{8}$$

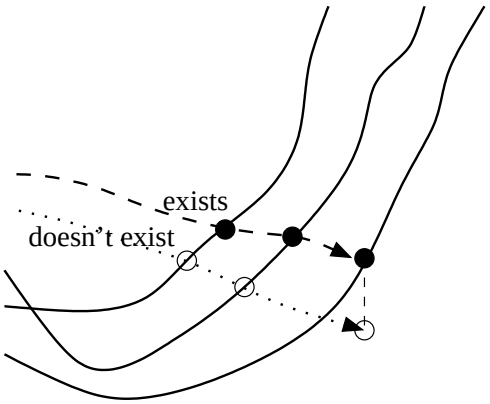


Figure 1: (property 1) of Definition 6 says that paths of points within epi f_s cannot end up below epi f, while (property 2) says that at least one such path hits every point of epi f.

This notion of functional limit is called an epigraphical limit (or epi-limit). Less formally, (property 1) is the condition that paths of the form $(x(s), f_s(x(s)))$ are, asymptotically, inside epi f, while (property 2) asserts the existence of a path which hits the boundary of epi f, as depicted in figure 1. Considering (property 1) with the function $x(s) = x_0$, it is clear that the epigraphical limit minorizes the pointwise limit (assuming both exist), but the two need not coincide. An example of this distinction is given by the family of functions

$$f_s(x) = \frac{2}{s}x(x-s) + 1,$$

illustrated by Figure 2. The pointwise limit is $f_0(0) = 1, f_0(x) = \infty$ for $x \neq 0$. The epi-limit is 0 at 0.

We say that a quadratic form is *finite* if $f(x) < \infty$ for all x. (We note in passing that if a quadratic form is not finite, $f(x) = \infty$ almost everywhere.) The pointwise and epi-limits of quadratic forms agree when the limiting quadratic form is finite, but the example in the figure is not of that sort. This behavior is typical of the applications we consider. In what follows, we take all functional limits to be epi-limits.

It is the epi-limit of functions which is appropriate for optimization theory, as the following theorem (a variation of one one from Rockafellar and Wets (2004)) shows.

Theorem 7 *Let* $f_s : \mathbb{R}^n \to (-\infty, \infty]$ *be eventually ccp, with* $\lim_{s\to 0} f_s = f$. *If* f_s, f *have unique minimizers* $\dot{x}(s), \dot{x}$ *then*

$$\lim_{s\to 0} \dot{x}(s) = \dot{x} \quad \textit{and} \quad \lim_{s\to 0} f_s(\dot{x}(s)) = \inf_x f(x).$$

Proof Given $\delta > 0$, let $B_\delta = \{x \in \mathbb{R}^n : f(x) < f(\dot{x}) + 2\delta\}$. Since $\dot{x}$ is the unique minimizer of f and f is ccp, B_δ is bounded and open, and for any open neighborhood U of $\dot{x}$, $\exists \delta > 0 : B_\delta \subset U$. Note that $x \in \partial B_\delta$ iff $f(x) = f(\dot{x}) + 2\delta$.

Let $\hat{x} : [0, \infty) \to \mathbb{R}$ satisfy property 2 of definition 6 with $\hat{x}(0) = \dot{x}$. Let $s_0 > 0$ be such that $\forall s \in (0, s_0) : f_s(\hat{x}(s)) < f(\dot{x}) + \delta$ and $\hat{x}(s) \in B_\delta$. Thus, $\forall s \in (0, s_0) : \inf_{x \in B_\delta} f_s(x) < f(\dot{x}) + \delta$.

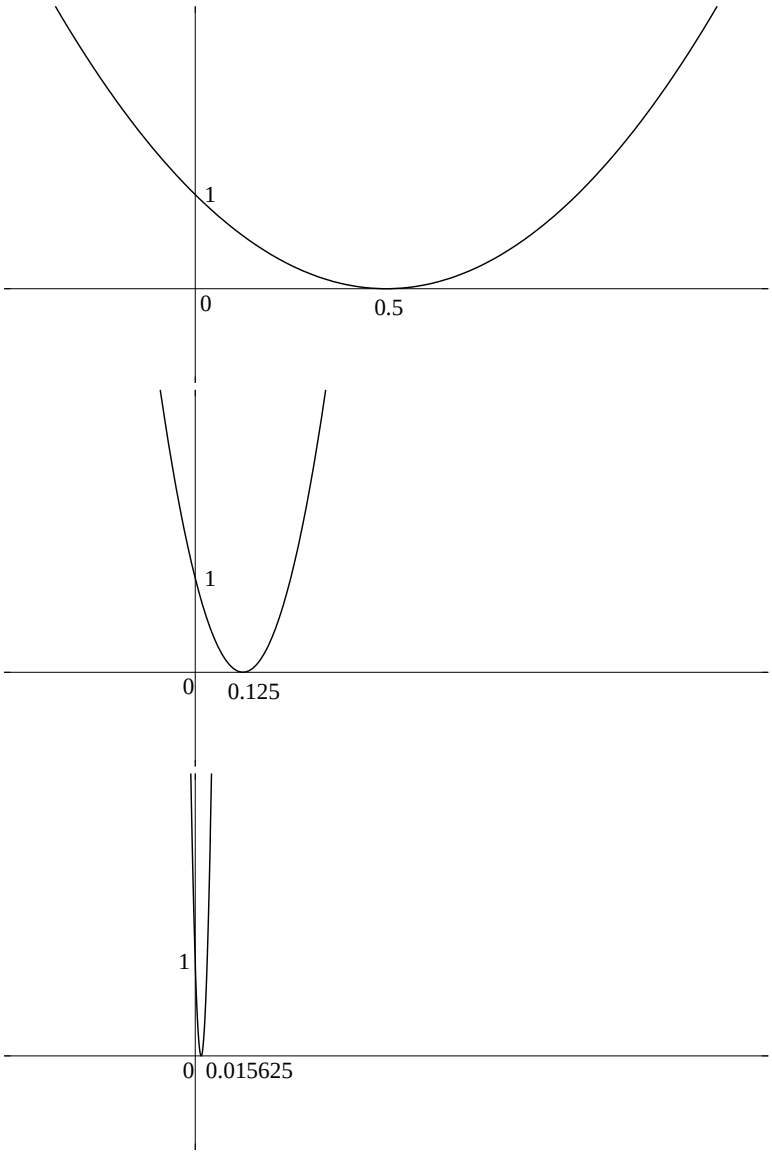


Figure 2: The function above, $f_s(x) = \frac{2}{s}x(x-s)+1$, has different pointwise and epi-limits, having values 1 and 0, respectively, at $x = 0$ and ∞ for all other x.

By property 1 of definition 6, $\forall x \in \mathbb{R}^n, \liminf_{s\to 0} f_s(x) \geq f(x)$, in particular, $\forall x \in \partial B_\delta$, $\exists s_1 \in (0,s_0), \forall s \in (0,s_1) : f_s(x) \geq f(x) - \delta = f(\dot{x}) + \delta$. Since ∂B_δ is compact, we can choose $s_1 \in (0,s_0), \forall x \in \partial B_\delta, s \in (0,s_1) : f_s(x) \geq f(\dot{x}) + \delta$.

Thus $\forall x \in \partial B_\delta, s < s_1 : f_s(x) \geq f(\dot{x}) + \delta > \inf_{x \in B_\delta} f_s(x)$, and therefore $\dot{x}(s) \in B_\delta$ by the convexity of f_s.

Summarizing, $\forall \delta > 0, \exists s_1 > 0, \forall s \in (0,s_1) : \dot{x}(s) \in B_\delta$. Hence $\dot{x}(s) \to \dot{x}$ and we have the first limit.

The second limit is a consequence of the first ($\lim_{s\to 0} \dot{x}(s) = \dot{x}$) and definition 6. In particular, $\limsup_{s\to 0} f_s(\hat{x}(s)) \leq f(\dot{x})$ and $f(\dot{x}) \leq \liminf_{s\to 0} f_s(\dot{x}(s))$. Since $\forall s : f_s(\dot{x}(s)) \leq f_s(\hat{x}(s))$, we have $\limsup_{s\to 0} f_s(\dot{x}(s)) \leq f(\dot{x})$ and hence $f(\dot{x}) \leq \liminf_{s\to 0} f_s(\dot{x}(s)) \leq \limsup_{s\to 0} f_s(\dot{x}(s)) \leq f(\dot{x})$. ■

We now apply this theorem to characterize limits of quadratic forms (which are becoming infinite in the limit). The following lemma is elementary.

Lemma 8 *Let $A(s)$ be a continuous matrix-valued function. If $A(0)$ is non-singular, then $A(s)^{-1}$ exists for a neighborhood of $s = 0$.*

Lemma 9 *Let $Z(s) \in \mathbb{R}^{n\times n}$ be a continuous matrix valued function defined for $s \geq 0$ such that $Z(0) = 0$ and $Z(s)$ is non-singular for $s > 0$.*

Let $M(s) = \begin{pmatrix} A(s) & B(s)^t \\ B(s) & C(s) \end{pmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$ be a continuous symmetric matrix valued function of s such that $M(s)$ is positive semi-definite and $C(s)$ is positive definite for $s \geq 0$. If

$$f_s(x,y) = \begin{pmatrix} x \\ Z(s)^{-1}y \end{pmatrix}^t \begin{pmatrix} A(s) & B(s)^t \\ B(s) & C(s) \end{pmatrix} \begin{pmatrix} x \\ Z(s)^{-1}y \end{pmatrix}$$

then $\lim_{s\to 0} f_s = f$, where

$$f(x,y) = \begin{cases} \infty & y \neq 0 \\ x^t(A(0) - B(0)^t C(0)^{-1} B(0))x & y = 0 \end{cases}.$$

Proof Completing the square,

$$f_s(x,y) = ||x||^2_{\tilde{A}(s)} + ||Z(s)^{-1}y + C(s)^{-1}B(s)x||^2_{C(s)}$$

where $||v||^2_W = v^t W v$ and $\tilde{A}(s) = A(s) - B(s)^t C(s)^{-1} B(s)$. Note that $\tilde{A}$(s) is positive semi-definite and continuous at $s = 0$.

Let $b, c, s_0 > 0$ be chosen such that $\forall s < s_0$

$$b||\cdot|| > ||B(s)\cdot||, \quad ||\cdot||_{C(s)} > c||\cdot||$$

(Such quantities arise from the the singular values of the matrices involved, which are continuous in s). Let $z(s) = ||Z(s)||$ (matrix 2-norm). Note: z is continuous with $z(0) = 0$.

Let $x(s), y(s)$ be continuous at $s = 0$. If $y(0) \neq 0$, then for $s < s_0$

$$\begin{aligned} \sqrt{f_s(x(s),y(s))} &\geq ||Z(s)^{-1}y(s) + C(s)^{-1}B(s)x(s)||_{C(s)} \\ &\geq ||Z(s)^{-1}y(s)||_{C(s)} - ||C(s)^{-1}B(s)x(s)||_{C(s)}, \end{aligned}$$

by the triangle inequality

$$\begin{aligned}\sqrt{f_s(x(s),y(s))} &\geq ||Z(s)^{-1}y(s)||_{C(s)} - ||B(s)x(s)||_{C(s)^{-1}} \\ &> c\frac{||y(s)||}{z(s)} - \frac{b}{c}||x(s)|| \\ &= c\left(\frac{||y(s)||}{z(s)} - \frac{b}{c^2}||x(s)||\right).\end{aligned}$$

By continuity, $\exists s_1 \in (0,s_0)$ such that $\forall s \in (0,s_1)$,

$$||x(s)|| < \frac{3}{2}||x(0)||, \quad ||y(s)|| > \frac{1}{2}||y(0)||, \quad \frac{b}{c^2}||x(0)|| < \frac{||y(0)||}{6z(s)}.$$

Thus, for all $s < s_1 : \sqrt{f_s(x(s),y(s))} > c\frac{||y(0)||}{4z(s)}$, and hence $\liminf_{s\to 0} f_s(x(s),y(s)) = \infty$, which implies property 1 of definition 6 (and property 2, since $\liminf \leq \limsup$). Otherwise ($y(0) = 0$), $f_s(x(s),y(s)) \geq ||x(s)||^2_{\tilde{A}(s)}$ and thus

$$\begin{aligned}\lim_{s\to 0}||x(s)||^2_{\tilde{A}(s)} &\leq \liminf_{s\to 0} f(x(s),y(s)) \\ ||x(0)||^2_{\tilde{A}(0)} &\leq \liminf_{s\to 0} f(x(s),y(s))\end{aligned}$$

(property 1). $f_s(x(s),y(s)) = ||x(s)||^2_{\tilde{A}(s)}$ when $y(s) = -Z(s)C(s)^{-1}B(s)x(s)$ (which is continuous and vanishing at $s = 0$), and thus

$$\limsup_{s\to 0} f(x(s),y(s)) = \lim_{s\to 0}||x(s)||^2_{\tilde{A}(s)} = ||x(0)||^2_{\tilde{A}(0)}$$

(property 2). ■

The following application of the lemma allows us to deal with matrices which will be of specific interest to us.

Corollary 10 *Let $Z_1(s) \in \mathbb{R}^{l\times l}$ and $Z_2(s) \in \mathbb{R}^{n\times n}$ be continuous matrix valued functions defined for $s \geq 0$ such that $Z_i(0) = 0$ and $Z_i(s)$ is non-singular for $s > 0$.*

Let $M(s) = \begin{pmatrix} A(s) & B(s)^t & C(s)^t \\ B(s) & D(s) & E(s)^t \\ C(s) & E(s) & F(s)\end{pmatrix} \in \mathbb{R}^{(l+m+n)\times(l+m+n)}$ be a continuous symmetric matrix valued function of s such that $M(s)$ is positive semi-definite and $F(s)$ is positive definite for $s \geq 0$. If

$$f_s(q_a,q_b,q_c) = \begin{pmatrix} Z_1(s)q_a \\ q_b \\ Z_2(s)^{-1}q_c\end{pmatrix}^t \begin{pmatrix} A(s) & B(s)^t & C(s)^t \\ B(s) & D(s) & E(s)^t \\ C(s) & E(s) & F(s)\end{pmatrix} \begin{pmatrix} Z_1(s)q_a \\ q_b \\ Z_2(s)^{-1}q_c\end{pmatrix}$$

then $\lim_{s\to 0} f_s = f$, where

$$f(q_a,q_b,q_c) = \begin{cases} \infty & q_c \neq 0 \\ q_b^t(D(0) - E(0)^tF(0)^{-1}E(0))q_b & z = 0\end{cases}.$$

Proof We apply Lemma 9 to the quadratic form given by

$$\begin{pmatrix} \begin{pmatrix} q_a \\ q_b \end{pmatrix} \\ Z_2^{-1} q_c \end{pmatrix}^t \begin{pmatrix} \begin{pmatrix} Z_1^t A Z_1 & Z_1^t B^t \\ B Z_1 & D \end{pmatrix} & \begin{pmatrix} Z_1^t C^t \\ E^t \end{pmatrix} \\ \begin{pmatrix} C Z_1 & E \end{pmatrix} & F \end{pmatrix} \begin{pmatrix} \begin{pmatrix} q_a \\ q_b \end{pmatrix} \\ Z_2^{-1} q_c \end{pmatrix}$$

(s dependence suppressed). ■

We will have occasion to apply Corollary 10 when some of q_a, q_b and q_c are empty. In all cases, the appropriate result can be re-derived under the convention that a quadratic form over a 0 variables is identically 0.[1]

5. Kernel Expansions and Regularization Limits

In this section, we present our key result, characterizing the asymptotic behavior of the regularization term of Tikhonov regularization. We define a family of quadratic forms on the polynomials in x; these forms will turn out to be the limits of the quadratic Tikhonov regularizer.

Definition 11 *Let $\kappa(x,x') = \sum_{i,j\geq 0} M_{ij} x^{I_i} x'^{I_j}$, with M symmetric, positive definite. For any $p > 0$, define $R_p^K : f \to [0,\infty]$ by*

$$R_p^\kappa(f) = \begin{cases} 0 & f(x) = \sum_{0\leq i\leq d_{\lfloor p \rfloor}} q_i x^{I_i} \\ \infty & else \end{cases}, \quad \text{if } p \notin \mathbb{Z}$$

$$R_p^\kappa(f) = \begin{cases} \begin{pmatrix} q_{d_{p-1}+1} \\ \vdots \\ q_{d_p} \end{pmatrix}^t C \begin{pmatrix} q_{d_{p-1}+1} \\ \vdots \\ q_{d_p} \end{pmatrix} & f(x) = \sum_{0\leq i\leq d_p} q_i x^{I_i} \\ \infty & else \end{cases}, \quad \text{if } p \in \mathbb{Z}$$

where, for $p \in \mathbb{Z}$, $C = (M_{bb} - M_{ba} M_{aa}^{-1} M_{ab})^{-1}$ where M_{aa} and $\begin{pmatrix} M_{aa} & M_{ab} \\ M_{ba} & M_{bb} \end{pmatrix}$ are the $d_{p-1} \times d_{p-1}$ and $d_p \times d_p$ upper-left submatrices of K.

The q_i in the conditions for f above are arbitary, and hence the conditions are both equivalent to $f \in \text{span}\{x^I : |I| \leq p\}$. We have written the q_i explicitly merely to define the value R_p^κ when $p \in \mathbb{Z}$.

Define $v(X) = \begin{pmatrix} 1 & X^{I_1} & X^{I_2} & \cdots \end{pmatrix} \in \mathbb{R}^{n\times\infty}$. Let $v(X) = \begin{pmatrix} v_\alpha(X) & v_\beta(X) \end{pmatrix}$ be a block decomposition into an $n \times n$ block (a *Vandermonde* matrix on the data) and an $n \times \infty$ block. Because our data set is generic, $v_\alpha(X)$ is non-singular, and the interpolating polynomial through the points (x_i, y_i) over the monomials $\{x^{I_i} : i < n\}$ is given by $f(x) = v_\alpha(x) v_\alpha(X)^{-1} y$.

We now state and prove our key result, showing the convergence of the regularization term of Tikhonov regularization to R_p^κ.

Theorem 12 *Let X be generic and $\kappa(x,x') = \sum_{i,j\geq 0} M_{ij} x^{I_i} x'^{I_j}$ be a valid kernel. Let $p \in [0, |I_{n-1}|)$. Let $f_s(y) = s^{2p} y^t \kappa(sX, sX)^{-1} y$. Then*

$$\lim_{s\to 0} f_s = f,$$

where $f(y) = R_p^\kappa(q)$, and $q(x) = v_\alpha(x)\tilde{q} = \sum_{0\leq i<n} \tilde{q}_i x^{I_i}$ and $\tilde{q} = v_\alpha(X)^{-1} y$.

1. This is not a definition. We are merely stating (without proof) that *if* we were to go through the proofs omitting some of q_a, q_b, and q_c, we would obtain the same result.

Proof Recalling that $v_\alpha(X)$ is non-singular by genericity, define $\chi = v_\alpha(X)^{-1}v_\beta(X)$. Let $\Sigma(s)$ be the infinite diagonal matrix valued function of s whose i^{th} diagonal element is $s^{|I_i|}$. We define a block decomposition $\Sigma(s) = \begin{pmatrix} \Sigma_\alpha(s) & 0 \\ 0 & \Sigma_\beta(s) \end{pmatrix}$ where $\Sigma_\alpha(s)$ is $n \times n$. We likewise partition M into blocks $M = \begin{pmatrix} M_{\alpha\alpha} & M_{\alpha\beta} \\ M_{\beta\alpha} & M_{\beta\beta} \end{pmatrix}$ where $M_{\alpha\alpha}$ is $n \times n$.

Thus,

$$\begin{aligned}
& \kappa(sX, sX) \\
= \;& v(X)\Sigma(s)M\Sigma(s)v(X)^t \\
= \;& v_\alpha(X)\begin{pmatrix} I & \chi \end{pmatrix}\Sigma(s)M\Sigma(s)\begin{pmatrix} I \\ \chi^t \end{pmatrix}v_\alpha(X)^t \\
= \;& v_\alpha(X)\Sigma_\alpha(s)\begin{pmatrix} I & \Sigma_\alpha(s)^{-1}\chi\Sigma_\beta(s) \end{pmatrix}M\begin{pmatrix} I \\ (\Sigma_\alpha(s)^{-1}\chi\Sigma_\beta(s))^t \end{pmatrix}\Sigma_\alpha(s)v_\alpha(X)^t \\
= \;& v_\alpha(X)\Sigma_\alpha(s)\begin{pmatrix} I & \tilde{\chi}(s) \end{pmatrix}M\begin{pmatrix} I \\ \tilde{\chi}(s)^t \end{pmatrix}\Sigma_\alpha(s)v_\alpha(X)^t \\
= \;& v_\alpha(X)\Sigma_\alpha(s)\tilde{M}(s)\Sigma_\alpha(s)v_\alpha(X)^t,
\end{aligned}$$

where we have implicitly defined

$$\begin{aligned}
\tilde{\chi}(s) &\equiv \Sigma_\alpha(s)^{-1}\chi\Sigma_\beta(s) \\
\tilde{M}(s) &\equiv \begin{pmatrix} I & \tilde{\chi}(s) \end{pmatrix}M\begin{pmatrix} I \\ \tilde{\chi}(s)^t \end{pmatrix}.
\end{aligned}$$

For $0 \le i < n$, $0 \le j < \infty$, the i,jth entry of $\tilde{\chi}(s)$ is $s^{|I_{j+n}|-|I_i|}\chi_{ij}$, and $|I_{j+n}| - |I_i| \ge 0$. Thus, $\lim_{s\to 0}\tilde{\chi}(s)$ exists and we denote it $\tilde{\chi}(0)$. We note that $\tilde{\chi}_{ij}(0)$ is non-zero if and only if $|I_i| = |I_{j+n}|$. In particular,

$$\tilde{\chi}_{ij}(0) = \begin{cases} \chi_{ij} & d_{|I_n|-1} \le i < n \text{ and } 0 \le j < d_{|I_n|} - n \\ 0 & \text{otherwise} \end{cases}$$

Therefore, $\lim_{s\to 0}\tilde{M}(s) = \begin{pmatrix} I & \tilde{\chi}(0) \end{pmatrix}M\begin{pmatrix} I \\ \tilde{\chi}(0)^t \end{pmatrix}$ exists and is positive definite (since $\begin{pmatrix} I & \tilde{\chi}(0) \end{pmatrix}^t$ is full rank); we denote it by $\tilde{M}(0)$. Additionally, since the first $d_{|I_n|-1}$ rows of $\tilde{\chi}(0)$ (and therefore the first $d_{|I_n|-1}$ columns of $\tilde{\chi}(0)^t$) are identically zero, the $d_{|I_n|-1} \times d_{|I_n|-1}$ upper-left submatrices of $\tilde{M}(0)$ and M are equal.

Summarizing,

$$\begin{aligned}
f_s(y) &= s^{2p}y^t\kappa(sX, sX)^{-1}y \\
&= (v_\alpha(X)^{-1}y)^t(s^p\Sigma_\alpha(1/s))\tilde{M}(s)^{-1}(s^p\Sigma_\alpha(1/s))(v_\alpha(X)^{-1}y) \\
&= \tilde{q}^t(s^p\Sigma_\alpha(1/s))\tilde{M}(s)^{-1}(s^p\Sigma_\alpha(1/s))\tilde{q},
\end{aligned}$$

where $\tilde{q} \equiv v_\alpha(X)^{-1}y$. We will take the limit by applying Corollary 10.

Consider first the situation where $p \in \mathbb{Z}$. The first d_{p-1} diagonal entries are of the form s^k for $k > 0$, the "middle" $d_{p-1} - d_p$ entries are exactly 1, and the last $n - d_p$ diagonal entries are of the

form s^{-k} for $k > 0$, We define three subsets of $\{0,\ldots,n-1\}$ (with subvectors and submatrices defined accordingly): $lo = \{0,\ldots,d_{p-1}-1\}$, $mi = \{d_{p-1},\ldots,d_p-1\}$, and $hi = \{d_p,\ldots,n-1\}$. (Note it is possible for one of lo or hi to be empty, in (respectively) the cases where $p = 0$ or $d_p = n$.) By Corollary 10, with $q_1 = \tilde{q}_{lo}$, $q_2 = \tilde{q}_{mi}$, and $q_3 = \tilde{q}_{hi}$, $Z_1(s) = (s^p\Sigma_\alpha(1/s))_{lo,lo}$, and $Z_2^{-1}(S) = (s^p\Sigma_\alpha(1/s))_{hi,hi}$, and

$$\begin{pmatrix} A(s) & B(s) & C(s) \\ B(s)^t & D(s) & E(s) \\ C(s)^t & E(s)^t & F(s) \end{pmatrix} = \begin{pmatrix} \tilde{M}(s)^{-1}_{lo,lo} & \tilde{M}(s)^{-1}_{lo,mi} & \tilde{M}(s)^{-1}_{lo,hi} \\ \tilde{M}(s)^{-1}_{mi,lo} & \tilde{M}(s)^{-1}_{mi,mi} & \tilde{M}(s)^{-1}_{mi,hi} \\ \tilde{M}(s)^{-1}_{hi,lo} & \tilde{M}(s)^{-1}_{hi,mi} & \tilde{M}(s)^{-1}_{hi,hi} \end{pmatrix}.$$

By Lemma 16

$$\begin{aligned} D(0) - E(0)^t F(0)^{-1} E(0) &= \tilde{M}(0)^{-1}_{mi,mi} - (\tilde{M}(0)^{-1}_{hi,mi})^t (\tilde{M}(0)^{-1}_{hi,hi})^{-1} \tilde{M}(0)^{-1}_{hi,mi} \\ &= (\tilde{M}(0)_{mi,mi} - \tilde{M}(0)_{mi,lo}\tilde{M}(0)^{-1}_{lo,lo}\tilde{M}(0)_{lo,mi})^{-1} \\ &= (M_{mi,mi} - M_{mi,lo}M^{-1}_{lo,lo}M_{lo,mi})^{-1}, \end{aligned}$$

where the final equality is the result of the $d_{|I_n|-1} \times d_{|I_n|-1}$ upper-left submatrices of $\tilde{M}(0)$ and M are equal, shown above.

By Corollary 10, we have that $\lim_{s\to 0} \tilde{q}^t (s^p\Sigma_\alpha(1/s))\tilde{M}(s)^{-1}(s^p\Sigma_\alpha(1/s))\tilde{q}$ is ∞ if ($hi \neq \emptyset$ and) $\tilde{q}_{hi} \neq 0$. If ($hi = \emptyset$ or) $\tilde{q}_{hi} = 0$, the limit is:

$$\tilde{q}^t_{mi}(M_{mi,mi} - M_{mi,lo}M^{-1}_{lo,lo}M_{lo,mi})^{-1}\tilde{q}_{mi},$$

hence $f_s(y) \to R^\kappa_p(q)$ for $p \in \mathbb{Z}$.

When $p \notin \mathbb{Z}$, the proof proceeds along very similar lines; we merely point out that in this case, we will take $lo = \{0,\ldots,d_{\lfloor p\rfloor}-1\}, mi = \emptyset$, and $hi = \{d_{\lfloor p\rfloor},\ldots,n-1\}$. Since mi is empty, the application of Corollary 10 yields 0 when $\tilde{q}_{hi} = 0$, and ∞ otherwise. ■

The proof assumes $p \in [0, |I_{n-1}|)$. In other words, we can get polynomial behavior of degree $\lfloor p\rfloor$ for any p, but we must have at least $d_{\lfloor p\rfloor} = O(d^{\lfloor p\rfloor})$ generic data points in order to do so.

We have shown that if $\lambda(s) = s^{2p}$ for a p in a suitable range, that the regularization term approaches the indicator function for polynomials of degree p in the data with (when $p \in \mathbb{Z}$) a residual regularization on the degree p monomial coefficients which is a quadratic form given by some combination of the coefficients of the power series expansion of $\kappa(x,x')$. Obtaining these coefficients in general may be awkward. However, for kernels which satisfy Lemma 1, this can be done easily.

Lemma 13 *If κ satisfies the conditions of Lemma 1, then for $p \in \mathbb{Z}$ and $q(x) = \sum_{|I|\le p} \tilde{q}_I x^I$*

$$R^\kappa_p(q) = (g_p(0))^{-2} \sum_{|I|=p} \binom{|I|}{I}^{-1} q_I^2. \tag{9}$$

Proof Let L be defined according to the proof of Lemma 1. Lemma 17 applies with G and J being the consecutive $d_{p-1} \times d_{p-1}$ and $(d_p - d_{p-1}) \times (d_p - d_{p-1})$ diagonal blocks of L. Finally, we note that J is itself a diagonal matrix and hence, $(JJ^t)^{-1}$ is diagonal with elements equal to the inverse squares of J's, i.e. of the form $\binom{|I|}{I}^{-1}(g_{|I|}(0))^{-2}$ where $|I| = p$. ■

Note, the Gaussian kernel is of this sort with $(g_p(0))^2 = \frac{1}{p!}$.

It is also worth noting that for kernels admitting such a decomposition $R_p^\kappa(q)$ is invariant under "rotations" of the form $q \to q'$ where $q(x) = q'(Ux)$ with U a rotation matrix. Since any $R_p^\kappa(q) = 0$ for q of degree $< p$ it is clearly translation invariant. We speculate that any quadratic function of the coefficients of a polynomial which is both translation and rotation invariant in this way must have of the form (9).

6. The Asymptotic Regularized Solution

By Theorem 12, the regularization term (under certain conditions) becomes a penalty on degree $> p$ behavior of the regularized solution. Since the loss function is fixed as $\sigma, \frac{1}{\lambda} \to \infty$, the objective function in (1) approaches a limiting constrained optimization problem.

Theorem 14 *Let $v : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a valid loss function and $\kappa(x,x')$ be a valid kernel function. Let $\kappa_\sigma(x,x') = \kappa(\sigma^{-1}x, \sigma^{-1}x')$. Let $p \in [0, |I_{n-1}|)$ with $\lambda(\sigma) = \tilde{\lambda}\sigma^{-2p}$ for some fixed $\tilde{\lambda} > 0$.*

Let $\dot{f}_\sigma, \dot{f}_\infty \in \mathcal{H}$ be the unique minimizers of

$$n\lambda(\sigma)||f||^2_{\kappa_\sigma} + \sum_{i=1}^{n} v(f(x_i), \hat{y}_i) \tag{10}$$

and

$$n\tilde{\lambda}R_p^\kappa(f) + \sum_{i=1}^{n} v(f(x_i), \hat{y}_i) \tag{11}$$

respectively.

Then $\forall x_0 \in \mathbb{R}^d$ such that $X_0 = \begin{pmatrix} x_0 \\ X \end{pmatrix}$ is generic,

$$\lim_{\sigma \to \infty} \dot{f}_\sigma(x_0) = \dot{f}_\infty(x_0).$$

Proof In the value-based learning formulation, problem 10 becomes

$$n\lambda(\sigma) y^t K_\sigma^{-1} y + \sum_{i=1}^{n} v(y_i, \hat{y}_i) \tag{12}$$

where $y \in \mathbb{R}^n$.

By Corollary 3, if we consider the expanded problem which includes the test point in the regularization but not in the loss,

$$n\lambda(\sigma) z^t \begin{pmatrix} \kappa(x_0,x_0) & k_\sigma^t \\ k_\sigma & K_\sigma \end{pmatrix}^{-1} z + \sum_{i=1}^{n} v(z_i, \hat{y}_i), \tag{13}$$

then the minimizers of problems 12 and 13 are related via $\dot{z}_{\sigma i} = \dot{y}_{\sigma i} = \dot{f}_\sigma(x_i), 1 \le i \le n$ and $\dot{z}_{\sigma 0} = k_\sigma K_\sigma^{-1} \dot{y}_\sigma = \dot{f}_\sigma(x_0)$. Because X_0 is generic, we can make the change of variables $z_i = q(x_i) = \sum_{j=0}^{n} \beta_j x_i^{I_j}$ in (13), yielding

$$g_\sigma(q) = n\lambda(\sigma)||q||^2_{\kappa_\sigma} + \sum_{i=1}^{n} v(q(x_i), \hat{y}_i) \tag{14}$$

with minimizer $\dot{q}_\sigma$ satisfying $\dot{q}_\sigma(x_i) = \dot{z}_{\sigma i}$ (in particular $\dot{q}_\sigma(x_0) = \dot{z}_{\sigma 0} = \dot{f}_\sigma(x_0)$).

Let $g_\infty(q) = n\tilde{\lambda}R_p^\kappa(q) + \sum_{i=1}^n v(q(x_i), \hat{y}_i)$ with minimizer $\dot{q}_\infty$. By Theorem 12, $g_\sigma \to g_\infty$, thus, by Theorem 7, $\dot{q}_\sigma(x_0) \to \dot{q}_\infty(x_0) = \dot{f}_\infty(x_0)$. ■

We note that in Theorem 14, we have assumed that problems 10 and 11 have unique minimizers. For any fixed σ, $||f||^2_{\kappa_\sigma}$ is strictly convex, so problem 10 will always have a unique minimizer. For strictly convex loss functions, such as the square loss used in regularized least squares, problem 11 will have a unique minimizer as well. If we consider a non-strictly convex loss function, such as the hinge loss used in SVMs, problem 11 may not have a unique minimizer; for example, it is easy to see that in a classification task where the data is *separable* by a degree p polynomial, any (appropriately scaled) degree p polynomial that separates the data will yield an optimal solution to problem 11 with cost 0. In these cases, Theorem 12 still determines the *value* of the limiting solution, but Theorem 14 does not completely determine the limiting minimizer. Theorem 7.33 of Rockafellar and Wets (2004) provides a generalization of Theorem 14 which applies when the minimizers are non-unique (and even when the objective functions are non-convex, as long as certain *local convexity* conditions hold). It can be shown that the minimizer of problem 10 will converge to one of the minimizers of problem 11, though not knowing which one, we cannot predict the limiting regularized solution. In practice, we expect that when the data is not separable by a low-degree polynomial (most real-world data sets are not), problem 11 will have a unique minimizer.

Additionally, we note that our work has focused on "standard" Tikhonov regularization problems, in which the function f is "completely" regularized. In practice, the SVM (for reasons that we view as largely historical, although that is beyond the scope of this paper) is usually implemented with an unregularized bias term b. We point out that our main result still applies. In this case,

$$\begin{aligned}
&\inf_{b\in\mathbb{R}, f\in\mathcal{H}} \left\{ n\lambda||f||_{\kappa_\sigma} + \sum_{i=1}^n (1 - (f(x_i)+b)\hat{y}_i)_+ \right\} \\
= \quad &\inf_b \left\{ \inf_f \left\{ n\lambda||f||_{\kappa_\sigma} + \sum_{i=1}^n (1 - (f(x_i)+b)\hat{y}_i)_+ \right\} \right\} \\
\to \quad &\inf_b \left\{ \inf_f \left\{ n\tilde{\lambda}R_p^\kappa(f) + \sum_{i=1}^n (1 - (f(x_i)+b)\hat{y}_i)_+ \right\} \right\},
\end{aligned}$$

with our results applying to the inner optimization problem (where b is fixed). When an unregularized bias term is used, problem 10 may not have a unique minimizer either. The conditions for non-uniqueness of 10 for the case of support vector machines are explored in Burges and Crisp (1999); the conditions are fairly pathological, and SVMs nearly always have unique solutions in practice. Finally, we note that the limiting problem is one where all polynomials of degree $< p$ are free, and hence, the bias term is "absorbed" into what is already free in the limiting problem.

7. Prior Work

We are now in a position to discuss in some detail the previous work on this topic.

In Keerthi and Lin (2003), it was observed that SVMs with Gaussian kernels produce classifiers which approach those of linear SVMs as $\sigma \to \infty$ (and $\frac{1}{2\lambda} = C = \tilde{C}\sigma^2 \to \infty$). The proof is based on an expansion of the kernel function (Equation 2.8 from Keerthi and Lin (2003)):

$$\kappa_\sigma(x, x') \quad = \quad \exp(-||x - x'||^2/\sigma^2)$$

$$= \quad 1 - \frac{||x||^2}{2\sigma^2} - \frac{||x'||^2}{2\sigma^2} + \frac{x \cdot x'}{\sigma^2} + o(||x - x'||/\sigma^2)$$

where κ_σ is approximated by the four leading terms in this expansion. This approximation ($\kappa_\sigma(x,x') \sim 1 - \sigma^{-2}(||x||^2 - ||x'||^2 + 2x \cdot x')/2$) does not satisfy the Mercer condition, so the resulting dual objective function is not positive definite (remark 3 of Keerthi and Lin (2003)). However, by showing that the domain of the dual optimization problem is bounded (because of the dual box constraints), one avoids the unpleasant effects of the Mercer violation. The Keerthi and Lin (2003) result is a special case of our result, where we choose the Gaussian loss function and $p = 1$.

In Lippert and Rifkin (2006), a similar observation was made in the case of Gaussian regularized least squares. In this case, for any degree p, an asymptotic regime was identified in which the regularized solution approached the least squares degree-p polynomial. The result hinges upon the simultaneous cancellation effects between the coefficients $c(\sigma,\lambda)$ and the kernel function κ_σ in the kernel expansion of $f(x)$, with $f(x)$ and $c(\sigma,\lambda)$ given by

$$\begin{aligned} f(x) &= \sum_i c_i(\sigma,\lambda)\kappa_\sigma(x,x_i) \\ c(\sigma,\lambda) &= (\kappa_\sigma(X,X) + n\lambda I)^{-1} y \end{aligned}$$

when $\kappa_\sigma(x,x') = \exp(-||x-x'||^2/\sigma^2)$. In that work, we considered only *non-integer* p, so there was no residual regularization. The present work generalizes the result to arbitrary p and arbitrary convex loss-functions. Note that in our previous work, we did not work with the value-based formulation of learning, and we were forced to take the limit of an expression combining training and testing kernel products, exploiting the explicit nature of the regularized least squares equations. In the present work, the value-based learning formulation allows us to avoid such issues, obtaining much more general results.

8. Experimental Evidence

In this section, we present a simple experiment that illustrates our results. This example was first presented in our earlier work (Lippert and Rifkin, 2006).

We consider the fifth-degree polynomial function

$$f(x) = .5(1-x) + 150x(x-.25)(x-.3)(x-.75)(x-.95),$$

over the range $x \in [0,1]$. Figure 3 plots f, along with a 150 point data set drawn by choosing x_i uniformly in $[0,1]$, and choosing $y = f(x) + \varepsilon_i$, where ε_i is a Gaussian random variable with mean 0 and standard deviation .05. Figure 3 also shows (in red) the best polynomial approximations to the data (not to the ideal f) of various orders. (We omit third order because it is nearly indistinguishable from second order.)

According to Theorem 14, if we parametrize our system by a variable s, and solve a Gaussian regularized least-squares problem with $\sigma^2 = s^2$ and $\lambda = \tilde{\lambda} s^{-(2p+1)}$ for some integer p, then, as $s \to \infty$, we expect the solution to the system to tend to the pth-order data-based polynomial approximation to f. Asymptotically, the value of the constant $\tilde{\lambda}$ does not matter, so we (arbitrarily) set it to be 1. Figure 4 demonstrates this result.

We note that these experiments frequently require setting λ much smaller than machine-ε. As a consequence, we need more precision than IEEE double-precision floating-point, and our results

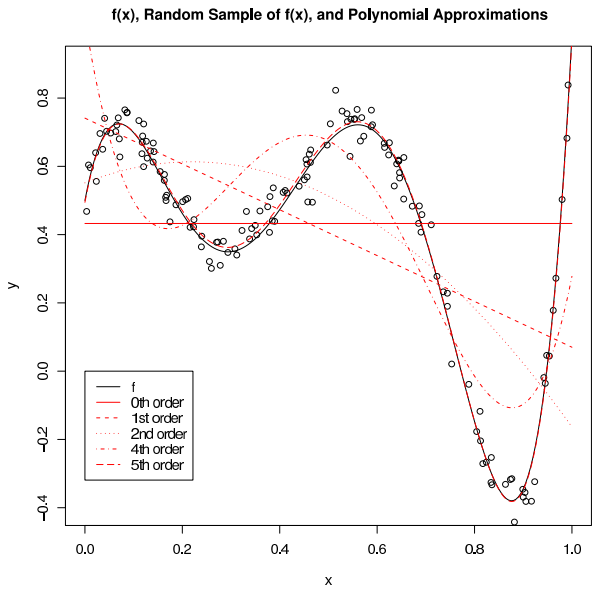


Figure 3: $f(x) = .5(1-x) + 150x(x-.25)(x-.3)(x-.75)(x-.95)$, a random data set drawn from $f(x)$ with added Gaussian noise, and data-based polynomial approximations to f.

cannot be obtained via many standard tools (e.g. MATLAB(TM)). We performed our experiments using CLISP, an implementation of Common Lisp that includes arithmetic operations on arbitrary-precision floating point numbers.

9. Discussion

We have shown, under mild technical conditions, that the minimizer of a Tikhonov regularization problem with a Gaussian kernel with bandwidth σ behaves, as $\sigma \to \infty$ and $\lambda = \tilde{\lambda}\sigma^{-p}$, like the degree-$p$ polynomial that minimizes empirical risk (with some additional regularization on the degree p coefficients when p is an integer). Our approach rested on two key ideas, epi-convergence, which allowed us to make precise statements about when the limits of minimizers converges to the minimizer of a limit, and value-based learning, which allowed us to work in terms of the predicted functional values, y_i, as opposed to the more common technique of working with the coefficients c_i in a functional expansion of the form $f(x) = \sum_i c_i K(x, x_i)$. This in turn allowed us to avoid discussing the limits of the c_i, which we do not know how to characterize.

We are *not* suggesting that practicioners wishing to do polynomial approximation use Gaussian kernels with extreme σ, λ values; there is no difficulty in using standard polynomial kernels directly, and using extreme σ and λ values invites numerical difficulties. However, we think this result highlights a phenomenon which may mislead automated parameter tuning methods (such as selecting σ or λ to minimize some hold-out error). In fact, our earlier work (Lippert and Rifkin, 2006), was motivated by experiments in globally optimizing the LOO error in (λ, σ), where, for some data sets we observed large ranges of decreasing λ and increasing σ which had similar, nearly optimal performance. Wahba et al. (2001) observed the same phenomenon for the SVM, optimizing

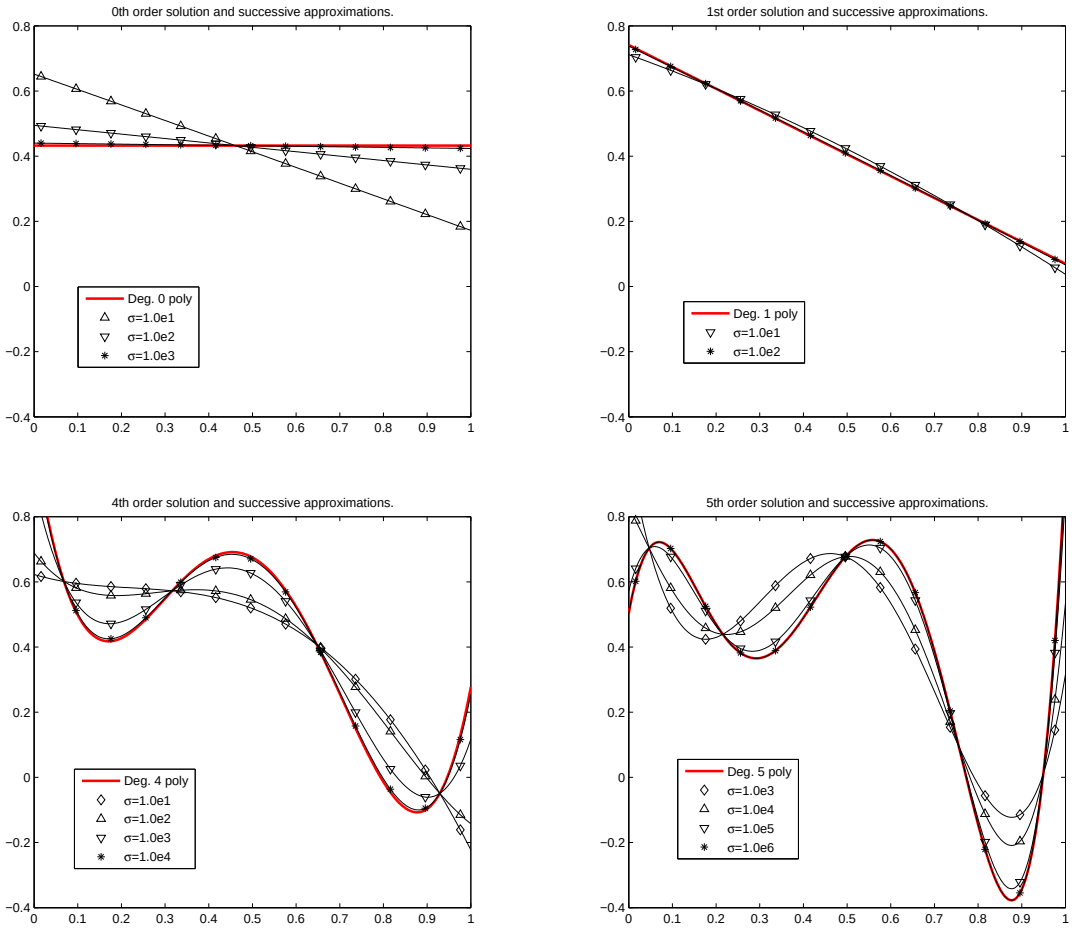


Figure 4: As $s \to \infty$, $\sigma^2 = s^2$ and $\lambda = s^{-(2k+1)}$, the solution to Gaussian RLS approaches the kth order polynomial solution.

performance of a Bayesian weighted misclassification score. One can get some intuition about this tradeoff between smaller λ and larger σ by considering example 4 of Zhou (2002) where a tradeoff between σ and R is seen for the covering numbers of balls in an RKHS induced by a Gaussian kernel (R can be thought of as roughly $\frac{1}{\sqrt{\lambda}}$).

We think it is interesting that some low-rank approximations to Gaussian kernel matrix-vector products (see Yang et al. (2005)) tend to work much better for large values of σ. Our results raise the possibility that these low-rank approximations are merely recovering low-order polynomial behavior; this will be a topic of future study.

We believe the value-based formulation is of quite general utility, and expect to work with it in the future. Because of our choice of kernels, we were able to assume that the kernel matrix K was invertible, and we worked directly with K^{-1} in the value-based formulation. This is not a strong

requirement; it is possible to work with the pseudoinverse of K for finite-dimensional kernels (such as the dot-product kernel).

Acknowledgments

The authors would like to acknowledge Roger Wets and Adrian Lewis for patiently answering questions regarding epi-convergence. We would also like to thank our reviewers for their comments and suggestions.

Appendix A

In this appendix, we state and prove several matrix identities that we use in the main body of the paper.

Lemma 15 *Let $X,U \in \mathbb{R}^{m\times m}$, $Z,W \in \mathbb{R}^{n\times n}$, and $Y,V \in \mathbb{R}^{n\times m}$ with $\begin{pmatrix} X & Y^t \\ Y & Z \end{pmatrix}$ symmetric, positive definite. If*

$$\begin{pmatrix} U & V^t \\ V & W \end{pmatrix}\begin{pmatrix} X & Y^t \\ Y & Z \end{pmatrix} = \begin{pmatrix} I_m & 0 \\ 0 & I_n \end{pmatrix} \tag{15}$$

then

$$\begin{aligned} U &= (X - Y^t Z^{-1} Y)^{-1} && (16)\\ W^{-1}V &= -YX^{-1} && (17)\\ VU^{-1} &= -Z^{-1}Y && (18)\\ W &= (Z - YX^{-1}Y^t)^{-1} && (19) \end{aligned}$$

Proof Since $\begin{pmatrix} X & Y^t \\ Y & Z \end{pmatrix}$, is symmetric, positive definite, $\begin{pmatrix} U & V^t \\ V & W \end{pmatrix}$ is symmetric, positive definite, as are X,Z,U,W.

Multiplying out (15) in block form,

$$\begin{aligned} UX + V^tY &= I_m && (20)\\ VX + WY &= 0 && (21)\\ UY^t + V^tZ &= 0 && (22)\\ VY^t + WZ &= I_n && (23) \end{aligned}$$

Since U,W,X,Z are non-singular, (21) implies (17) and (22) implies (18). Substituting $V = -Z^{-1}YU$ into (20) yields $UX - UY^tZ^{-1}Y = U(X - Y^tZ^{-1}Y) = I_m$ and thus (16). Similarly, $V = -WYX^{-1}$ and (23) give (19). ■

Lemma 16 *Let*

$$M = \begin{pmatrix} A & B^t & C^t \\ B & D & E^t \\ C & E & F \end{pmatrix}$$

be symmetric positive definite. Let

$$M^{-1} = \begin{pmatrix} \bar{A} & \bar{B}^t & \bar{C}^t \\ \bar{B} & \bar{D} & \bar{E}^t \\ \bar{C} & \bar{E} & \bar{F} \end{pmatrix}.$$

Then $\bar{D} - \bar{E}^t \bar{F}^{-1} \bar{E} = (D - BA^{-1}B^t)^{-1}$.

Proof By (16) of Lemma 15, on M with $U = \begin{pmatrix} A & B^t \\ B & D \end{pmatrix}$,

$$\begin{pmatrix} A & B^t \\ B & D \end{pmatrix}^{-1} = \begin{pmatrix} \bar{A} & \bar{B}^t \\ \bar{B} & \bar{D} \end{pmatrix} - \begin{pmatrix} \bar{C}^t \\ \bar{E}^t \end{pmatrix} \bar{F}^{-1} \begin{pmatrix} \bar{C} & \bar{E} \end{pmatrix}.$$

By (19) of Lemma 15, on $\begin{pmatrix} A & B^t \\ B & D \end{pmatrix}$,

$$\begin{pmatrix} A & B^t \\ B & D \end{pmatrix}^{-1} = \begin{pmatrix} \cdots & \cdots \\ \cdots & (D - BA^{-1}B^t)^{-1} \end{pmatrix}.$$

Combining the lower-right blocks of the above two expansions yields the result. ■

Lemma 17 *If*

$$M = \begin{pmatrix} A & B^t & C^t \\ B & D & E^t \\ C & E & F \end{pmatrix} = \begin{pmatrix} G & 0 & 0 \\ H & J & 0 \\ K & N & P \end{pmatrix} \begin{pmatrix} G & 0 & 0 \\ H & J & 0 \\ K & N & P \end{pmatrix}^t.$$

is symmetric positive definite, then $JJ^t = D - BA^{-1}B^t$.

Proof Clearly $\begin{pmatrix} A & B^t \\ B & D \end{pmatrix} = \begin{pmatrix} G & 0 \\ H & J \end{pmatrix} \begin{pmatrix} G & 0 \\ H & J \end{pmatrix}^t$ and thus

$$A = GG^t, \quad B = HG^t, \quad D = JJ^t + HH^t$$

and hence $JJ^t = D - HH^t = D - BG^{-t}G^{-1}B^t = D - B(GG^t)^{-1}B^t = D - BA^{-1}B^t$. ■

References

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

C. J. C. Burges and D. J. Crisp. Uniqueness of the svm solution. In *Neural Information Processing Systems*, 1999.

Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Adv. In Comp. Math.*, 13(1):1–50, 2000.

Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.

Ross A. Lippert and Ryan M. Rifkin. Asymptotics of gaussian regularized least squares. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Adv. in Neural Info. Proc. Sys. 18*. MIT Press, Cambridge, MA, 2006.

R. Tyrrell Rockafellar and Roger J. B. Wets. *Variational Analysis*. Springer, Berlin, 2004.

Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *14th Annual Conference on Computational Learning Theory*, pages 416–426, 2001.

Grace Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Soc. for Industrial & Appl. Math., 1990.

Grace Wahba, Yi Lin, Yoonkyung Lee, and Hao Zhang. On the relation between the GACV and Joachims' $\xi\alpha$ method for tuning support vector machines, with extensions to the non-standard case. Technical Report 1039, U. Wisconsin department of Statistics, 2001. URL `citeseer.ist.psu.edu/wahba01relation.html`.

Changjiang Yang, Ramani Duraiswami, and Larry Davis. Efficient kernel machines using the improved fast gauss transform. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1561–1568, Cambridge, MA, 2005. MIT Press.

D.-X. Zhou. The covering number in learning theory. *J. of Complexity*, 18:739–767, 2002.

Journal of Machine Learning Research 7 (2006) 877-917 Submitted 6/05; Revised 12/05; Published 5/06

Evolutionary Function Approximation for Reinforcement Learning

Shimon Whiteson SHIMON@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Sciences
University of Texas at Austin
1 University Station, C0500
Austin, TX 78712-0233

Editor: Georgios Theocharous

Abstract

Temporal difference methods are theoretically grounded and empirically effective methods for addressing reinforcement learning problems. In most real-world reinforcement learning tasks, TD methods require a function approximator to represent the value function. However, using function approximators requires manually making crucial representational decisions. This paper investigates *evolutionary function approximation*, a novel approach to automatically selecting function approximator representations that enable efficient individual learning. This method *evolves* individuals that are better able to *learn*. We present a fully implemented instantiation of evolutionary function approximation which combines NEAT, a neuroevolutionary optimization technique, with Q-learning, a popular TD method. The resulting NEAT+Q algorithm automatically discovers effective representations for neural network function approximators. This paper also presents *on-line evolutionary computation*, which improves the on-line performance of evolutionary computation by borrowing selection mechanisms used in TD methods to choose individual actions and using them in evolutionary computation to select policies for evaluation. We evaluate these contributions with extended empirical studies in two domains: 1) the mountain car task, a standard reinforcement learning benchmark on which neural network function approximators have previously performed poorly and 2) server job scheduling, a large probabilistic domain drawn from the field of autonomic computing. The results demonstrate that evolutionary function approximation can significantly improve the performance of TD methods and on-line evolutionary computation can significantly improve evolutionary methods. This paper also presents additional tests that offer insight into what factors can make neural network function approximation difficult in practice.

Keywords: reinforcement learning, temporal difference methods, evolutionary computation, neuroevolution, on-line learning

1. Introduction

In many machine learning problems, an agent must learn a *policy* for selecting actions based on its current *state*. *Reinforcement learning* problems are the subset of these tasks in which the agent never sees examples of correct behavior. Instead, it receives only positive and negative rewards for the actions it tries. Since many practical, real world problems (such as robot control, game playing, and system optimization) fall in this category, developing effective reinforcement learning algorithms is critical to the progress of artificial intelligence.

The most common approach to reinforcement learning relies on the concept of *value functions*, which indicate, for a particular policy, the long-term value of a given state or state-action pair. *Temporal difference methods* (TD) (Sutton, 1988), which combine principles of dynamic programming with statistical sampling, use the immediate rewards received by the agent to incrementally improve both the agent's policy and the estimated value function for that policy. Hence, TD methods enable an agent to learn during its "lifetime" i.e. from its individual experience interacting with the environment.

For small problems, the value function can be represented as a table. However, the large, probabilistic domains which arise in the real-world usually require coupling TD methods with a *function approximator*, which represents the mapping from state-action pairs to values via a more concise, parameterized function and uses supervised learning methods to set its parameters. Many different methods of function approximation have been used successfully, including CMACs, radial basis functions, and neural networks (Sutton and Barto, 1998). However, using function approximators requires making crucial representational decisions (e.g. the number of hidden units and initial weights of a neural network). Poor design choices can result in estimates that diverge from the optimal value function (Baird, 1995) and agents that perform poorly. Even for reinforcement learning algorithms with guaranteed convergence (Baird and Moore, 1999; Lagoudakis and Parr, 2003), achieving high performance in practice requires finding an appropriate representation for the function approximator. As Lagoudakis and Parr observe, "The crucial factor for a successful approximate algorithm is the choice of the parametric approximation architecture(s) and the choice of the projection (parameter adjustment) method." (Lagoudakis and Parr, 2003, p. 1111) Nonetheless, representational choices are typically made manually, based only on the designer's intuition.

Our goal is to automate the search for effective representations by employing sophisticated optimization techniques. In this paper, we focus on using evolutionary methods (Goldberg, 1989) because of their demonstrated ability to discover effective representations (Gruau et al., 1996; Stanley and Miikkulainen, 2002). Synthesizing evolutionary and TD methods results in a new approach called *evolutionary function approximation*, which automatically selects function approximator representations that enable efficient individual learning. Thus, this method *evolves* individuals that are better able to *learn*. This biologically intuitive combination has been applied to computational systems in the past (Hinton and Nowlan, 1987; Ackley and Littman, 1991; Boers et al., 1995; French and Messinger, 1994; Gruau and Whitley, 1993; Nolfi et al., 1994) but never, to our knowledge, to aid the discovery of good TD function approximators.

Our approach requires only 1) an evolutionary algorithm capable of optimizing representations from a class of functions and 2) a TD method that uses elements of that class for function approximation. This paper focuses on performing evolutionary function approximation with neural networks. There are several reasons for this choice. First, they have great experimental value. Nonlinear function approximators are often the most challenging to use; hence, success for evolutionary function approximation with neural networks is good reason to hope for success with linear methods too. Second, neural networks have great potential, since they can represent value functions linear methods cannot (given the same basis functions). Finally, employing neural networks is feasible because they have previously succeeded as TD function approximators (Crites and Barto, 1998; Tesauro, 1994) and sophisticated methods for optimizing their representations (Gruau et al., 1996; Stanley and Miikkulainen, 2002) already exist.

This paper uses NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002) to select neural network function approximators for Q-learning (Watkins, 1989), a popular

TD method. The resulting algorithm, called NEAT+Q, uses NEAT to evolve topologies and initial weights of neural networks that are better able to learn, via backpropagation, to represent the value estimates provided by Q-learning.

Evolutionary computation is typically applied to *off-line* scenarios, where the only goal is to discover a good policy as quickly as possible. By contrast, TD methods are typically applied to *on-line* scenarios, in which the agent tries to learn a good policy quickly *and* to maximize the reward it obtains while doing so. Hence, for evolutionary function approximation to achieve its full potential, the underlying evolutionary method needs to work well on-line.

TD methods excel on-line because they are typically combined with action selection mechanisms like ε-greedy and softmax selection (Sutton and Barto, 1998). These mechanisms improve on-line performance by explicitly balancing two competing objectives: 1) searching for better policies (*exploration*) and 2) gathering as much reward as possible (*exploitation*). This paper investigates a novel approach we call *on-line evolutionary computation*, in which selection mechanisms commonly used by TD methods to choose individual actions are used in evolutionary computation to choose policies for evaluation. We present two implementations, based on ε-greedy and softmax selection, that distribute evaluations within a generation so as to favor more promising individuals. Since on-line evolutionary computation can be used in conjunction with evolutionary function approximation, the ability to optimize representations need not come at the expense of the on-line aspects of TD methods. On the contrary, the value function and its representation can be optimized simultaneously, all while the agent interacts with its environment.

We evaluate these contributions with extended empirical studies in two domains: 1) mountain car and 2) server job scheduling. The mountain car task (Sutton and Barto, 1998) is a canonical reinforcement learning benchmark domain that requires function approximation. Though the task is simple, previous researchers have noted that manually designed neural network function approximators are often unable to master it (Boyan and Moore, 1995; Pyeatt and Howe, 2001). Hence, this domain is ideal for a preliminary evaluation of NEAT+Q.

Server job scheduling (Whiteson and Stone, 2004), is a large, probabilistic reinforcement learning task from the field of *autonomic computing* (Kephart and Chess, 2003). In server job scheduling, a server, such as a website's application server or database, must determine in what order to process a queue of waiting jobs so as to maximize the system's aggregate utility. This domain is challenging because it is large (the size of both the state and action spaces grow in direct proportion to the size of the queue) and probabilistic (the server does not know what type of job will arrive next). Hence, it is a typical example of a reinforcement learning task that requires effective function approximation.

Using these domains, our experiments test Q-learning with a series of manually designed neural networks and compare the results to NEAT+Q and regular NEAT (which trains action selectors in lieu of value functions). The results demonstrate that evolutionary function approximation can significantly improve the performance of TD methods. Furthermore, we test NEAT and NEAT+Q with and without ε-greedy and softmax versions of evolutionary computation. These experiments confirm that such techniques can significantly improve the on-line performance of evolutionary computation. Finally, we present additional tests that measure the effect of continual learning on function approximators. The results offer insight into why certain methods outperform others in these domains and what factors can make neural network function approximation difficult in practice.

We view the impact of this work as two-fold. First, it provides a much-needed practical approach to selecting TD function approximators, automating a critical design step that is typically performed

manually. Second, it provides an objective analysis of the strengths and weaknesses of evolutionary and TD methods, opportunistically combining the strengths into a single approach. Though the TD and evolutionary communities are mostly disjoint and focus on somewhat different problems, we find that each can benefit from the progress of the other. On the one hand, we show that methods for evolving neural network topologies can find TD function approximators that perform better. On the other hand, we show that established techniques from the TD community can make evolutionary methods applicable to on-line learning problems.

The remainder of this paper is organized as follows. Section 2 provides background on Q-learning and NEAT, the constituent learning methods used in this paper. Section 3 introduces the novel methods and details the particular implementations we tested. Section 4 describes the mountain car and server job scheduling domains and Section 5 presents and discusses empirical results. Section 7 overviews related work, Section 8 outlines opportunities for future work, and Section 9 concludes.

2. Background

We begin by reviewing Q-learning and NEAT, the algorithms that form the building blocks of our implementations of evolutionary function approximation.

2.1 Q-Learning

There are several different TD methods currently in use, including Q-learning (Watkins, 1989), Sarsa (Sutton and Barto, 1998), and LSPI (Lagoudakis and Parr, 2003). The experiments presented in this paper use Q-learning because it is a well-established, canonical method that has also enjoyed empirical success, particularly when combined with neural network function approximators (Crites and Barto, 1998). We present it as a representative method but do not claim it is superior to other TD approaches. In principle, evolutionary function approximation can be used with any of them. For example, many of the experiments described in Section 5 have been replicated with Sarsa (Sutton and Barto, 1998), another popular TD method, in place of Q-learning, yielding qualitatively similar results.

Like many other TD methods, Q-learning attempts to learn a value function $Q(s,a)$ that maps state-action pairs to values. In the tabular case, the algorithm is defined by the following update rule, applied each time the agent transitions from state s to state s':

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))$$

where $\alpha \in [0,1]$ is a learning rate parameter, $\gamma \in [0,1]$ is a discount factor, and r is the immediate reward the agent receives upon taking action a.

Algorithm 1 describes the Q-learning algorithm when a neural network is used to approximate the value function. The inputs to the network describe the agent's current state; the outputs, one for each action, represent the agent's current estimate of the value of the associated state-action pairs. The initial weights of the network are drawn from a Gaussian distribution with mean 0.0 and standard deviation σ (line 5). The EVAL-NET function (line 9) returns the activation on the network's outputs after the given inputs are fed to the network and propagated forward. Since the network uses a sigmoid activation function, these values will all be in $[0,1]$ and hence are rescaled according to a parameter k. At each step, the weights of the neural network are adjusted (line 13) such that its

output better matches the current value estimate for the state-action pair: $r + \gamma \max_{a'} Q(s', a')$. The adjustments are made via the BACKPROP function, which implements the standard backpropagation algorithm (Rumelhart et al., 1986) with the addition of accumulating eligibility traces controlled by λ (Sutton and Barto, 1998). The agent uses ε-greedy selection (Sutton and Barto, 1998) to ensure it occasionally tests alternatives to its current policy (lines 10–11). The agent interacts with the environment via the TAKE-ACTION function (line 15), which returns a reward and a new state.

Algorithm 1 Q-LEARN($S, A, \sigma, c, \alpha, \gamma, \lambda, \varepsilon_{td}, e$)

```
1:  // S: set of all states, A: set of all actions, σ: standard deviation of initial weights
2:  // c: output scale, α: learning rate, γ: discount factor, λ: eligibility decay rate
3:  // ε_td: exploration rate, e: total number of episodes
4:
5:  N ← INIT-NET(S, A, σ)                          // make a new network N with random weights
6:  for i ← 1 to e do
7:      s, s' ← null, INIT-STATE(S)                // environment picks episode's initial state
8:      repeat
9:          Q[] ← c × EVAL-NET(N, s')              // compute value estimates for current state
10:         with-prob(ε_td) a' ← RANDOM(A)         // select random exploratory action
11:         else a' ← argmax_j Q[j]                // or select greedy action
12:         if s ≠ null then
13:             BACKPROP(N, s, a, (r + γ max_j Q[j])/c, α, γ, λ)   // adjust weights toward target
14:         s, a ← s', a'
15:         r, s' ← TAKE-ACTION(a')                // take action and transition to new state
16:     until TERMINAL-STATE?(s)
```

2.2 NEAT[1]

The implementation of evolutionary function approximation presented in this paper relies on NeuroEvolution of Augmenting Topologies (NEAT) to automate the search for appropriate topologies and initial weights of neural network function approximators. NEAT is an appropriate choice because of its empirical successes on difficult reinforcement learning tasks like non-Markovian double pole balancing (Stanley and Miikkulainen, 2002), game playing (Stanley and Miikkulainen, 2004b), and robot control (Stanley and Miikkulainen, 2004a), and because of its ability to automatically optimize network topologies.

In a typical neuroevolutionary system (Yao, 1999), the weights of a neural network are strung together to form an individual genome. A population of such genomes is then evolved by evaluating each one and selectively reproducing the fittest individuals through crossover and mutation. Most neuroevolutionary systems require the designer to manually determine the network's topology (i.e. how many hidden nodes there are and how they are connected). By contrast, NEAT automatically evolves the topology to fit the complexity of the problem. It combines the usual search for network weights with evolution of the network structure.

NEAT is an optimization technique that can be applied to a wide variety of problems. Section 3 below describes how we use NEAT to optimize the topology and initial weights of TD function

1. This section is adapted from the original NEAT paper (Stanley and Miikkulainen, 2002).

approximators. Here, we describe how NEAT can be used to tackle reinforcement learning problems without the aid of TD methods, an approach that serves as one baseline of comparison in Section 5. For this method, NEAT does not attempt to learn a value function. Instead, it finds good policies directly by training *action selectors*, which map states to the action the agent should take in that state. Hence it is an example of *policy search* reinforcement learning. Like other policy search methods, e.g. (Sutton et al., 2000; Ng and Jordan, 2000; Mannor et al., 2003; Kohl and Stone, 2004), it uses global optimization techniques to directly search the space of potential policies.

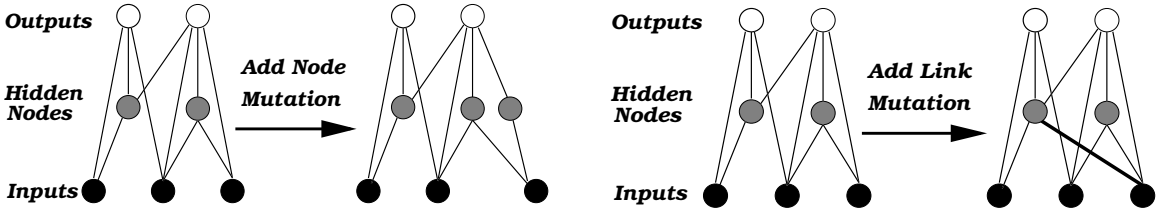
Algorithm 2 NEAT(S, A, p, m_n, m_l, g, e)

```
1:  // S: set of all states, A: set of all actions, p: population size, m_n: node mutation rate
2:  // m_l: link mutation rate, g: number of generations, e: episodes per generation
3:
4:  P[] ← INIT-POPULATION(S, A, p)          // create new population P with random networks
5:  for i ← 1 to g do
6:      for j ← 1 to e do
7:          N, s, s' ← RANDOM(P[]), null, INIT-STATE(S)          // select a network randomly
8:          repeat
9:              Q[] ← EVAL-NET(N, s')          // evaluate selected network on current state
10:             a' ← argmax_i Q[i]          // select action with highest activation
11:             s, a ← s', a'
12:             r, s' ← TAKE-ACTION(a')          // take action and transition to new state
13:             N.fitness ← N.fitness + r          // update total reward accrued by N
14:         until TERMINAL-STATE?(s)
15:         N.episodes ← N.episodes + 1          // update total number of episodes for N
16:     P'[] ← new array of size p          // new array will store next generation
17:     for j ← 1 to p do
18:         P'[j] ← BREED-NET(P[])          // make a new network based on fit parents in P
19:         with-probability m_n: ADD-NODE-MUTATION(P'[j]) // add a node to new network
20:         with-probability m_l: ADD-LINK-MUTATION(P'[j])          // add a link to new network
21:     P[] ← P'[]
```

Algorithm 2 contains a high-level description of the NEAT algorithm applied to an episodic reinforcement learning problem. This implementation differs slightly from previous versions of NEAT in that evaluations are conducted by randomly selecting individuals (line 7), instead of the more typical approach of stepping through the population in a fixed order. This change does not significantly alter NEAT's behavior but facilitates the alterations we introduce in Section 3.2. During each step, the agent takes whatever action corresponds to the output with the highest activation (lines 10–12). NEAT maintains a running total of the reward accrued by the network during its evaluation (line 13). Each generation ends after e episodes, at which point each network's average fitness is $N.fitness/N.episodes$. In stochastic domains, e typically must be much larger than $|P|$ to ensure accurate fitness estimates for each network. NEAT creates a new population by repeatedly calling the BREED-NET function (line 18), which performs crossover on two highly fit parents. The new resulting network can then undergo mutations that add nodes or links to its structure. (lines 19–20). The remainder of this section provides an overview of the reproductive process that occurs in lines 17–20. Stanley and Miikkulainen (2002) present a full description.

2.2.1 MINIMIZING DIMENSIONALITY

Unlike other systems that evolve network topologies and weights (Gruau et al., 1996; Yao, 1999) NEAT begins with a uniform population of simple networks with no hidden nodes and inputs connected directly to outputs. New structure is introduced incrementally via two special mutation operators. Figure 1 depicts these operators, which add new hidden nodes and links to the network. Only the structural mutations that yield performance advantages tend to survive evolution's selective pressure. In this way, NEAT tends to search through a minimal number of weight dimensions and find an appropriate complexity level for the problem.



(a) A mutation operator for adding new nodes

(b) A mutation operator for adding new links

Figure 1: Examples of NEAT's mutation operators for adding structure to networks. In (a), a hidden node is added by splitting a link in two. In (b), a link, shown with a thicker black line, is added to connect two nodes.

2.2.2 GENETIC ENCODING WITH HISTORICAL MARKINGS

Evolving network structure requires a flexible genetic encoding. Each genome in NEAT includes a list of *connection genes*, each of which refers to two *node genes* being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an *innovation number*, which allows NEAT to find corresponding genes during crossover.

In order to perform crossover, the system must be able to tell which genes match up between *any* individuals in the population. For this purpose, NEAT keeps track of the historical origin of every gene. Whenever a new gene appears (through structural mutation), a *global innovation number* is incremented and assigned to that gene. The innovation numbers thus represent a chronology of every gene in the system. Whenever these genomes crossover, innovation numbers on inherited genes are preserved. Thus, the historical origin of every gene in the system is known throughout evolution.

Through innovation numbers, the system knows exactly which genes match up with which. Genes that do not match are either *disjoint* or *excess*, depending on whether they occur within or outside the range of the other parent's innovation numbers. When crossing over, the genes in both genomes with the same innovation numbers are lined up. Genes that do not match are inherited from the more fit parent, or if they are equally fit, from both parents randomly.

Historical markings allow NEAT to perform crossover without expensive topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of matching different topologies (Radcliffe, 1993) is essentially avoided.

2.2.3 SPECIATION

In most cases, adding new structure to a network initially reduces its fitness. However, NEAT speciates the population, so that individuals compete primarily within their own niches rather than with the population at large. Hence, topological innovations are protected and have time to optimize their structure before competing with other niches in the population.

Historical markings make it possible for the system to divide the population into species based on topological similarity. The distance δ between two network encodings is a simple linear combination of the number of excess (E) and disjoint (D) genes, as well as the average weight differences of matching genes ($\overline{W}$):

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}$$

The coefficients c_1, c_2, and c_3 adjust the importance of the three factors, and the factor N, the number of genes in the larger genome, normalizes for genome size. Genomes are tested one at a time; if a genome's distance to a randomly chosen member of the species is less than δ_t, a compatibility threshold, it is placed into this species. Each genome is placed into the first species where this condition is satisfied, so that no genome is in more than one species.

The reproduction mechanism for NEAT is *explicit fitness sharing* (Goldberg and Richardson, 1987), where organisms in the same species must share the fitness of their niche, preventing any one species from taking over the population.

3. Method

This section describes evolutionary function approximation and a complete implementation called NEAT+Q. It also describes on-line evolutionary computation and details two ways of implementing it in NEAT+Q.

3.1 Evolutionary Function Approximation

When evolutionary methods are applied to reinforcement learning problems, they typically evolve a population of action selectors, each of which remains fixed during its fitness evaluation. The central insight behind evolutionary function approximation is that, if evolution is directed to evolve value functions instead, then those value functions can be updated, using TD methods, during each fitness evaluation. In this way, the system can *evolve* function approximators that are better able to *learn* via TD.

In addition to automating the search for effective representations, evolutionary function approximation can enable synergistic effects between evolution and learning. How these effects occur depends on which of two possible approaches is employed. The first possibility is a *Lamarckian* approach, in which the changes made by TD during a given generation are written back into the original genomes, which are then used to breed a new population. The second possibility is a *Darwinian* implementation, in which the changes made by TD are discarded and the new population is bred from the original genomes, as they were at birth.

It has long since been determined that biological systems are Darwinian, not Lamarckian. However, it remains unclear which approach is better computationally, despite substantial research (Pereira and Costa, 2001; D. Whitley, 1994; Yamasaki and Sekiguchi, 2000). The potential advantage of Lamarckian evolution is obvious: it prevents each generation from having to repeat the same learn-

ing. However, Darwinian evolution can be advantageous because it enables each generation to reproduce the genomes that led to success in the previous generation, rather than relying on altered versions that may not thrive under continued alteration. Furthermore, in a Darwinian system, the learning conducted by previous generations can be indirectly recorded in a population's genomes via a phenomenon called the *Baldwin Effect* (Baldwin, 1896), which has been demonstrated in evolutionary computation (Hinton and Nowlan, 1987; Ackley and Littman, 1991; Boers et al., 1995; Arita and Suzuki, 2000). The Baldwin Effect occurs in two stages. In the first stage, the learning performed by individuals during their lifetimes speeds evolution, because each individual does not have to be exactly right at birth; it need only be in the right neighborhood and learning can adjust it accordingly. In the second stage, those behaviors that were previously learned during individuals' lifetimes become known at birth. This stage occurs because individuals that possess adaptive behaviors at birth have higher overall fitness and are favored by evolution.

Hence, synergistic effects between evolution and learning are possible regardless of which implementation is used. In Section 5, we compare the two approaches empirically. The remainder of this section details NEAT+Q, the implementation of evolutionary function approximation used in our experiments.

3.1.1 NEAT+Q

All that is required to make NEAT optimize value functions instead of action selectors is a reinterpretation of its output values. The structure of neural network action selectors (one input for each state feature and one output for each action) is already identical to that of Q-learning function approximators. Therefore, if the weights of the networks NEAT evolves are updated during their fitness evaluations using Q-learning and backpropagation, they will effectively evolve value functions instead of action selectors. Hence, the outputs are no longer arbitrary values; they represent the long-term discounted values of the associated state-action pairs and are used, not just to select the most desirable action, but to update the estimates of other state-action pairs.

Algorithm 3 summarizes the resulting NEAT+Q method. Note that this algorithm is identical to Algorithm 2, except for the delineated section containing lines 13–16. Each time the agent takes an action, the network is backpropagated towards Q-learning targets (line 16) and ε-greedy selection occurs just as in Algorithm 1 (lines 13–14). If α and ε_{td} are set to zero, this method degenerates to regular NEAT.

NEAT+Q combines the power of TD methods with the ability of NEAT to learn effective representations. Traditional neural network function approximators put all their eggs in one basket by relying on a single manually designed network to represent the value function. NEAT+Q, by contrast, explores the space of such networks to increase the chance of finding a representation that will perform well.

In NEAT+Q, the weight changes caused by backpropagation accumulate in the current population's networks throughout each generation. When a network is selected for an episode, its weights begin exactly as they were at the end of its last episode. In the Lamarckian approach, those changes are copied back into the networks' genomes and inherited by their offspring. In the Darwinian approach, those changes are discarded at the end of each generation.

Algorithm 3 NEAT+Q($S, A, c, p, m_n, m_l, g, e, \alpha, \gamma, \lambda, \varepsilon_{td}$)

// S: set of all states, A: set of all actions, c: output scale, p: population size
// m_n: node mutation rate, m_l: link mutation rate, g: number of generations
// e: number of episodes per generation, α: learning rate, γ: discount factor
// λ: eligibility decay rate, ε_{td}: exploration rate

$P[] \leftarrow$ INIT-POPULATION(S, A, p) *// create new population P with random networks*
for $i \leftarrow 1$ to g **do**
  for $j \leftarrow 1$ to e **do**
    $N, s, s' \leftarrow$ RANDOM($P[]$), null, INIT-STATE(S) *// select a network randomly*
    repeat
      $Q[] \leftarrow c \times$ EVAL-NET(N, s') *// compute value estimates for current state*
      ---
      with-prob(ε_{td}) $a' \leftarrow$ RANDOM(A) *// select random exploratory action*
      else $a' \leftarrow \text{argmax}_k Q[k]$ *// or select greedy action*
      if $s \neq$ null **then**
        BACKPROP($N, s, a, (r + \gamma \max_k Q[k])/c, \alpha, \gamma, \lambda$) *// adjust weights toward target*

      $s, a \leftarrow s', a'$
      $r, s' \leftarrow$ TAKE-ACTION(a') *// take action and transition to new state*
      $N.fitness \leftarrow N.fitness + r$ *// update total reward accrued by N*
    until TERMINAL-STATE?(s)
    $N.episodes \leftarrow N.episodes + 1$ *// update total number of episodes for N*
  $P'[] \leftarrow$ new array of size p *// new array will store next generation*
  for $j \leftarrow 1$ to p **do**
    $P'[j] \leftarrow$ BREED-NET($P[]$) *// make a new network based on fit parents in P*
    with-probability m_n: ADD-NODE-MUTATION($P'[j]$) *// add a node to new network*
    with-probability m_l: ADD-LINK-MUTATION($P'[j]$) *// add a link to new network*
  $P[] \leftarrow P'[]$

3.2 On-Line Evolutionary Computation

To excel in on-line scenarios, a learning algorithm must effectively balance two competing objectives. The first objective is exploration, in which the agent tries alternatives to its current best policy in the hopes of improving it. The second objective is exploitation, in which the agent follows the current best policy in order to maximize the reward it receives. TD methods excel at on-line tasks because they are typically combined with action selection mechanisms that achieve this balance (e.g ε-greedy and softmax selection).

Evolutionary methods, though lacking explicit selection mechanisms, do implicitly perform this balance. In fact, in one of the earliest works on evolutionary computation, Holland (1975) argues that the reproduction mechanism encourages exploration, since crossover and mutation result in novel genomes, but also encourages exploitation, since each new generation is based on the fittest members of the last one. However, reproduction allows evolutionary methods to balance exploration and exploitation only *across* generations, not *within* them. Once the members of each generation have been determined, they all typically receive the same evaluation time, even if some individuals dramatically outperform others in early episodes. Hence, within a generation, a typical evolutionary method is purely exploratory, as it makes no effort to favor those individuals that have performed well so far.

Therefore, to excel on-line, evolutionary methods need a way to limit the exploration that occurs within each generation and force more exploitation. In a sense, this problem is the opposite of that faced by TD methods, which naturally exploit (by following the greedy policy) and thus need a way to force more exploration. Nonetheless, the ultimate goal is the same: a proper balance between the two extremes. Hence, we propose that the solution can be the same too. In this section, we discuss ways of borrowing the action selection mechanisms traditionally used in TD methods and applying them in evolutionary computation.

To do so, we must modify the level at which selection is performed. Evolutionary algorithms cannot perform selection at the level of individual actions because, lacking value functions, they have no notion of the value of individual actions. However, they can perform selection at the level of evaluations, in which entire policies are assessed holistically. The same selection mechanisms used to choose individual actions in TD methods can be used to select policies for evaluation, an approach we call on-line evolutionary computation. Using this technique, evolutionary algorithms can excel on-line by balancing exploration and exploitation within *and* across generations.

The remainder of this section presents two implementations. The first, which relies on ε-greedy selection, switches probabilistically between searching for better policies and re-evaluating the best known policy to garner maximal reward. The second, which relies on softmax selection, distributes evaluations in proportion to each individual's estimated fitness, thereby focusing on the most promising individuals and increasing the average reward accrued.

3.2.1 USING ε-GREEDY SELECTION IN EVOLUTIONARY COMPUTATION

When ε-greedy selection is used in TD methods, a single parameter, ε_{td}, is used to control what fraction of the time the agent deviates from greedy behavior. Each time the agent selects an action, it chooses probabilistically between exploration and exploitation. With probability ε_{td}, it will explore by selecting randomly from the available actions. With probability $1-\varepsilon_{td}$, it will exploit by selecting the greedy action.

In evolutionary computation, this same mechanism can be used to determine which policies to evaluate within each generation. With probability ε_{ec}, the algorithm explores by behaving exactly as it would normally: selecting a policy for evaluation, either randomly or by iterating through the population. With probability $1-\varepsilon_{ec}$, the algorithm exploits by selecting the best policy discovered so far in the current generation. The score of each policy is just the average reward per episode it has received so far. Each time a policy is selected for evaluation, the total reward it receives is incorporated into that average, which can cause it to gain or lose the rank of best policy.

To apply ε-greedy selection to NEAT and NEAT+Q, we need only alter the assignment of the candidate policy N in lines 7 and 9 of Algorithms 2 and 3, respectively. Instead of a random selection, we use the result of the ε-greedy selection function described in Algorithm 4, where $N.average = N.fitness/N.episodes$. In the case of NEAT+Q, two different ε parameters control exploration throughout the system: ε_{td} controls the exploration that helps Q-learning estimate the value function and ε_{ec} controls exploration that helps NEAT discover appropriate topologies and initial weights for the neural network function approximators.

Algorithm 4 ε-GREEDY SELECTION(P, ε_{ec})

1: *// P: population,* ε_{ec}*: NEAT's exploration rate*
2:
3: **with-prob**(ε_{ec}) return RANDOM(P) *// select random network*
4: **else** return $N \in P \mid \forall (N' \in P)\, N.average \geq N'.average$ *// or select champion*

Using ε-greedy selection in evolutionary computation allows it to thrive in on-line scenarios by balancing exploration and exploitation. For the most part, this method does not alter evolution's search but simply interleaves it with exploitative episodes that increase average reward during learning. The next section describes how softmax selection can be applied to evolutionary computation to intelligently focus search with each generation and create a more nuanced balance between exploration and exploitation.

3.2.2 USING SOFTMAX SELECTION IN EVOLUTIONARY COMPUTATION

When softmax selection is used in TD methods, an action's probability of selection is a function of its estimated value. In addition to ensuring that the greedy action is chosen most often, this technique focuses exploration on the most promising alternatives. There are many ways to implement softmax selection but one popular method relies on a Boltzmann distribution (Sutton and Barto, 1998), in which case an agent in state s chooses an action a with probability

$$\frac{e^{Q(s,a)/\tau}}{\sum_{b \in A} e^{Q(s,b)/\tau}}$$

where A is the set of available actions, $Q(s,a)$ is the agent's value estimate for the given state-action pair and τ is a positive parameter controlling the degree to which actions with higher values are favored in selection. The higher the value of τ, the more equiprobable the actions are.

As with ε-greedy selection, we use softmax selection in evolutionary computation to select policies for evaluation. At the beginning of each generation, each individual is evaluated for one episode, to initialize its fitness. Then, the remaining $e - |P|$ episodes are allocated according to a Boltzmann distribution. Before each episode, a policy p in a population P is selected with probabil-

ity

$$\frac{e^{S(p)/\tau}}{\sum_{q \in P} e^{S(q)/\tau}}$$

where $S(p)$ is the average fitness of the policy p.

To apply softmax selection to NEAT and NEAT+Q, we need only alter the assignment of the candidate policy N in lines 7 and 9 of Algorithms 2 and 3, respectively. Instead of a random selection, we use the result of the softmax selection function shown in Algorithm 5. In the case of NEAT+Q, ε_{td} controls Q-learning's exploration and τ controls NEAT's exploration. Of course, softmax exploration could be used within Q-learning too. However, since comparing different selection mechanisms for TD methods is not the subject of our research, in this paper we use only ε-greedy selection with TD methods.

Algorithm 5 SOFTMAX SELECTION(P, τ)

```
1: // P: population, τ: softmax temperature
2:
3: if ∃N ∈ P | N.episodes = 0 then
4:    return N                          // give each network one episode before using softmax
5: else
6:    total ← Σ_{N∈P} e^{N.average/τ}   // compute denominator of Boltzmann function
7:    for all N ∈ P do
8:       with-prob(e^{N.average/τ} / total) return N   // select N for evaluation
9:       else total ← total − e^{N.average/τ}          // or skip N and reweight probabilities
```

In addition to providing a more nuanced balance between exploration and exploitation, softmax selection also allows evolutionary computation to more effectively focus its search within each generation. Instead of spending the same number of evaluations on each member of the population, softmax selection can quickly abandon poorly performing policies and spend more episodes evaluating the most promising individuals.

In summary, on-line evolutionary computation enables the use of evolutionary computation during an agent's interaction with the world. Therefore, the ability of evolutionary function approximation to optimize representations need not come at the expense of the on-line aspects of TD methods. On the contrary, the value function and its representation can be optimized simultaneously, all while the agent interacts with its environment.

4. Experimental Setup

To empirically compare the methods described above, we used two different reinforcement learning domains. The first domain, mountain car, is a standard benchmark task requiring function approximation. We use this domain to establish preliminary, proof-of-concept results for the novel methods described in this paper. The second domain, server job scheduling, is a large, probabilistic domain drawn from the field of autonomic computing. We use this domain to assess whether these new methods can scale to a much more complex task. The remainder of this section details each of these domains and describes our approach to solving them with reinforcement learning.

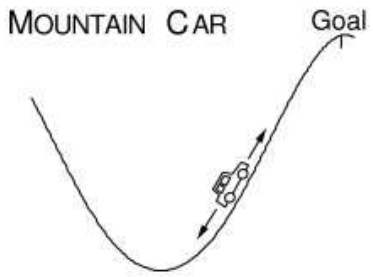


Figure 2: The Mountain Car Task. This figure was taken from Sutton and Barto (1998).

4.1 Mountain Car

In the mountain car task (Boyan and Moore, 1995), depicted in Figure 2, an agent strives to drive a car to the top of a steep mountain. The car cannot simply accelerate forward because its engine is not powerful enough to overcome gravity. Instead, the agent must learn to drive backwards up the hill behind it, thus building up sufficient inertia to ascend to the goal before running out of speed.

The agent's state at timestep t consists of its current position p_t and its current velocity v_t. It receives a reward of -1 at each time step until reaching the goal, at which point the episode terminates. The agent's three available actions correspond to the throttle settings 1,0, and -1. The following equations control the car's movement:

$$p_{t+1} = bound_p(p_t + v_{t+1})$$

$$v_{t+1} = bound_v(v_t + 0.001a_t - 0.0025cos(3p_t))$$

where a_t is the action the agent takes at timestep t, $bound_p$ enforces $-1.2 \leq p_{t+1} \leq 0.5$, and $bound_v$ enforces $-0.07 \leq v_{t+1} \leq 0.07$. In each episode, the agent begins in a state chosen randomly from these ranges. To prevent episodes from running indefinitely, each episode is terminated after 2,500 steps if the agent still has not reached the goal.

Though the agent's state has only two features, they are continuous and hence learning the value function requires a function approximator. Previous research has demonstrated that TD methods can solve the mountain car task using several different function approximators, including CMACs (Sutton, 1996; Kretchmar and Anderson, 1997), locally weighted regression (Boyan and Moore, 1995), decision trees (Pyeatt and Howe, 2001), radial basis functions (Kretchmar and Anderson, 1997), and instance-based methods (Boyan and Moore, 1995). By giving the learner *a priori* knowledge about the goal state and using methods based on experience replay, the mountain car problem has been solved with neural networks too (Reidmiller, 2005). However, the task remains notoriously difficult for neural networks, as several researchers have noted that value estimates can easily diverge (Boyan and Moore, 1995; Pyeatt and Howe, 2001).

We hypothesized that the difficulty of using neural networks in this task is due at least in part to the problem of finding an appropriate representation. Hence, as a preliminary evaluation of evolutionary function approximation, we applied NEAT+Q to the mountain car task to see if it could learn better than manually designed networks. The results are presented in Section 5.

To represent the agent's current state to the network, we divided each state feature into ten regions. One input was associated with each region (for a total of twenty inputs) and was set to one

if the agent's current state fell in that region, and to zero otherwise. Hence, only two inputs were activated for any given state. The networks have three outputs, each corresponding to one of the actions available to the agent.

4.2 Server Job Scheduling

While the mountain car task is a useful benchmark, it is a very simple domain. To assess whether our methods can scale to a much more complex problem, we use a challenging reinforcement learning task called server job scheduling. This domain is drawn from the burgeoning field of autonomic computing (Kephart and Chess, 2003). The goal of autonomic computing is to develop computer systems that automatically configure themselves, optimize their own behavior, and diagnose and repair their own failures. The demand for such features is growing rapidly, since computer systems are becoming so complex that maintaining them with human support staff is increasingly infeasible.

The vision of autonomic computing poses new challenges to many areas of computer science, including architecture, operating systems, security, and human-computer interfaces. However, the burden on artificial intelligence is especially great, since intelligence is a prerequisite for self-managing systems. In particular, we believe machine learning will play a primary role, since computer systems must be adaptive if they are to perform well autonomously. There are many ways to apply supervised methods to autonomic systems, e.g. for intrusion detection (Ertoz et al., 2004), spam filtering (Dalvi et al., 2004), or system configuration (Wildstrom et al., 2005). However, there are also many tasks where no human expert is available and reinforcement learning is applicable, e.g network routing (Boyan and Littman, 1994), job scheduling (Whiteson and Stone, 2004), and cache allocation (Gomez et al., 2001).

One such task is server job scheduling, in which a server, such as a website's application server or database, must determine in what order to process the jobs currently waiting in its queue. Its goal is to maximize the aggregate utility of all the jobs it processes. A *utility function* (not to be confused with a TD value function) for each job type maps the job's completion time to the utility derived by the user (Walsh et al., 2004). The problem of server job scheduling becomes challenging when these utility functions are nonlinear and/or the server must process multiple types of jobs. Since selecting a particular job for processing necessarily delays the completion of all other jobs in the queue, the scheduler must weigh difficult trade-offs to maximize aggregate utility. Also, this domain is challenging because it is large (the size of both the state and action spaces grow in direct proportion to the size of the queue) and probabilistic (the server does not know what type of job will arrive next). Hence, it is a typical example of a reinforcement learning task that requires effective function approximation.

The server job scheduling task is quite different from traditional scheduling tasks (Zhang and Dietterich, 1995; Zweben and Fox, 1998). In the latter case, there are typically multiple resources available and each job has a partially ordered list of resource requirements. Server job scheduling is simpler because there is only one resource (the server) and all jobs are independent of each other. However, it is more complex in that performance is measured via arbitrary utility functions, whereas traditional scheduling tasks aim solely to minimize completion times.

Our experiments were conducted in a Java-based simulator. The simulation begins with 100 jobs preloaded into the server's queue and ends when the queue becomes empty. During each timestep, the server removes one job from its queue and completes it. During each of the first 100 timesteps, a new job of a randomly selected type is added to the end of the queue. Hence, the agent must make

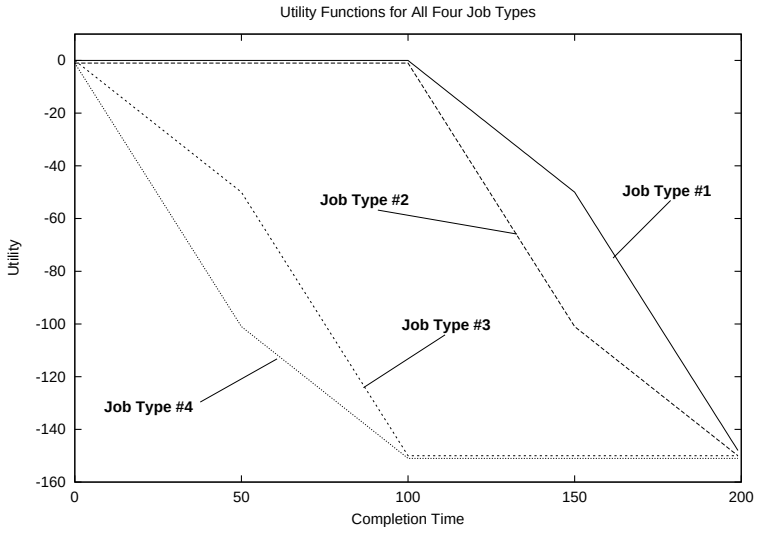


Figure 3: The four utility functions used in our experiments.

decisions about which job to process next even as new jobs are arriving. Since one job is processed at each timestep, each episode lasts 200 timesteps. For each job that completes, the scheduling agent receives an immediate reward determined by that job's utility function.

Four different job types were used in our experiments. Hence, the task can generate 4^{200} unique episodes. Utility functions for the four job types are shown in Figure 3. Users who create jobs of type #1 or #2 do not care about their jobs' completion times so long as they are less than 100 timesteps. Beyond that, they get increasingly unhappy. The rate of this change differs between the two types and switches at timestep 150. Users who create jobs of type #3 or #4 want their jobs completed as quickly as possible. However, once the job becomes 100 timesteps old, it is too late to be useful and they become indifferent to it. As with the first two job types, the slopes for job types #3 and #4 differ from each other and switch, this time at timestep 50. Note that all these utilities are negative functions of completion time. Hence, the scheduling agent strives to bring aggregate utility as close to zero as possible.

A primary obstacle to applying reinforcement learning methods to this domain is the size of the state and action spaces. A complete state description includes the type and age of each job in the queue. The scheduler's actions consist of selecting jobs for processing; hence a complete action space includes every job in the queue. To render these spaces more manageable, we discretize them. The range of job ages from 0 to 200 is divided into four sections and the scheduler is told, at each timestep, how many jobs in the queue of each type fall in each range, resulting in 16 state features. The action space is similarly discretized. Instead of selecting a particular job for processing, the scheduler specifies what type of job it wants to process and which of the four age ranges that job should lie in, resulting in 16 distinct actions. The server processes the youngest job in the queue that matches the type and age range specified by the action.

These discretizations mean the agent has less information about the contents of the job queue. However, its state is still sufficiently detailed to allow effective learning. Although the utility functions can change dramatically within each age range, their slopes do not change. It is the slope

of the utility function, not the utility function itself, which determines how much utility is lost by delaying a given job.

Even after discretization, the state space is quite large. If the queue holds at most q_{max} jobs, $\binom{q_{max}+1}{16}$ is a loose upper bound on the number of states, since each job can be in one of 16 buckets. Some of these states will not occur (e.g. ones where all the jobs in the queue are in the youngest age range). Nonetheless, with 16 actions per state, it is clearly infeasible to represent the value function in a table. Hence, success in this domain requires function approximation, as addressed in the following section.

5. Results

We conducted a series of experiments in the mountain car and server job scheduling domains to empirically evaluate the methods presented in this paper. Section 5.1 compares manual and evolutionary function approximators. Section 5.2 compares off-line and on-line evolutionary computation. Section 5.3 tests evolutionary function approximation combined with on-line evolutionary computation. Section 5.4 compares these novel approaches to previous learning and non-learning methods. Section 5.5 compares Darwinian and Lamarckian versions of evolutionary function approximation. Finally, Section 5.6 presents some addition tests that measure the effect of continual learning on function approximators. The results offer insight into why certain methods outperform others in these domains and what factors can make neural network function approximation difficult in practice.

Each of the graphs presented in these sections include error bars indicating 95% confidence intervals. In addition, to assess statistical significance, we conducted Student's t-tests on each pair of methods evaluated. The results of these tests are summarized in Appendix A.

5.1 Comparing Manual and Evolutionary Function Approximation

As an initial baseline, we conducted, in each domain, 25 runs in which NEAT attempts to discover a good policy using the setup described in Section 4. In these runs, the population size p was 100, the number of generations g was 100, the node mutation rate m_n was 0.02, the link mutation rate m_l was 0.1, and the number of episodes per generation e was 10,000. Hence, each individual was evaluated for 100 episodes on average. See Appendix B for more details on the NEAT parameters used in our experiments.

Next, we performed 25 runs in each domain using NEAT+Q, with the same parameter settings. The eligibility decay rate λ was 0.0. and the learning rate α was set to 0.1 and annealed linearly for each member of the population until reaching zero after 100 episodes.[2] In scheduling, γ was 0.95 and ε_{td} was 0.05. Those values of γ and ε_{td} work well in mountain car too, though in the experiments presented here they were set to 1.0 and 0.0 respectively, since Sutton (1996) found that discounting and exploration are unnecessary in mountain car. The output scale c was set to -100 in mountain car and -1000 in scheduling.

We tested both Darwinian and Lamarckian NEAT+Q in this manner. Both perform well, though which is preferable appears to be domain dependent. For simplicity, in this section and those that follow, we present results only for Darwinian NEAT+Q. In Section 5.5 we present a comparison of the two approaches.

2. Other values of λ were tested in the context of NEAT+Q but had little effect on performance.

To test Q-learning without NEAT, we tried 24 different configurations in each domain. These configurations correspond to every possible combination of the following parameter settings. The networks had feed-forward topologies with 0, 4, or 8 hidden nodes. The learning rate α was either 0.01 or 0.001. The annealing schedules for α were linear, decaying to zero after either 100,000 or 250,000 episodes. The eligibility decay rate λ was either 0.0 or 0.6. The other parameters, γ and ϵ, were set just as with NEAT+Q, and the standard deviation of initial weights σ was 0.1. Each of these 24 configurations was evaluated for 5 runs. In addition, we experimented informally with higher and lower values of α, higher values of γ, slower linear annealing, exponential annealing, and no annealing at all, though none performed as well as the results presented here.

In these experiments, each run used a different set of initial weights. Hence, the resulting performance of each configuration, by averaging over different initial weight settings, does not account for the possibility that some weight settings perform consistently better than others. To address this, for each domain, we took the best performing configuration[3] and randomly selected five fixed initial weight settings. For each setting, we conducted 5 additional runs. Finally, we took the setting with the highest performance and conducted an additional 20 runs, for a total of 25. For simplicity, the graphs that follow show only this Q-learning result: the best configuration with the best initial weight setting.

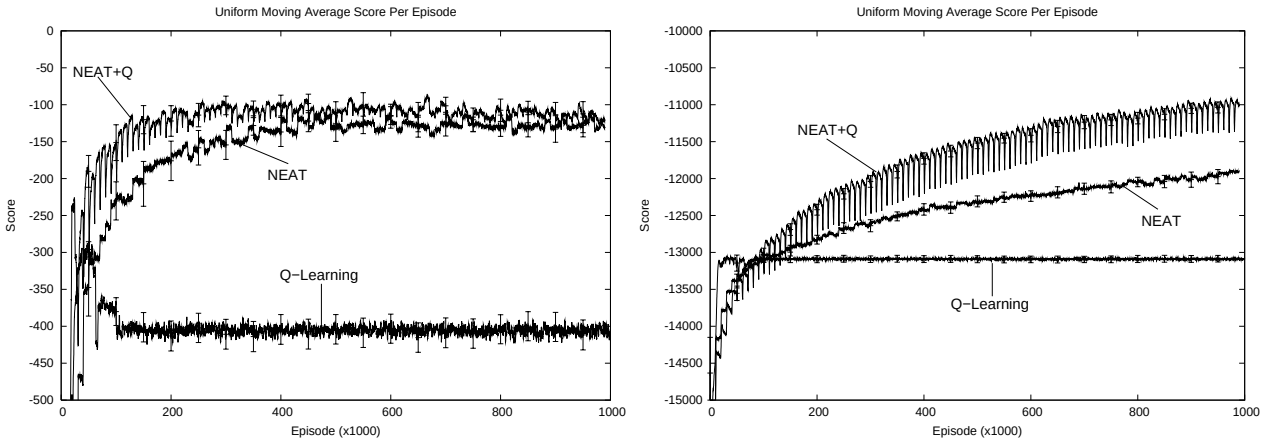
Figure 4 shows the results of these experiments. For each method, the corresponding line in the graph represents a uniform moving average over the aggregate utility received in the past 1,000 episodes, averaged over all 25 runs. Using average performance, as we do throughout this paper, is somewhat unorthodox for evolutionary methods, which are more commonly evaluated on the performance of the generation champion. There are two reasons why we adopt average performance. First, it creates a consistent metric for all the methods tested, including the TD methods that do not use evolutionary computation and hence have no generation champions. Second, it is an on-line metric because it incorporates *all* the reward the learning system accrues. Plotting only generation champions is an implicitly off-line metric because it does not penalize methods that discover good policies but fail to accrue much reward while learning. Hence, average reward is a better metric for evaluating on-line evolutionary computation, as we do in Section 5.2.

To make a larger number of runs computationally feasible, both NEAT and NEAT+Q were run for only 100 generations. In the scheduling domain, neither method has completely plateaued by this point. However, a handful of trials conducted for 200 generations verified that only very small additional improvements are made after 100 generation, without a qualitative effect on the results.

Note that the progress of NEAT+Q consists of a series of 10,000-episode intervals. Each of these intervals corresponds to one generation and the changes within them are due to learning via Q-learning and backpropagation. Although each individual learns for only 100 episodes on average, NEAT's system of randomly selecting individuals for evaluation causes that learning to be spread across the entire generation: each individual changes gradually during the generation as it is repeatedly evaluated. The result is a series of intra-generational learning curves within the larger learning curve.

For the particular problems we tested and network configurations we tried, evolutionary function approximation significantly improves performance over manually designed networks. In the scheduling domain, Q-learning learns much more rapidly in the very early part of learning. In both domains, however, Q-learning soon plateaus while NEAT and NEAT+Q continue to improve. Of

3. Mountain car parameters were: 4 hidden nodes, $\alpha = 0.001$, annealed to zero at episode 100,000, $\lambda = 0.0$. Server job scheduling parameters were: 4 hidden nodes, $\alpha = 0.01$, annealed to zero at episode 100,000, $\lambda = 0.6$.



(a) Mountain Car (b) Server Job Scheduling

Figure 4: A comparison of the performance of manual and evolutionary function approximators in the mountain car and server job scheduling domains.

course, after 100,000 episodes, Q-learning's learning rate α has annealed to zero and no additional learning is possible. However, its performance plateaus well before α reaches zero and, in our experiments, running Q-learning with slower annealing or no annealing at all consistently led to inferior and unstable performance.

Nonetheless, the possibility remains that additional engineering of the network structure, the feature set, or the learning parameters would significantly improve Q-learning's performance. In particular, when Q-learning is started with one of the best networks discovered by NEAT+Q and the learning rate is annealed aggressively, Q-learning matches NEAT+Q's performance without directly using evolutionary computation. However, it is unlikely that a manual search, no matter how extensive, would discover these successful topologies, which contain irregular and partially connected hidden layers. Figure 5 shows examples of typical networks evolved by NEAT+Q.

NEAT+Q also significantly outperforms regular NEAT in both domains. In the mountain car domain, NEAT+Q learns faster, achieving better performance in earlier generations, though both plateau at approximately the same level. In the server job scheduling domain, NEAT+Q learns more rapidly and also converges to significantly higher performance. This result highlights the value of TD methods on challenging reinforcement learning problems. Even when NEAT is employed to find effective representations, the best performance is achieved only when TD methods are used to estimate a value function. Hence, the relatively poor performance of Q-learning is not due to some weakness in the TD methodology but merely to the failure to find a good representation.

Furthermore, in the scheduling domain, the advantage of NEAT+Q over NEAT is not directly explained just by the learning that occurs via backpropagation within each generation. After 300,000 episodes, NEAT+Q clearly performs better even at the beginning of each generation, before such learning has occurred. Just as predicted by the Baldwin Effect, evolution proceeds more quickly in NEAT+Q because the weight changes made by backpropagation, in addition to improving that individual's performance, alter selective pressures and more rapidly guide evolution to useful regions of the search space.

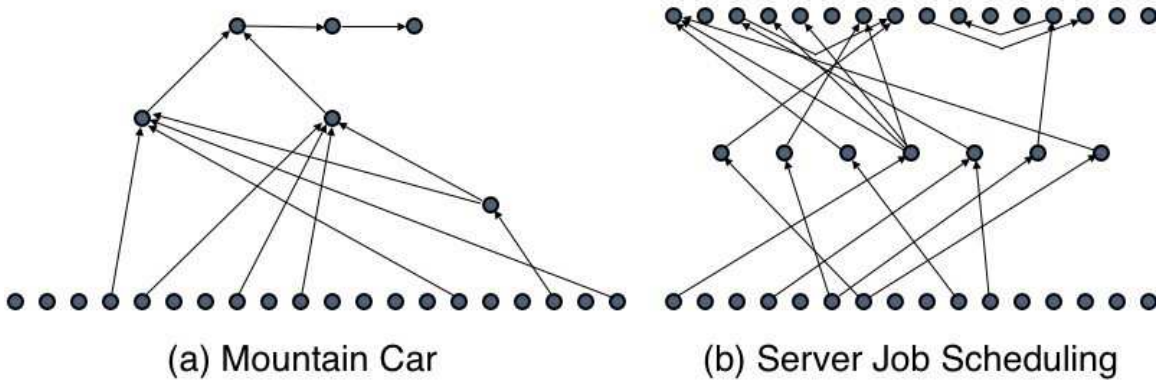


Figure 5: Typical examples of the topologies of the best networks evolved by NEAT+Q in both the mountain car and scheduling domains. Input nodes are on the bottom, hidden nodes in the middle, and output nodes on top. In addition to the links shown, each input node is directly connected to each output node. Note that two output nodes can be directly connected, in which case the activation of one node serves not only as an output of the network, but as an input to the other node.

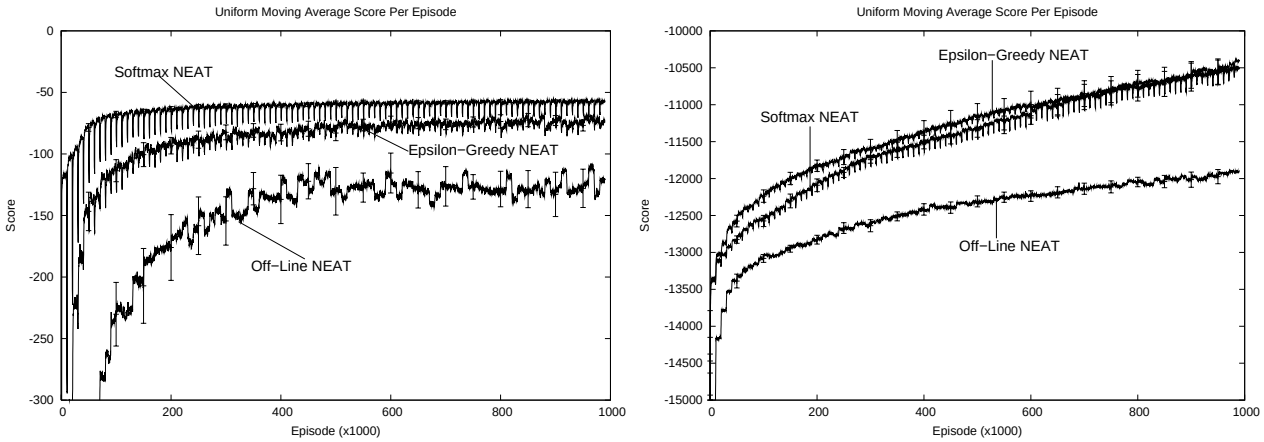
5.2 Comparing Off-Line and On-Line Evolutionary Computation

In this section, we present experiments evaluating on-line evolutionary computation. Since on-line evolutionary computation does not depend on evolutionary function approximation, we first test it using regular NEAT, by comparing an off-line version to on-line versions using ϵ-greedy and softmax selection. In Section 5.3 we study the effect of combining NEAT+Q with on-line evolutionary computation.

Figure 6 compares the performance of off-line NEAT to its on-line counterparts in both domains. The results for off-line NEAT are the same as those presented in Figure 4. To test on-line NEAT with ϵ-greedy selection, 25 runs were conducted with ϵ_{ec} set to 0.25. This value is larger than is typically used in TD methods but makes intuitive sense, since exploration in NEAT is safer than in TD methods. After all, even when NEAT explores, the policies it selects are not drawn randomly from policy space. On the contrary, they are the children of the previous generation's fittest parents. To test on-line NEAT with softmax selection, 25 runs were conducted with τ set to 50 in mountain car and 500 in the scheduling domain. These values are different because a good value of τ depends on the range of possible values, which differ dramatically between the two domains.

These results demonstrate that both versions of on-line evolutionary computation can significantly improve NEAT's average performance. In addition, in mountain car, on-line evolutionary computation with softmax selection boosts performance even more than ϵ-greedy selection.

Given the way these two methods work, the advantage of softmax over ϵ-greedy in mountain car is not surprising. ϵ-greedy selection is a rather naïve approach because it treats all exploratory actions equally, with no attempt to favor the most promising ones. For the most part, it conducts the search for better policies in the same way as off-line evolutionary computation; it simply interleaves that search with exploitative episodes that employ the best known policy. Softmax selection, by contrast, concentrates exploration on the most promising alternatives and hence alters the way the



(a) Mountain Car (b) Server Job Scheduling

Figure 6: A comparison of the performance off-line and on-line evolutionary computation in the mountain car and server job scheduling domains.

search for better policies is conducted. Unlike ε-greedy exploration, softmax selection spends fewer episodes on poorly performing individuals and more on those with the most promise. In this way, it achieves better performance.

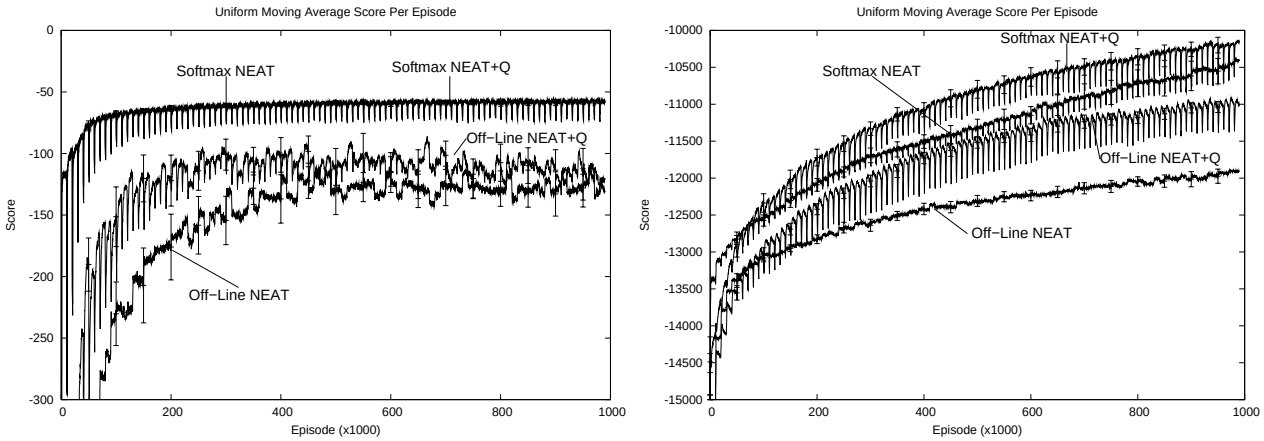
More surprising is that this effect is not replicated in the scheduling domain. Both on-line methods perform significantly better than their off-line counterpart but softmax performs only as well as ε-greedy. It is possible that softmax, though focusing exploration more intelligently, exploits less aggressively than ε-greedy, which gives so many evaluations to the champion. It is also possible that some other setting of τ would make softmax outperform ε-greedy, though our informal parameter search did not uncover one. Even achieving the performance shown here required using different values of τ in the two domains, whereas the same value of ε worked in both cases. This highlights one disadvantage of using softmax selection: the difficulty of choosing τ. As Sutton and Barto write "Most people find it easier to set the ε parameter with confidence; setting τ requires knowledge of the likely action values and of powers of *e*." (Sutton and Barto, 1998, pages 27-30)

It is interesting that the intra-generational learning curves characteristic of NEAT+Q appear in the on-line methods even though backpropagation is not used. The average performance increases during each generation without the help of TD methods because the system becomes better informed about which individuals to select on exploitative episodes. Hence, on-line evolutionary computation can be thought of as another way of combining evolution and learning. In each generation, the system learns which members of the population are strongest and uses that knowledge to boost average performance.

5.3 Combining Evolutionary Function Approximation with On-Line Evolutionary Computation

Sections 5.1 and 5.2 verify that both evolutionary function approximation and on-line evolutionary computation can significantly boost performance in reinforcement learning tasks. In this section, we present experiments that assess how well these two ideas work together.

Figure 7 presents the results of combining NEAT+Q with softmax evolutionary computation, averaged over 25 runs, and compares it to using each of these methods individually, i.e. using off-line NEAT+Q (as done in Section 5.1) and using softmax evolutionary computation with regular NEAT (as done in Section 5.2). For the sake of simplicity we do not present results for ε-greedy NEAT+Q though we tested it and found that it performed similarly to softmax NEAT+Q.



(a) Mountain Car (b) Server Job Scheduling

Figure 7: The performance of combining evolutionary function approximation with on-line evolutionary computation compared to using each individually in the mountain car and server job scheduling domains.

In both domains, softmax NEAT+Q performs significantly better than off-line NEAT+Q. Hence, just like regular evolutionary computation, evolutionary function approximation performs better when supplemented with selection techniques traditionally used in TD methods. Surprisingly, in the mountain car domain, softmax NEAT+Q performs only as well softmax NEAT. We attribute these results to a ceiling effect, i.e. the mountain car domain is easy enough that, given an appropriate selection mechanism, NEAT is able to learn quite rapidly, even without the help of Q-learning. In the server job scheduling domain, softmax NEAT+Q does perform better than softmax NEAT, though the difference is rather modest. Hence, in both domains, the most critical factor to boosting the performance of evolutionary computation is the use of an appropriate selection mechanism.

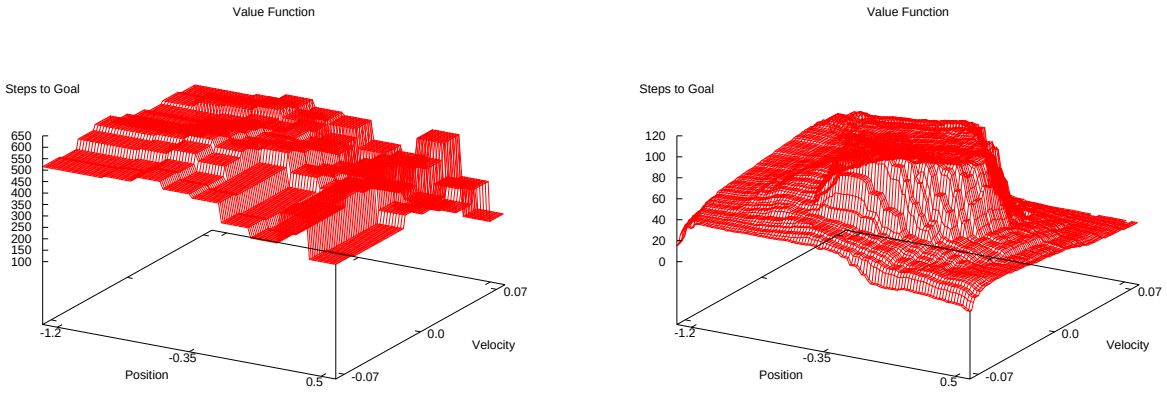
5.4 Comparing to Previous Approaches

The experiments presented thus far verify that the novel methods presented in this paper can improve performance over the constituent techniques upon which they are built. In this section, we present experiments that compare the performance of the highest performing novel method, softmax NEAT+Q, to previous approaches. In the mountain car domain, we compare to previous results that use TD methods with a linear function approximator (Sutton, 1996). In the server job scheduling domain, we compare to a random scheduler, two non-learning schedulers from previous research (van Mieghem, 1995; Whiteson and Stone, 2004), and an analytical solution computed using integer linear programming.

In the mountain car domain, the results presented above make clear that softmax NEAT+Q can rapidly learn a good policy. However, since these results use an on-line metric, performance is averaged over all members of the population. Hence, they do not reveal how close the best learned policies are to optimal. To assess this, we selected the generation champion from the final generation of each softmax NEAT+Q run and evaluated it for an additional 1,000 episodes. Then we compared this to the performance of a learner using Sarsa, a TD method similar to Q-learning (Sutton and Barto, 1998), with CMACs, a popular linear function approximator (Sutton and Barto, 1998), using a setup that matches that of Sutton (1996) as closely as possible. We found their performance to be nearly identical: softmax NEAT+Q received an average score of -52.75 while the Sarsa CMAC learner received -52.02. We believe this performance is approximately optimal, as it matches the best results published by other researchers, e.g. (Smart and Kaelbling, 2000).

This does not imply that neural networks are the function approximator of choice for the mountain car domain. On the contrary, Sutton's CMACs converge in many fewer episodes. Nonetheless, these results demonstrate that evolutionary function approximation and on-line evolution make it feasible to find approximately optimal policies using neural networks, something that some previous approaches (Boyan and Moore, 1995; Pyeatt and Howe, 2001), using manually designed networks, were unable to do.

Since the mountain car domain has only two state features, it is possible to visualize the value function. Figure 8 compares the value functions learned by softmax NEAT+Q to that of Sarsa with CMACs. For clarity, the graphs plot estimated steps to the goal. Since the agent receives a reward of -1 for each timestep until reaching the goal, this is equivalent to $-\max_a(Q(s,a))$. Surprisingly, the two value functions bear little resemblance to one another. While they share some very general characteristics, they differ markedly in both shape and scale. Hence, these graphs highlight a fact that has been noted before (Tesauro, 1994): that TD methods can learn excellent policies even if they estimate the value function only very grossly. So long as the value function assigns the highest value to the correct action, the agent will perform well.



(a) NEAT+Q Network

(b) CMAC

Figure 8: The value function, shown as estimated steps to the goal, of policies learned by softmax NEAT+Q and Sarsa using CMACs.

In the server job scheduling domain, finding alternative approaches for comparison is less straightforward. Substantial research about job scheduling already exists but most of the methods involved are not applicable here because they do not allow jobs to be associated with arbitrary utility functions. For example, Liu and Layland (1973) present methods for job scheduling in a real-time environment, in which a hard deadline is associated with each job. McWherter et al. (2004) present methods for scheduling jobs with different priority classes. However, unlike the utility functions shown in Section 4.2, the relative importance of a job type does not change as a function of time. McGovern et al. (2002) use reinforcement learning for CPU instruction scheduling but aim only to minimize completion time.

One method that can be adapted to the server job scheduling task is the generalized $c\mu$ rule (van Mieghem, 1995), in which the server always processes at time t the oldest job of that type k which maximizes $C'_k(o_k)/p_k$, where C'_k is the derivative of the cost function for job type k, o_k is the age of the oldest job of type k and p_k is the average processing time for jobs of type k. Since in our simulation all jobs require unit time to process and the cost function is just the additive inverse of the utility function, this is equivalent to processing the oldest job of that type k that maximizes $-U'_k(o_k)$, where U'_k is the derivative of the utility function for job type k. The generalized $c\mu$ rule has been proven approximately optimal given convex cost functions (van Mieghem, 1995). Since the utility functions, and hence the cost functions, are both convex and concave in our simulation, there is no theoretical guarantee about its performance in the server job scheduling domain. To see how well it performs in practice, we implemented it in our simulator and ran it for 1,000 episodes, obtaining an average score of -10,891.

Another scheduling algorithm applicable to this domain is the insertion scheduler, which performed the best in a previous study of a very similar domain (Whiteson and Stone, 2004). The insertion scheduler uses a simple, fast heuristic: it always selects for processing the job at the head of the queue but it keeps the queue ordered in a way it hopes will maximize aggregate utility. For any given ordering of a set of J jobs, the aggregate utility is:

$$\sum_{i \in J} U_i(a_i + p_i)$$

where $U_i(\cdot)$, a_i, and p_i are the utility function, current age, and position in the queue, respectively, of job i. Since there are $|J|!$ ways to order the queue, it is clearly infeasible to try them all. Instead, the insertion scheduler uses the following simple, fast heuristic: every time a new job is created, the insertion scheduler tries inserting it into each position in the queue, settling on whichever position yields the highest aggregate utility. Hence, by bootstrapping off the previous ordering, the insertion scheduler must consider only $|J]$ orderings. We implemented the insertion scheduler in our simulator and ran it for 1,000 episodes, obtaining an average score of -13,607.

Neither the $c\mu$ rule nor the insertion scheduler perform as well as softmax NEAT+Q, whose final generation champions received an average score of -9,723 over 1,000 episodes. Softmax NEAT+Q performed better despite the fact that the alternatives rely on much greater *a priori* knowledge about the dynamics of the system. Both alternatives require the scheduler to have a predictive model of the system, since their calculations depend on knowledge of the utility functions and the amount of time each job takes to complete. By contrast, softmax NEAT+Q, like many reinforcement learning algorithms, assumes such information is hidden and discovers a good policy from experience, just by observing state transitions and rewards.

If, in addition to assuming the scheduler has a model of the system, we make the unrealistic assumption that unlimited computation is available to the scheduler, then we can obtain an informa-

tive upper bound on performance. At each time step of the simulation, we can compute the optimal action analytically by treating the scheduling problem as an integer linear program. For each job $i \in J$ and for each position j in which it could be placed, the linear program contains a variable $x_{ij} \in \{0,1\}$. Associated with each variable is a weight $w_{ij} = U_i(a_i + j)$, which represents the reward the scheduler will receive when job i completes given that it currently resides in position j. Since the scheduler's goal is to maximize aggregate utility, the linear program must maximize $\sum_i \sum_j w_{ij} x_{ij}$. In addition to the constraint that $\forall i\, j : x_{ij} \in \{0,1\}$, the program is also constrained such that each job is in exactly one position: $\forall i : \sum_j x_{ij} = 1$ and that each position holds exactly one job: $\forall j : \sum_i x_{ij} = 1$.

A solution to the resulting integer linear program is an ordering that will maximize the aggregate utility of the jobs currently in the queue. If the scheduler always processes the job in the first position of this ordering, it will behave optimally *assuming no more jobs arrive*. Since new jobs are constantly arriving, the linear program must be re-solved anew at each time step. The resulting behavior may still be suboptimal since the decision about which job to process is made without reasoning about what types of jobs are likely to arrive later. Nonetheless, this analytical solution represents an approximate upper bound on performance in this domain.

Using the CPLEX software package, we implemented a scheduler based on the linear program described above and tested in our simulator for 1,000 episodes, obtaining an average score of -7,819. Not surprisingly, this performance is superior to that of softmax NEAT+Q, though it takes, on average, 741 times as long to run. The computational requirements of this solution are not likely to scale well either, since the number of variables in the linear program grows quadratically with respect to the size of the queue.

Figure 9 summarizes the performance of the alternative scheduling methods described in this section and compares them to softmax NEAT+Q. It also includes, as a lower bound on performance, a random scheduler, which received an average score of -15,502 over 1,000 episodes. A Student's t-test verified that the difference in performance between each pair of methods is statistically significant with 95% confidence. Softmax NEAT+Q performs the best except for the linear programming approach, which is computationally expensive and relies on a model of the system. Prior to learning, softmax NEAT+Q performs similarly to the random scheduler. The difference in performance between the best learned policies and the linear programming upper bound is 75% better than that of the baseline random scheduler and 38% better than that of the next best method, the $c\mu$ scheduler.

5.5 Comparing Darwinian and Lamarckian Evolutionary Computation

As described in Section 3.1, evolutionary function approximation can be implemented in either a Darwinian or Lamarckian fashion. The results presented so far all use the Darwinian implementation of NEAT+Q. However, it is not clear that this approach is superior even though it more closely matches biological systems. In this section, we compare the two approaches empirically in both the mountain car and server job scheduling domains. Many other empirical comparisons of Darwinian and Lamarckian systems have been conducted previously (D. Whitley, 1994; Yamasaki and Sekiguchi, 2000; Pereira and Costa, 2001) but ours is novel in that individual learning is based on a TD function approximator. In other words, these experiments address the question: when trying to approximate a TD value function, is a Darwinian or Lamarckian approach superior?

Figure 10 compares the performance of Darwinian and Lamarckian NEAT+Q in both the mountain car and server job scheduling domains. In both cases, we use off-line NEAT+Q, as the on-line versions tend to mute the differences between the two implementations. Though both implementa-

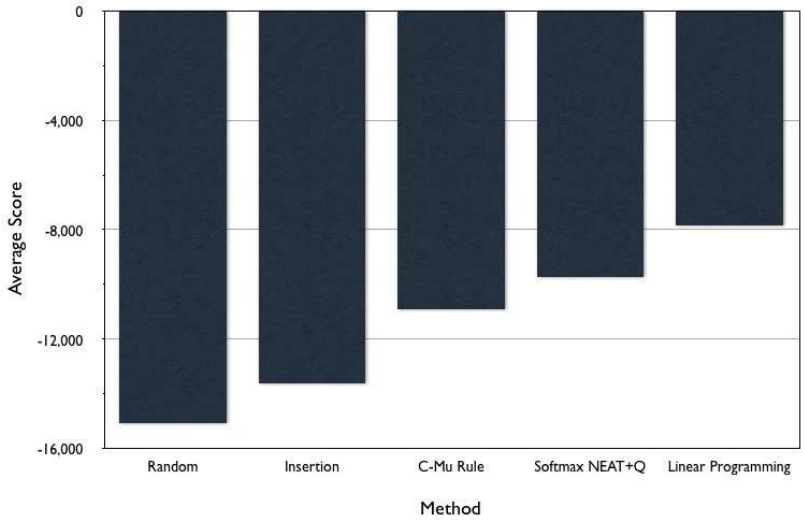
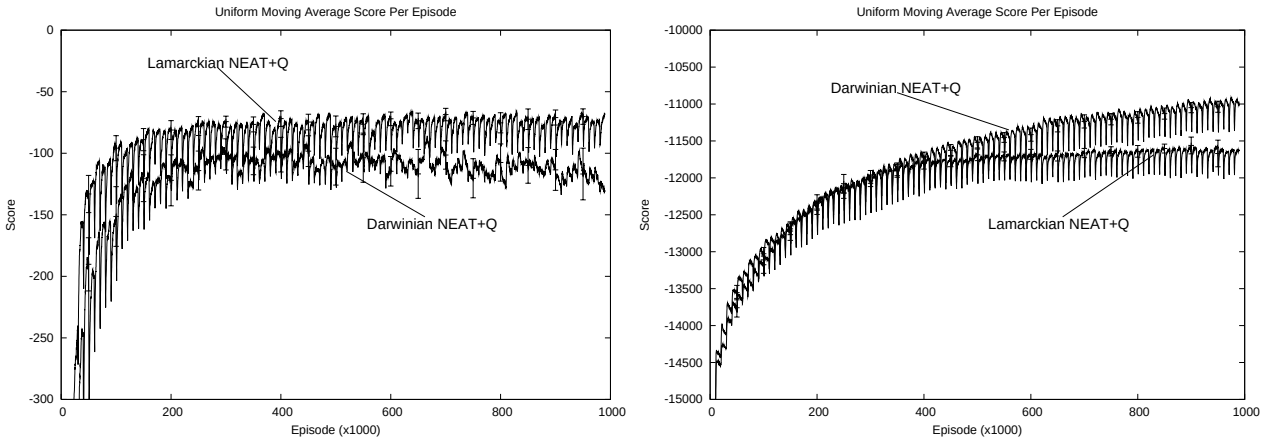


Figure 9: A comparison of the performance of softmax NEAT+Q and several alternative methods in the server job scheduling domain.

tions perform well in both domains, Lamarckian NEAT+Q does better in mountain car but worse in server job scheduling. Hence, the relative performance of these two approaches seems to depend critically on the dynamics of the domain to which they are applied. In the following section, we present some additional results that elucidate which factors affect their performance.



(a) Mountain Car (b) Server Job Scheduling

Figure 10: A comparison of Darwinian and Lamarckian NEAT+Q in the mountain car and server job scheduling domains.

5.6 Continual Learning Tests

In this section, we assess the performance of the best networks discovered by NEAT+Q when evaluated for many additional episodes. We compare two scenarios, one where the learning rate is annealed to zero after 100 episodes, just as in training, and one where it is not annealed at all. Comparing performance in these two scenarios allows us to assess the effect of continual learning on the evolved networks.

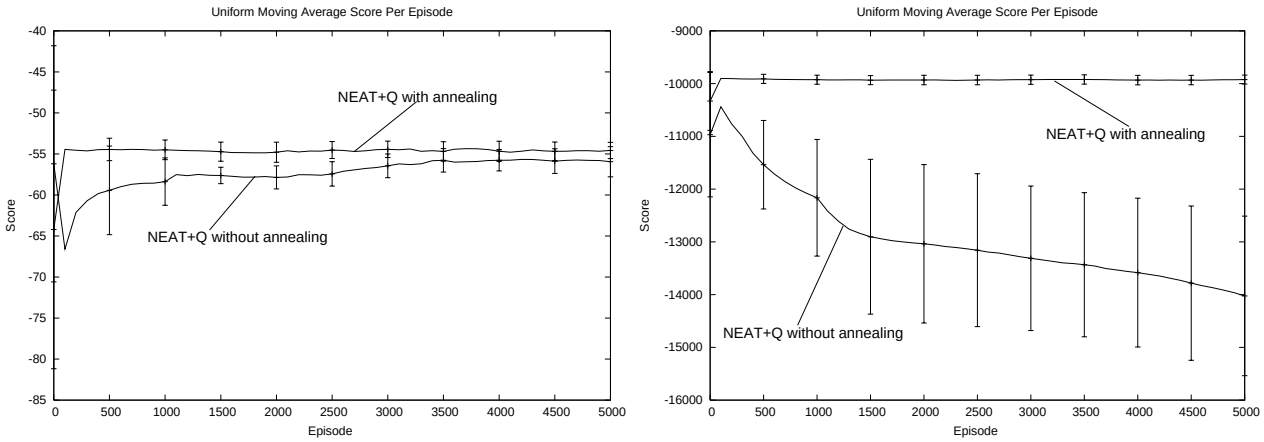
We hypothesized that NEAT+Q's best networks would perform well under continual learning in the mountain car domain but not in server job scheduling. This hypothesis was motivated by the results of early experiments with NEAT+Q. Originally, we did not anneal α at all. This setup worked fine in the mountain car domain but in scheduling it worked only with off-line NEAT+Q; on-line NEAT+Q actually performed worse than off-line NEAT+Q! Annealing NEAT+Q's learning rate eliminated the problem, as the experiments in Section 5.2 verify. If finding weights that remain stable under continual learning is more difficult in scheduling than in mountain car, it could explain this phenomenon, since ε-greedy and softmax selection, by giving many more episodes of learning to certain networks, could cause those networks to become unstable and perform poorly.

To test the best networks without continual learning, we selected the final generation champion from each run of off-line Darwinian NEAT+Q and evaluated it for an additional 5,000 episodes, i.e. 50 times as many episodes as it saw in training. During these additional episodes, the learning rate was annealed to zero by episode 100, just as in training. To test the best networks with continual learning, we repeated this experiment but did not anneal the learning rate at all. To prevent any unnecessary discrepancies between training and testing, we repeated the original NEAT+Q runs with annealing turned off and used the resulting final generation champions.

Figure 11 shows the results of these tests. In the mountain car domain, performance remains relatively stable regardless of whether the networks continue to learn. The networks tested without annealing show more fluctuation but maintain performance similar to those that were annealed. However, in the scheduling domain, the networks subjected to continual learning rapidly plummet in performance whereas those that are annealed continue to perform as they did in training. These results directly confirm our hypothesis that evolutionary computation can find weights that perform well under continual learning in mountain car but not in scheduling. This explains why on-line NEAT+Q does not require an annealed learning rate in mountain car but does in scheduling.

These tests also shed light on the comparison between Darwinian and Lamarckian NEAT+Q presented in Section 5.5. A surprising feature of the Darwinian approach is that it is insensitive to the issue of continual learning. Since weight changes do not affect offspring, evolution need only find weights that remain suitable during one individual's lifetime. By contrast, in the Lamarckian approach, weight changes accumulate from generation to generation. Hence, the TD updates that helped in early episodes can hurt later on. In this light it makes perfect sense that Lamarckian NEAT+Q performs better in mountain car than in scheduling, where continual learning is problematic.

These results suggest that the problem of stability under continual learning can greatly exacerbate the difficulty of performing neural network function approximation in practice. This issue is not specific to NEAT+Q, since Q-learning with manually designed networks achieved decent performance only when the learning rate was properly annealed. Darwinian NEAT+Q is a novel way of coping with this problem, since it obviates the need for long-term stability. In on-line evolutionary computation annealing may still be necessary but it is less critical to set the rate of decay precisely.



(a) Mountain Car (b) Server Job Scheduling

Figure 11: A comparison of the performance of the best networks evolved by NEAT+Q when tested, with and without annealing, for an additional 5,000 episodes.

When learning ends, it prevents only a given individual from continuing to improve. The system as a whole can still progress, as evolution exerts selective pressure and learning begins anew in the next generation.

6. Discussion

The results in the mountain car domain presented in Section 5, demonstrate that NEAT+Q can successfully train neural network function approximators in a domain which is notoriously problematic for them. However, NEAT+Q requires many more episodes to find good solutions (by several orders of magnitude) than CMACs do in the same domain. This contrast highlights an important drawback of NEAT+Q: since each candidate network must be trained long enough to let Q-learning work, it has very high sample complexity. In ongoing research, we are investigating ways of making NEAT+Q more sample-efficient. For example, preliminary results suggest that, by pre-training networks using methods based on experience replay (Lin, 1992), NEAT+Q's sample complexity can be dramatically reduced.

It is not surprising that NEAT+Q takes longer to learn than CMACs because it is actually solving a more challenging problem. CMACs, like other linear function approximators, require the human designer to engineer a state representation in which the optimal value function is linear with respect to those state features (or can be reasonably approximated as such). For example, when CMACs were applied to the mountain car domain, the two state features were tiled conjunctively (Sutton, 1996). By contrast, nonlinear function approximators like neural networks can take a simpler state representation and *learn* the important nonlinear relationships. Note that the state representation used by NEAT+Q, while discretized, does not include any conjunctive features of the original two state features. The important conjunctive features are represented by hidden nodes that are evolved automatically by NEAT.

Conjunctively tiling all state features is feasible in mountain car but quickly becomes impractical in domains with more state features. For example, doing so in the scheduling domain would require 16 CMACs, one for each action. In addition, each CMAC would have multiple 16-dimensional tilings. If 10 tilings were used and each state feature were discretized into 10 buckets, the resulting function approximator would have $16 \times 10 \times 10^{16}$ cells. Conjunctively tiling only some state features is feasible only with a large amount of domain expertise. Hence, methods like NEAT+Q that automatically learn nonlinear representations promise to be of great practical importance.

The results in the scheduling domain demonstrate that the proposed methods scale to a much larger, probabilistic domain and can learn schedulers that outperform existing non-learning approaches. The difference in performance between the best learned policies and the linear programming upper bound is 75% better than that of the baseline random scheduler and 38% better than that of the next best method, the $c\mu$ scheduler. However, the results also demonstrate that non-learning methods can do quite well in this domain. If so, is it worth the trouble of learning? We believe so. In a real system, the utility functions that the learner maximizes would likely be drawn directly from Service Level Agreements (SLAs), which are legally binding contracts governing how much clients pay their service providers as a function of the quality of service they receive (Walsh et al., 2004). Hence, even small improvements in system performance can significantly affect the service provider's bottom line. Substantial improvements like those demonstrated in our results, if replicated in real systems, could be very valuable indeed.

Overall, the main limitation of the results presented in this paper is that they apply only to neural networks. In particular, the analysis about the effects of continual learning (Section 5.6) may not generalize to other types of function approximation that are not as prone to instability or divergence if over-trained. While evolutionary methods could in principle be combined with any kind of function approximation, in practice it is likely to work well only with very concise representations. Methods like CMACs, which use many more weights, would result in very large genomes and hence be difficult for evolutionary computation to optimize. However, since such methods methods become impractical as the number of state features and actions grow, concise methods like neural networks may become increasingly important in harder domains. If so, evolutionary function approximation could be an important tool for automatically optimizing their representations.

7. Related Work

A broad range of previous research is related in terms of both methods and goals to the techniques presented in this paper. This section highlights some of that research and contrasts it with this work.

7.1 Optimizing Representations for TD Methods

A major challenge of using TD methods is finding good representations for function approximators. This paper addresses that problem by coupling TD methods with evolutionary techniques like NEAT that are proven representation optimizers. However, many other approaches are also possible.

One strategy is to train the function approximator using supervised methods that also optimize representations. For example, Rivest and Precup (2003) train cascade-correlation networks as TD function approximators. Cascade-correlation networks are similar to NEAT in that they grow internal topologies for neural networks. However, instead of using evolutionary computation to find such topologies, they rely on the network's error on a given training set to compare alternative representations. The primary complication of Rivest and Precup's approach is that cascade-correlation

networks, like many representation-optimizing supervised methods, need the training set to be both large and stable. TD methods do not naturally accommodate this requirement since they produce training examples only in sequence. Furthermore, those examples quickly become stale as the values upon which they were based are updated. Rivest and Precup address this problem using a novel caching system that in effect creates a hybrid value function consisting of a table and a neural network. While this approach delays the exploitation of the agent's experience, it nonetheless represents a promising way to marry the representation-optimizing capacity of cascade-correlation networks and other supervised algorithms with the power of TD methods.

Mahadevan (2005) suggests another strategy: using spectral analysis to derive basis functions for TD function approximators. His approach is similar to this work in that the agent is responsible for learning both the value function and its representation. It is different in that the representation is selected by analyzing the underlying structural properties of the state space, rather than evaluating potential representations in the domain.

A third approach is advanced by Sherstov and Stone (2005): using the Bellman error generated by TD updates to assess the reliability of the function approximator in a given region of the state or action space. They use this metric to automatically adjust the breadth of generalization for a CMAC function approximator. An advantage of this approach is that feedback arrives immediately, since Bellman error can be computed after each update. A disadvantage is that the function approximator's representation is not selected based on its actual performance, which may correlate poorly with Bellman error.

There is also substantial research that focuses on optimizing the agent's state and action representations, rather than the value function representation. For example, Santamaria et al. (1998) apply skewing functions to state-action pairs before feeding them as inputs to a function approximator. These skewing functions make the state-action spaces non-uniform and hence make it possible to give more resolution to the most critical regions. Using various skewing functions, they demonstrate improvement in the performance of TD learners. However, they do not offer any automatic way of determining how a given space should be skewed. Hence, a human designer still faces the burdensome task of manually choosing a representation, though in some domains using skewing functions may facilitate this process.

Smith (2002) extends this work by introducing a method that uses self-organizing maps to automatically learn nonlinear skewing functions for the state-action spaces of TD agents. Self-organizing maps use unsupervised learning methods to create spatially organized internal representations of the inputs they receive. Hence, the system does not use any feedback on the performance of different skewing functions to determine which one is most appropriate. Instead it relies on the heuristic assumption that more resolution should be given to regions of the space that are more frequently visited. While this is an intuitive and reasonable heuristic, it does not hold in general. For example, a reinforcement learning agent designed to respond to rare emergencies may spend most of its life in safe states where its actions have little consequence and only occasionally experience crisis states where its choices are critical. Smith's heuristic would incorrectly devote most of its resolution to representing the value function of the unimportant but frequently visited states. Evolutionary function approximation avoids this problem because it evaluates competing representations by testing them in the actual task. It explicitly favors those representations that result in higher performance, regardless of whether they obey a given heuristic.

McCallum (1995) also presents a method for optimizing an agent's state representation. His approach automatically learns tree-structured short-term memories that allow reinforcement learning agents to prevent the state aliasing that results from hidden state.

7.2 Combining Evolutionary Computation with Other Learning Methods

Because of the potential performance gains offered by the Baldwin Effect, many researchers have developed methods that combine evolutionary computation with other learning methods that act within an individual's lifetime. Some of this work is applied to supervised problems, in which evolutionary computation can be coupled with any supervised learning technique such as backpropagation in a straightforward manner. For example, Boers et al. (1995) introduce a neuroevolution technique that, like NEAT, tries to discover appropriate topologies. They combine this method with backpropagation and apply the result to a simple supervised learning problem. Also, Giraud-Carrier (2000) uses a genetic algorithm to tune the parameters of RBF networks, which he applies to a supervised classification problem.

Inducing the Baldwin Effect on reinforcement learning problems is more challenging, since they do not automatically provide the target values necessary for supervised learning. The algorithms presented in this paper use TD methods to estimate those targets, though researchers have tried many other approaches. McQuestion and Miikkulainen (1997) present a neuroevolutionary technique that relies on each individual's parents to supply targets and uses backpropagation to train towards those targets. Stanley et al. (2003) avoid the problem of generating targets by using Hebbian rules, an unsupervised technique, to change a neural network during its fitness evaluation. The network's changes are not directed by any error signal but they allow the network to retain a memory of previously experienced input sequences. Hence their approach is an alternative to recurrent neural networks. Downing (2001) combines genetic programming with Q-learning using a simple tabular representation; genetic programming automatically learns how to discretize the state space.

Nolfi et al. (1994) present a neuroevolutionary system that adds extra outputs to the network that are designed to predict what inputs will be presented next. When those inputs actually arrive, they serve as targets for backpropagation, which adjusts the network's weights starting from the added outputs. This technique allows a network to be adjusted during its lifetime using supervised methods but relies on the assumption that forcing it to learn to predict future inputs will help it select appropriate values for the remaining outputs, which actually control the agent's behavior. Another significant restriction is that the weights connecting hidden nodes to the action outputs cannot be adjusted at all during each fitness evaluation.

Ackley and Littman (1991) combine neuroevolution with reinforcement learning in an artificial life context. Evolutionary computation optimizes the initial weights of an "action network" that controls an agent in a foraging scenario. The weights of the network are updated during each individual's lifetime using a reinforcement learning algorithm called CRBP on the basis of a feedback signal that is also optimized with neuroevolution. Hence, their approach is similar to the one described in this paper, though the neuroevolution technique they employ does not optimize network topologies and CRBP does not learn a value function.

XCS (Butz and Wilson, 2002), based on learning classifier systems (Lanzi et al., 2000), combine evolutionary computation and reinforcement learning in a different way. Each member of the population, instead of representing a complete policy, represents just a single classifier, which specifies the action the agent should take for some subset of the state space. Hence, the population as a whole

represents a single evolving policy. Classifiers are selected for reproduction based on the accuracy of their value estimates and speciation is used to ensure the state space is properly covered.

Other combinations of evolutionary computation with other learning methods include Arita and Suzuki (2000), who study iterated prisoner's dilemma; French and Messinger (1994) and Sasaki and Tokoro (1999), who use artificial life domains; and Niv et al. (2002) in a foraging bees domain.

Another important related method is VAPS (Baird and Moore, 1999). While it does not use evolutionary computation, it does combine TD methods with policy search methods. It provides a unified approach to reinforcement learning that uses gradient descent to try to simultaneously maximize reward and minimize error on Bellman residuals. A single parameter determines the relative weight of these goals. Because it integrates policy search and TD methods, VAPS is in much the same spirit as evolutionary function approximation. However, the resulting methods are quite different. While VAPS provides several impressive convergence guarantees, it does not address the question of how to represent the value function.

Other researchers have also sought to combine TD and policy search methods. For example, Sutton et al. (2000) use policy gradient methods to search policy space but rely on TD methods to obtain an unbiased estimate of the gradient. Similarly, in actor-critic methods (Konda and Tsitsiklis, 1999), the actor optimizes a parameterized policy by following a gradient informed by the critic's estimate of the value function. Like VAPS, these methods do not learn a representation for the value function.

7.3 Variable Evaluations in Evolutionary Computation

Because it allows members of the same population to receive different numbers of evaluations, the approach to on-line evolutionary computation presented here is similar to previous research about optimizing noisy fitness functions. For example, Stagge (1998) introduces mechanisms for deciding which individuals need more evaluations for the special case where the noise is Gaussian. Beielstein and Markon (2002) use a similar approach to develop tests for determining which individuals should survive. However, this area of research has a significantly different focus, since the goal is to find the best individuals using the fewest evaluations, not to maximize the reward accrued during those evaluations.

The problem of using evolutionary systems on-line is more closely related to other research about the exploration/exploitation tradeoff, which has been studied extensively in the context of TD methods (Watkins, 1989; Sutton and Barto, 1998) and multiarmed bandit problems (Bellman, 1956; Macready and Wolpert, 1998; Auer et al., 2002). The selection mechanisms we employ in our system are well-established though, to our knowledge, their application to evolutionary computation is novel.

8. Future Work

There are many ways that the work presented in this paper could be extended, refined, or further evaluated. This section enumerates a few of the possibilities.

Using Different Policy Search Methods This paper focuses on using evolutionary methods to automate the search for good function approximator representations. However, many other forms of policy search such as PEGASUS (Ng and Jordan, 2000) and policy gradient methods (Sutton et al., 2000; Kohl and Stone, 2004) have also succeeded on difficult reinforcement learning tasks. TD

methods could be combined with these methods in the same way they are combined with evolutionary computation in this paper. In the future, we plan to test some of these alternative combinations.

Reducing Sample Complexity As mentioned in Section 6, one disadvantage of evolutionary function approximation is its high sample complexity, since each fitness evaluation lasts for many episodes. However, in domains where the fitness function is not too noisy, each fitness evaluation could be conducted in a single episode if the candidate function approximator was pre-trained using methods based on experience replay (Lin, 1992). By saving sample transitions from the previous generation, each new generation could be be trained off-line. This method would use much more computation time but many fewer sample episodes. Since sample experience is typically a much scarcer resource than computation time, this enhancement could greatly improve the practical applicability of evolutionary function approximation.

Addressing Non-Stationarity In *non-stationary* domains, the environment can change in ways that alter the optimal policy. Since this phenomenon occurs in many real-world scenarios, it is important to develop methods that can handle it robustly. Evolutionary and TD methods are both well suited to non-stationary tasks and we expect them to retain that capability when combined. In fact, we hypothesize that evolutionary function approximation will adapt to non-stationary environments *better* than manual alternatives. If the environment changes in ways that alter the optimal representation, evolutionary function approximation can adapt, since it is continually testing different representations and retaining the best ones. By contrast, even if they are effective at the original task, manually designed representations cannot adapt in the face of changing environments.

On-line evolutionary computation should also excel in non-stationary environments, though some refinement will be necessary. The methods presented in this paper implicitly assume a stationary environment because they compute the fitness of each individual by averaging over *all* episodes of evaluation. In non-stationary environments, older evaluations can become stale and misleading. Hence, fitness estimates should place less trust in older evaluations. This effect could easily be achieved using recency-weighting update rules like those employed by table-based TD methods.

Using Steady-State Evolutionary Computation The NEAT algorithm used in this paper is an example of *generational* evolutionary computation, in which an entire population is is evaluated before any new individuals are bred. Evolutionary function approximation might be improved by using a *steady-state* implementation instead (Fogarty, 1989). Steady-state systems never replace an entire population at once. Instead, the population changes incrementally after each fitness evaluation, when one of the worst individuals is removed and replaced by a new offspring whose parents are among the best. Hence, an individual that receives a high score can more rapidly effect the search, since it immediately becomes a potential parent. In a generational system, that individual cannot breed until the beginning of the following generation, which might be thousands of episodes later. Hence, steady-state systems could help evolutionary function approximation perform better in on-line and non-stationary environments by speeding the adoption of new improvements. Fortunately, a steady-state version of NEAT already exists (Stanley et al., 2005) so this extension is quite feasible.

9. Conclusion

Reinforcement learning is an appealing and empirically successful approach to finding effective control policies in large probabilistic domains. However, it requires a good deal of expert knowledge

to put into practice, due in large part to the need for manually defining function approximator representations. This paper offers hope that machine learning methods can be used to discover those representations automatically, thus broadening the practical applicability of reinforcement learning.

This paper makes three main contributions. First, it introduces evolutionary function approximation, which automatically discovers effective representations for TD function approximators. Second, it introduces on-line evolutionary computation, which employs selection mechanisms borrowed from TD methods to improve the on-line performance of evolutionary computation. Third, it provides a detailed empirical study of these methods in the mountain car and server job scheduling domains.

The results demonstrate that evolutionary function approximation can significantly improve the performance of TD methods and on-line evolutionary computation can significantly improve evolutionary methods. Combined, our novel algorithms offer a promising and general approach to reinforcement learning in large probabilistic domains.

Acknowledgments

Thanks to Richard Sutton, Michael Littman, Gerry Tesauro, and Manuela Veloso for helpful discussions and ideas. Thanks to Risto Miikkulainen, Nick Jong, Bikram Banerjee, Shivaram Kalyanakrishnan, and the anonymous reviewers for constructive comments about earlier versions of this work. This research was supported in part by NSF CAREER award IIS-0237699 and an IBM faculty award.

Appendix A. Statistical Significance

To assess the statistical significance of the results presented in Section 5, we performed a series of Student's t-tests on each pair of methods in each domain. For each pair, we performed a t-test after every 100,000 episodes. Tables 1 and 2 summarize the results of these tests for the mountain car and server job scheduling domains, respectively. In each table, the values in each cell indicate the range of episodes for which performance differences were significant with 95% confidence.

Appendix B. NEAT Parameters

Table 3 details the NEAT parameters used in our experiments. Stanley and Miikkulainen (2002) describe the semantics of these parameters in detail.

Episodes (x1000)	Q-Learning	Off-Line NEAT	ε-Greedy NEAT	Softmax NEAT	Off-Line NEAT+Q	Softmax NEAT+Q	Lamarckian NEAT+Q
Q-Learning							
Off-Line NEAT	300 to 1000						
ε-Greedy NEAT	200 to 1000	200 to 1000					
Softmax NEAT	200 to 1000	200 to 1000	200 to 1000				
Off-Line NEAT+Q	200 to 1000	200 to 500	200 to 1000	200 to 1000			
Softmax NEAT+Q	100 to 1000	200 to 1000	200 to 1000	900 to 1000	200 to 1000		
Lamarkian NEAT+Q	200 to 1000	200 to 1000	200 to 1000	200 to 1000	200 to 1000	100 to 1000	

Table 1: A summary of the statistical significance of differences in average performance between each pair of methods in mountain car (see Figures 4, 6, 7 & 10). Values in each cell indicate the range of episodes for which differences were significant with 95% confidence.

Episodes (x1000)	Q-Learning	Off-Line NEAT	ε-Greedy NEAT	Softmax NEAT	Off-Line NEAT+Q	Softmax NEAT+Q	Lamarckian NEAT+Q
Q-Learning							
Off-Line NEAT	300 to 1000						
ε-Greedy NEAT	200 to 1000	200 to 1000					
Softmax NEAT	200 to 1000	200 to 1000	not significant throughout				
Off-Line NEAT+Q	300 to 1000	300 to 500	100 to 1000	200 to 1000			
Softmax NEAT+Q	200 to 1000	200 to 1000	400 to 1000	200 to 1000	200 to 1000		
Lamarckian NEAT+Q	300 to 1000	300 to 1000	100 to 1000	100 to 1000	700 to 1000	200 to 1000	

Table 2: A summary of the statistical significance of differences in average performance between each pair of methods in server job scheduling (see Figures 4, 6, 7 & 10). Values in each cell indicate the range of episodes for which differences were significant with 95% confidence.

Parameter	Value	Parameter	Value	Parameter	Value
weight-mut-power	0.5	recur-prop	0.0	disjoint-coeff (c_1)	1.0
excess-coeff (c_2)	1.0	mutdiff-coeff (c_3)	2.0	compat-threshold	3.0
age-significance	1.0	survival-thresh	0.2	mutate-only-prob	0.25
mutate-link-weights-prob	0.9	mutate-add-node-prob (m_n)	0.02	mutate-add-link-prob (m_l)	0.1
interspecies-mate-rate	0.01	mate-multipoint-prob	0.6	mate-multipoint-avg-prob	0.4
mate-singlepoint-prob	0.0	mate-only-prob	0.2	recur-only-prob	0.0
pop-size (p)	100	dropoff-age	100	newlink-tries	50
babies-stolen	0	num-compat-mod	0.3	num-species-target	6

Table 3: The NEAT parameters used in our experiments.

References

D. Ackley and M. Littman. Interactions between learning and evolution. *Artificial Life II, SFI Studies in the Sciences of Complexity*, 10:487–509, 1991.

T. Arita and R. Suzuki. Interactions between learning and evolution: The outstanding strategy generated by the Baldwin Effect. *Artificial Life*, 7:196–205, 2000.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

L. Baird and A. Moore. Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

J. M. Baldwin. A new factor in evolution. *The American Naturalist*, 30:441–451, 1896.

T. Beielstein and S. Markon. Threshold selection, hypothesis tests and DOE methods. In *2002 Congresss on Evolutionary Computation*, pages 777–782, 2002.

R. E. Bellman. A problem in the sequential design of experiments. *Sankhya*, 16:221–229, 1956.

E. J. W. Boers, M. V. Borst, and I. G. Sprinkhuizen-Kuyper. Evolving Artificial Neural Networks using the "Baldwin Effect". In *Artificial Neural Nets and Genetic Algorithms, Proceedings of the International Conference in Ales, France*, 1995.

J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 671–678. Morgan Kaufmann Publishers, Inc., 1994.

J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7*, 1995.

M. V. Butz and S. W. Wilson. An algorithmic description of XCS. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 6(3-4):144–153, 2002.

R. H. Crites and A. G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998.

K. Mathias D. Whitley, S. Gordon. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature - PPSN III*, pages 6–15, 1994.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adverserial classification. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–108, 2004.

K. L. Downing. Reinforced genetic programming. *Genetic Programming and Evolvable Machines*, 2(3):259–288, 2001.

L. Ertoz, A. Lazarevic, E. Eilerston, A. Lazarevic, P. Tan, P. Dokas, V. Kumar, and J. Srivastava. *The MINDS - Minnesota Intrustion Detection System*, chapter 3. MIT Press, 2004.

T. C. Fogarty. An incremental genetic algorithm for real-time learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 416–419, 1989.

R. French and A. Messinger. Genes, phenes and the Baldwin effect: Learning and evolution in a simulated population. *Artificial Life*, 4:277–282, 1994.

C. Giraud-Carrier. Unifying learning with evolution through Baldwinian evolution and Lamarckism: A case study. In *Proceedings of the Symposium on Computational Intelligence and Learning (CoIL-2000)*, pages 36–41, 2000.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989.

D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154, 1987.

F. Gomez, D. Burger, and R. Miikkulainen. A neuroevolution method for dynamic resource allocation on a chip multiprocessor. In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, pages 2355–2361, 2001.

F. Gruau and D. Whitley. Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect. *Evolutionary Computation*, 1:213–233, 1993.

F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89, 1996.

G. E. Hinton and S. J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI, 1975.

J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.

N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conference on Artificial Intelligence*, pages 611–616, July 2004.

V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 11*, pages 1008–1014, 1999.

R. M. Kretchmar and C. W. Anderson. Comparison of CMACs and radial basis functions for local function approximators in reinforcement learning. In *International Conference on Neural Networks*, 1997.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4(2003):1107–1149, 2003.

P. L. Lanzi, W. Stolzmann, and S. Wilson. *Learning classifier systems from foundations to applications*. Springer, 2000.

L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association of Computing Machinery*, 20(1):46–61, January 1973.

W. G. Macready and D. H. Wolpert. Bandit problems and the exploration/exploitation tradeoff. In *IEEE Transactions on Evolutionary Computation*, volume 2(1), pages 2–22, 1998.

S. Mahadevan. Samuel meets Amarel: Automating value function approximation using global state space analysis. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

S. Mannor, R. Rubenstein, and Y. Gat. The cross-entropy method for fast policy search. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 512–519, 2003.

A. R. McCallum. Instance-based utile distinctions for reinforcement learning. In *Proceedings of the Twelfth International Machine Learning Conference*, 1995.

A. McGovern, E. Moss, and A. G. Barto. Building a block scheduler using reinforcement learning and rollouts. *Machine Learning*, 49(2-3):141–160, 2002.

P. McQuesten and R. Miikkulainen. Culling and teaching in neuro-evolution. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 760–767, 1997.

D. McWherter, B. Schroeder, N. Ailamaki, and M. Harchol-Balter. Priority mechanisms for OLTP and transactional web applications. In *Proceedings of the Twentieth International Conference on Data Engineering*, 2004.

A. Y. Ng and M. I. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

Y. Niv, D. Joel, I. Meilijson, and E. Ruppin. Evolution of reinforcement learning in foraging bees: A simple explanation for risk averse behavior. *Neurocomputing*, 44(1):951–956, 2002.

S. Nolfi, J. L. Elman, and D. Parisi. Learning and evolution in neural networks. *Adaptive Behavior*, 2:5–28, 1994.

F. B. Pereira and E. Costa. Understanding the role of learning in the evolution of busy beaver: A comparison between the Baldwin Effect and a Lamarckian strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001.

L. D. Pyeatt and A. E. Howe. Decision tree function approximation in reinforcement learning. In *Proceedings of the Third International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, pages 70–77, 2001.

N. J. Radcliffe. Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90, 1993.

M. Reidmiller. Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the Sixteenth European Conference on Machine Learning*, pages 317–328, 2005.

F. Rivest and D. Precup. Combining TD-learning with cascade-correlation networks. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 632–639. AAAI Press, 2003.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, pages 318–362. 1986.

J. Santamaria, R. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2), 1998.

T. Sasaki and M. Tokoro. Evolving learnable neural networks under changing environments with various rates of inheritance of acquired characters: Comparison between Darwinian and Lamarckian evolution. *Artificial Life*, 5(3):203–223, 1999.

A. A. Sherstov and P. Stone. Function approximation via tile coding: Automating parameter choice. In J.-D. Zucker and I. Saitta, editors, *SARA 2005*, volume 3607 of *Lecture Notes in Artificial Intelligence*, pages 194–205. Springer Verlag, Berlin, 2005.

W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeeth International Conference on Machine Learning*, pages 903–910, 2000.

A. J. Smith. Applications of the self-organizing map to reinforcement learning. *Journal of Neural Networks*, 15:1107–1124, 2002.

P. Stagge. Averaging efficiently in the presence of noise. In *Parallel Problem Solving from Nature*, volume 5, pages 188–197, 1998.

K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Evolving adaptive neural networks with and without adaptive synapses. In *Proceeedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2557–2564, 2003.

K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Evolving neural network agents in the NERO video game. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, 2005.

K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004a.

K. O. Stanley and R. Miikkulainen. Evolving a roving eye for go. In *Proceedinngs of the Genetic and Evolutionary Computation Conference*, 2004b.

R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044, 1996.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

J. A. van Mieghem. Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *The Annals of Applied Probability*, 5(3):809–833, August 1995.

W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das. Utility functions in autonomic systems. In *Proceedings of the International Conference on Autonomic Computing*, pages 70–77, 2004.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

S. Whiteson and P. Stone. Adaptive job routing and scheduling. *Engineering Applications of Artificial Intelligence*, 17(7):855–869, 2004. Corrected version.

J. Wildstrom, P. Stone, E. Witchel, R. J. Mooney, and M. Dahlin. Towards self-configuring hardware for distributed computer systems. In *The Second International Conference on Autonomic Computing*, pages 241–249, June 2005.

K. Yamasaki and M. Sekiguchi. Clear explanation of different adaptive behaviors between Darwinian population and Lamarckian population in changing environment. In *Proceedings of the Fifth International Symposium on Artificial Life and Robotics*, volume 1, pages 120–123, 2000.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

W. Zhang and T. G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the 1995 Joint Conference on Artificial Intelligence*, pages 1114–1120, 1995.

M. Zweben and M. Fox, editors. *Intelligent Scheduling*. Morgan Kaufmann, 1998.

Journal of Machine Learning Research 7 (2006) 919–943 Submitted 2/05; Revised 12/05; Published 6/06

Rearrangement Clustering: Pitfalls, Remedies, and Applications

Sharlee Climer SHARLEE@CLIMER.US
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130-4899, USA

Weixiong Zhang ZHANG@CSE.WUSTL.EDU
Department of Computer Science and Engineering
and Department of Genetics
Washington University in St. Louis
St. Louis, MO 63130-4899, USA

Editor: Thorsten Joachims

Abstract

Given a matrix of values in which the rows correspond to objects and the columns correspond to features of the objects, rearrangement clustering is the problem of rearranging the rows of the matrix such that the sum of the similarities between adjacent rows is maximized. Referred to by various names and reinvented several times, this clustering technique has been extensively used in many fields over the last three decades. In this paper, we point out two critical pitfalls that have been previously overlooked. The first pitfall is deleterious when rearrangement clustering is applied to objects that form natural clusters. The second concerns a similarity metric that is commonly used. We present an algorithm that overcomes these pitfalls. This algorithm is based on a variation of the Traveling Salesman Problem. It offers an extra benefit as it automatically determines cluster boundaries. Using this algorithm, we *optimally* solve four benchmark problems and a 2,467-gene expression data clustering problem. As expected, our new algorithm identifies better clusters than those found by previous approaches in all five cases. Overall, our results demonstrate the benefits of rectifying the pitfalls and exemplify the usefulness of this clustering technique. Our code is available at our websites.

Keywords: clustering, visualization of patterns in data, bond energy algorithm, traveling salesman problem, asymmetric clustering

1. Introduction

> Science is organized knowledge. Wisdom is organized life.
> - *Immanuel Kant*

Clustering is aimed at discovering structures and patterns of a given data set. As a fundamental problem and technique for data analysis, clustering has become increasingly important, especially with the explosion of data on the World Wide Web and the advent of massive quantities of genomic data.

A given set of objects can be clustered in a variety of ways, depending on three criteria: the degree of granularity desired, the distance measure that is employed, and the objective that is stated as the goal for the clustering.

The degree of granularity affects clustering results. There is usually a range of values for the number of clusters k that are of interest. The desired degree of granularity is problem specific. Consider for example, clustering the population of a large geographical region. A company wishing to determine the location of a few distribution centers would desire a small k value, while a utility company may have applications requiring several thousand clusters. Once a range of values is established for k, it is frequently useful to determine clustering results for several values of k within this range and use domain knowledge to determine the best solution.

The choice of a distance measure also impacts clustering results. A distance measure is a means of quantifying the pair-wise dissimilarities between objects. Alternatively, a similarity measure is used to quantify pair-wise similarities. When objects can be accurately characterized as points residing within a metric space, the Euclidean distance is frequently employed. Distance functions are sometimes assumed to be symmetric (*i.e.*, $d(i,j) = d(j,i)$), obey the triangle inequality, and require that $d(i,i) = 0$. In this paper, we do not assume that any of these properties necessarily hold as there exist applications when effective distance measures do not obey these properties. For instance, in the realm of document clustering, the *cosine distance* is frequently employed, although this measure does not obey the triangle inequality (Steinbach et al., 2000).

Finally, the objective that is stated as the goal guides the clustering results. Clustering problems are interesting as there is no single objective that is universally applicable. Many objective functions have been proposed and used throughout the history of clustering. Some objectives optimize with respect to distances of objects to their respective cluster centers. Some base their optimizations on *diameters*, or maximum pair-wise distances of each cluster. These objectives tend to assume somewhat regular cluster configurations and can lead to undesirable results when cluster boundaries are complex as in Figure 1. Intuitive clustering using the Euclidean distance measure is shown in Figure 1(b). In this case, many objects are closer to the center of a different cluster than their own and the diameters are not minimized.

One clustering problem that has been studied extensively is the problem of identifying and displaying groups of similar objects that occur in complex data arrays (McCormick et al., 1972; Arabie and Hubert, 1990; Arabie et al., 1988; Alpert, 1996; Johnson et al., 2004; Torres-Velzquez and Estivill-Castro, 2004). The problem can be represented as a matrix where the rows correspond to the objects to be clustered and the columns are their features. Similar objects can be identified and displayed by rearranging the rows so that the overall similarity between all adjacent objects is maximized. After rearranging the rows, the clusters are identified either manually or automatically in a second step.

This clustering problem actually consists of two objectives. The first objective is consistently used for a number of applications and requires either the maximization of the sum of similarities between adjacent rows or the minimization of the sum of distances between adjacent rows. The second objective varies in the literature, however, the general goal is to identify clusters among the rearranged objects.

In 1972, McCormick, Schweitzer, and White introduced the *bond energy algorithm* (BEA) which yields an approximate solution for the first objective of this clustering problem. Since that time, a "fast-growing literature" (Marcotorchino, 1987, p. 73) has appeared on this subject. This problem has been applied to a number of different applications in diverse areas, such as database

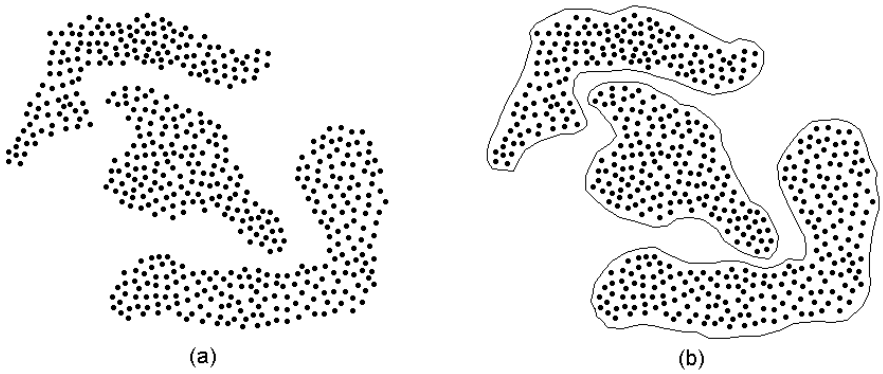


Figure 1: (a) A data set with Euclidean distance used for the distance measure. (b) Intuitive clustering of the data set. Many objects are closer to the center of a different cluster than their own and the diameters are not minimized.

design (Ozsu and Valduriez, 1999), data mining (Dunham, 2003), factorization of sparse matrices (Alpert, 1996), matrix compression (Johnson et al., 2004), information retrieval (March, 1983), manufacturing (Kusiak, 1985), imaging (Kusiak, 1984), marketing (Arabie et al., 1988), software engineering (Gorla and Zhang, 1999), VLSI circuit design (Alpert, 1996), clustering of web-users (Torres-Velzquez and Estivill-Castro, 2004), shelf space allocation (Lim et al., 2004), and clustering of genes (Liu et al., 2004).

For some of these applications, it is useful to also rearrange the columns. Since the rearrangement of the columns is independent of the rearrangement of rows, the columns can be rearranged in a separate step, using the same technique that is used for the rows.

The core problem does not seem to have been given a consistent name and has been reinvented several times[1] (McCormick et al., 1972; Alpert and Kahng, 1997; Johnson et al., 2004; Torres-Velzquez and Estivill-Castro, 2004). It has been referred to as "structuring of matrices" (Punnen, 2002), "data reorganization" (McCormick et al., 1972), "clustering of data arrays" (Lenstra, 1974), "restricted partitioning" (Alpert and Kahng, 1997), and "matrix reordering" (Johnson et al., 2004). Due to its nature and for the convenience of our discussion, we call this clustering problem *rearrangement clustering*.

Almost all of the existing rearrangement clustering algorithms have focused on arranging the objects to approximately maximize the overall similarity (or minimize the overall dissimilarity) between adjacent objects, while few methods have been developed to automatically identify the clusters of objects that form natural groups. An exception is the work of Alpert and Kahng (1997), in which they identified optimal partitioning for a given number of clusters k. For many rearrangement clustering algorithms, the objects are first rearranged, then a domain expert determines the cluster intervals.

Although rearrangement clustering has been extensively used for more than 30 years, there are two serious pitfalls that have been previously overlooked. The first pitfall is deleterious when the objects to be rearranged form natural clusters; which is the case for every application we have

1. In fact, we also reinvented it ourselves at the beginning of this research.

observed. The second pitfall concerns the use of the *measure of effectiveness* (ME) metric, which is employed by the bond energy algorithm.

In this paper, we first briefly summarize background material. Then we identify two pitfalls of previous approaches. In Section 4 we present techniques for rectifying these pitfalls. Section 5 describes the implementation of rearrangement clustering without the pitfalls. We summarize the results of using this implementation for four benchmark problems and a 2,467 gene expression data clustering problem in Section 6. We conclude this paper with a brief discussion. A preliminary report on this work appeared in an earlier paper (Climer and Zhang, 2004).

2. Background

Given a matrix in which each row corresponds to an object and each column corresponds to a feature of the objects, rearrangement clustering is the problem of shuffling the rows around until the sum of the similarities between adjacent rows is maximized. The similarity of two objects can be measured by a similarity score defined on their features.

More formally, let P represent the set of all possible permutations of rows for a given matrix and $s(i, j)$ represent a non-negative similarity measure for objects (rows) i, j. Then an optimal permutation $p \in P$ for the given similarity measure is

$$V(P) = max\left(\sum_{i=1}^{n-1} s(i, i+1)\right) \tag{1}$$

for n objects. Conversely, given a non-negative *dissimilarity* function, $d(i, j)$, an optimal permutation $p \in P$ is

$$W(P) = min\left(\sum_{i=1}^{n-1} d(i, i+1)\right). \tag{2}$$

2.1 Bond Energy Algorithm

One of the first algorithms to tackle rearrangement clustering was the bond energy algorithm (BEA) (McCormick et al., 1972). BEA uses the measure of effectiveness (ME) in which the similarity measure for two rows, i and j, is

$$s(i, j) = \sum_{k=1}^{m} a_{ik} a_{jk} \tag{3}$$

where m is the number of features and a_{ik} is the (non-negative) kth feature of object i.[2] Hence, each element in the matrix, except those in the last row, is multiplied by the element directly below it, and ME is equal to the sum of these products. The intuition behind this similarity measure is that large values will be drawn to other large values, and small values to other small values, so as to increase the overall sum of the products. The term *bond energy* expresses this concept. BEA computes an approximate solution that attempts to maximize ME.

BEA has gained wide recognition and remains the algorithm of choice for a number of applications. One such use arises in manufacturing. In these applications, parts or machines with similar features are grouped into families in a process referred to as *cell formation*. Chu and Tsai (1990)

2. McCormick et al. used a single ME function to simultaneously quantify similarities of adjacent columns as well as adjacent rows.

compared three rearrangement algorithms for this application: rank order clustering (ROC) (King, 1980), direct clustering analysis (DCA) (Chan and Milner, 1982), and BEA. They ran trials for various manufacturing applications and found that BEA outperformed the other two algorithms in all of their tests.

BEA is also popular for database design. The goal here is to determine sets of attributes that are accessed by distinct sets of applications, using a process referred to as *vertical fragmentation* (Ozsu and Valduriez, 1999). BEA has been promoted for this use (Hoffer and Severance, 1975; Navathe et al., 1984). Furthermore, BEA is included in textbooks on database design (Ozsu and Valduriez, 1999) and data mining (Dunham, 2003).

BEA has also been used for analyzing program structure in the field of software engineering (Gorla and Zhang, 1999). The locations of all of the components, their respective calls, and the depth of nested calls all contribute to the difficulties that can be expected during the debugging and maintenance phases of a program's life. Due to the fact that these phases are generally much more expensive than the other phases, structural improvements are valuable. BEA has been used to determine the placement of components with good results (Gorla and Zhang, 1999).

A recent application of BEA was the clustering of gene expression data (Liu et al., 2004). The current microarray gene expression profiling technology (Baldi and Hatfield, 2002; Eisen et al., 1998) is able to examine the expressions of hundreds, thousands or even tens of thousands of genes at once. A large amount of microarray data has been collected on numerous species and organisms, ranging from microbial organisms to plants to animals. The results of a set of microarray experiments on a collection of genes under different conditions are typically arranged as a matrix of gene expression levels in real values, where the rows represent the genes to be analyzed and the columns corresponds to experimental conditions (Baldi and Hatfield, 2002; Eisen et al., 1998). The objective is to identify and display clusters of genes that have similar expression patterns. BEA was shown to outperform k-means for the clustering of 44 yeast genes (Liu et al., 2004).

2.2 Traveling Salesman Problem

It has been pointed out that rearrangement clustering is equivalent to the Traveling Salesman Problem (TSP) and can be solved to *optimality* by solving the TSP (Lenstra, 1974; Lenstra and Kan, 1975). The TSP for n cities is the problem of finding a tour visiting all the cities and returning to the starting city such that the sum of the distances between consecutive cities is minimized. In other words, the TSP is to find a cyclic permutation of the cities so that the total distance of adjacent cities under the permutation is minimized. It is well known that TSP is NP-hard (Karp, 1972).

The mapping from a rearrangement clustering problem instance to a TSP instance is straightforward (Lenstra, 1974; Lenstra and Kan, 1975). We first view each object as a city and transform the dissimilarity between two objects to the distance between the corresponding cities. The TSP tour, which must have the minimum distance among all complete tours, is an optimal rearrangement of the objects with the minimum dissimilarity. (We use the words *distance* and *dissimilarity* synonymously in this paper.) Thus, the TSP is the same problem as finding an optimal permutation p, except that the TSP finds a cycle through the cities and rearrangement clustering finds a path.

This discrepancy can easily be rectified by adding a *dummy city*. A dummy city is an added city whose distance to each of the other cities is equal to a constant C. The location of the dummy city is the optimal point for breaking the TSP cycle into a path (Lenstra and Kan, 1975). The TSP path is defined as the TSP tour with the dummy city and its two incident edges excluded. The length of

this path is equal to the length of the tour minus $2C$. Following are two critical observations on the above extended TSP.

Lemma 1 *The direct distance between the two cities that are separated by the dummy city is greater than or equal to any of the distances between adjacent pairs of cities on the TSP tour, and the total distance of the TSP path is the smallest possible.*

Proof: We prove the first part of the Lemma by contradiction. Assume that the distance $d(x,y)$ between an adjacent pair of cities, x and y, on the TSP tour T is greater than the direct distance $d(u,v)$ of the two cities, u and v, which are spanned by the dummy city. That is, $d(u,v) < d(x,y)$. Then we can directly connect cities u and v, and insert the dummy city between cities x and y, with a net difference of $d(u,v) - d(x,y) < 0$ to the final tour length. This contradicts the fact that T is a minimum-distance complete tour.

We prove the second part of the Lemma by contradiction also. Assume the length of the TSP path is D and that there exists a path with a length D' where $D' < D$. A cycle that includes the dummy city can be constructed using the new path and its length is $D' + 2C$. This cycle is a feasible solution to the original TSP, but has a length that is shorter than the original TSP solution of $D + 2C$. This contradicts the fact that the original cycle has the minimum possible length. □

2.3 Restricted Partitioning

A well-known approximation algorithm for solving the Euclidean TSP was introduced by Karp (1977) and uses the rule of thumb that every city within the current cluster is visited before moving out of the cluster. This work was cited two decades later and it was proposed that the "inverse" of Karp's algorithm be used to determine clusters *i.e.*, solve the TSP to find the clusters (Alpert and Kahng, 1997). In other words, Alpert and Kahng reinvented rearrangement clustering and referred to it as *restricted partitioning* (RP). They took rearrangement clustering a step further, however, as they introduced an algorithm for automatically determining the locations of cluster boundaries for a given TSP solution and a given number of clusters k. This algorithm computes the boundaries that will yield a set of clusters in which the largest diameter is as small as possible. This partitioning algorithm is based on dynamic programming and runs in $O(kn^3)$ time when applied after solving a TSP tour and $O(kn^2)$ time when applied after solving a TSP path.

Alpert and Kahng applied rearrangement clustering to various problems, including clustering of flower types and clustering cities according to their average temperatures throughout the year (Alpert and Kahng, 1997). However, the main focus of their work was on partitioning circuits for use in the computer-aided design of VLSI circuits (Alpert, 1996).

2.4 Matrix Reordering

Rearrangement clustering was recently reinvented by David Johnson et al. and referred to as *matrix reordering* (Johnson et al., 2004). This work presents a lossless compression strategy for effective storage and access of large, but sparse, boolean matrices on disk. The columns of these matrices are rearranged so as to bring together the one's in the rows. In their paper, the problem was identified as a TSP. This work was demonstrated by compressing matrices within the domains of interactive visualization and telephone call data.

As with Alpert and Kahng's work, rearrangement clustering was taken a step further in Johnson et al.'s paper. For the problems they addressed, finding even an approximate solution for the TSP

was obstructed as many of these problems were too large to fit into main memory. To address this obstacle, they devised a multi-faceted approach that blends classical TSP heuristics with instance-partitioning and sampling. Their approach resulted in significant improvements in access time as well as compression.

2.5 Rearrangement Clustering

Just prior to the work done by Johnson et al., we reinvented rearrangement clustering ourselves (Climer and Zhang, 2004). We were motivated by the need to cluster gene expression data and inspired by the work of Eisen et al. (1998). In their work, Eisen et al. clustered a 2,467 yeast gene set using hierarchical clustering, then arranged the results in a linear order, creating a matrix in which each row corresponded to a gene and each column to an experimental condition. After creating this matrix, they examined it and manually determined cluster boundaries.

Given that the objective was to derive a matrix from which clusters could be identified, we set out to optimize such a matrix. That is when we reinvented rearrangement clustering and identified it as the TSP. However, we soon realized there was a flaw in our approach. This pitfall is described in the next section.

3. Pitfalls

Although rearrangement clustering has been extensively studied and used over the last three decades, there is a serious flaw in previous approaches when applied to data that falls into natural clusters. Consider the example illustrated in Figure 2, where objects have only two features (their horizontal and vertical coordinates) and the dissimilarity between objects is the Euclidean distance. When these objects are rearranged according to the objective in (2), the large cluster on the bottom is broken in half and placed at each end of the ordering. Although objects x and y are very similar, they are separated by 16 objects in two different clusters. We use this simple example as the optimal solution is obvious. However, it is clear that in general, clusters may be broken in pieces in order to minimize the "jumps" to adjacent clusters.

When natural clusters occur, the inter-cluster distances are much greater than the intra-cluster distances. Therefore, the sum of distances between adjacent objects in objective (2) is dominated by the inter-cluster distances. The rearrangement may skew itself in order to minimize these large distances. In the next section, we propose an alternative objective function that addresses this defect and present a technique for resolving this new objective.

The second pitfall applies to the measure of effectiveness (ME) that is used by BEA. ME uses the similarity measure that is defined in equation (3). Two problems associated with ME are that it can fail to ascertain the quality of clustering of non-maximal values and it tends to push small values to the top and bottom of the rearranged matrix. Consider the following examples. Table 1 shows three arrangements of a binary matrix that have the same ME $= 0$, which is the highest value possible. ME fails to distinguish between the levels of clustering of the pairs of zeros. This behavior is not limited to zeros. Table 2 shows three arrangements of a ternary matrix. The first two have ME values of 16, which is optimal. However, the first fails to bring together any of the three identical rows, each containing all ones. Moreover, note how the small values are pushed to the top and bottom of the array. The third arrangement is more likely to be of use for most applications, but it has a sub-optimal ME value of 15.

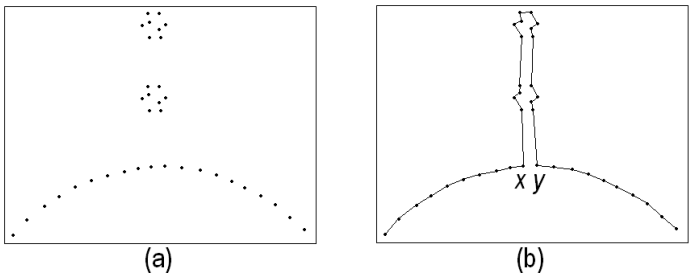


Figure 2: (a) Three clusters. (b) The TSP path specifying the optimal rearrangement. Although x and y are very close, their placement in the rearrangement is far apart.

0	0	0	0	0	0	0	1	0
1	0	1	0	1	0	0	0	0
0	1	0	1	0	1	1	0	1
	(a)			(b)			(c)	

Table 1: Three rearrangements of a matrix with an optimal ME. (a) One pair, (b) two pairs, (c) three pairs of zeros are aligned.

4. Remedies

The second pitfall can be easily rectified by using a different similarity measure. There are a number of similarity measures that are available, including simple Euclidean distances and somewhat more complicated correlation coefficients. In general, the measure that is used can have a profound effect on clustering results and should be selected to suit the problem that is addressed.

We now turn our attention to the first pitfall. A remedy to this pitfall is to omit the inter-cluster distances from the sum in objective (2). We redefine our objective as follows:

$$W(p,k) = min\left(\sum_{i=1}^{k}\sum_{j=u_i}^{v_i-1} d(j,j+1)\right) \tag{4}$$

where u_i is the first item and v_i is the last item of cluster i, and k is the number of clusters. The inner summation of objective (4) is the sum of distances between adjacent rows within a cluster and the outer summation is over all the clusters. In this way, we minimize the intra-cluster distances while disregarding the inter-cluster distances. The inter-cluster distances will assume whatever values best suit the minimization of intra-cluster distances.

This revised problem can be solved using the TSP with a twist. The key to solving this problem lies in Lemma 1. What if we introduce k dummy cities to the TSP representation of the clustering problem? Just as one dummy node cuts the TSP cycle into a path, these dummy cities virtually cut the tour into k segments and form the cluster borders. The distances between pairs of dummy cities are set to infinity, to ensure that no two dummy cities are adjacent on the tour. After this "TSP+k" problem is solved, the dummy cities and their incident edges are removed and replaced

0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0
1	2	1	1	2	1	1	1	1
1	1	1	2	1	1	1	1	1
2	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	2	1
0	0	0	0	0	0	2	1	1
	(a)			(b)			(c)	

Table 2: Three rearrangements of a matrix. (a) ME $= 16$, which is optimal. The three identical rows of ones are all separated. (b) ME $= 16$. Two of the rows of ones are adjacent, but the third is separated by two intervening rows. (c) A clustering that would probably be preferred, but ME $= 15$.

by cluster borders. The lengths of the edges that span the borders, or *borderline* edges, are not of any consequence in the solution of TSP+k. In this way, the TSP+k solution optimizes the intra-cluster distances, while disregarding inter-cluster distances. As a bonus, the cluster boundaries are automatically identified.

Theorem 2 *When there exist k dummy cities, the sum of the lengths of the k paths that are defined by the TSP+k tour is minimized, and every edge in these paths has a distance that is no longer than any of the resulting k borderline edge lengths.*

Proof: No two dummy cities are adjacent on the TSP+k tour, as the distance between them is infinity. Therefore, every TSP+k tour has $2k$ edges of cost C that are adjacent to the dummy cities. The rest of the proof is similar to the proof of Lemma 1. □

In Figure 3, an example of the use of this new objective function is shown. A set of color samples are rearranged, using the intensities of their red, green, and blue components as their features. BEA finds a suboptimal solution as shown in the figure. Solving the TSP with objective (2) leads to splitting the large color cluster in half and inserting the gray color cluster in order to reduce the inter-cluster distance. Note that the color immediately above the gray cluster is very similar to the color immediately below the gray cluster, yet they are far apart in the rearrangement. Moreover, none of the gray colors separating them are nearly as similar to either of them as the two are to each other. This solution is optimal for objective (2). Restricted partitioning (RP) automatically identifies the cluster boundaries as shown in Figure 3(d). RP yields the same linear ordering as TSP+k with $k = 1$. The partitioning minimizes the maximum diameter of the clusters. Notice that this goal splits the gray colors between the two clusters. Finally, by using the new objective in (4) and adding a second dummy city, the inter-cluster distance is ignored and the two clusters are correctly formed as shown in Figure 3(e).

Theorem 2 guarantees the optimality of identifying k clusters for a given k, based on the objective function (4). Assuming that a range of k values is specified, determining the best value for k within this range is the next consideration.

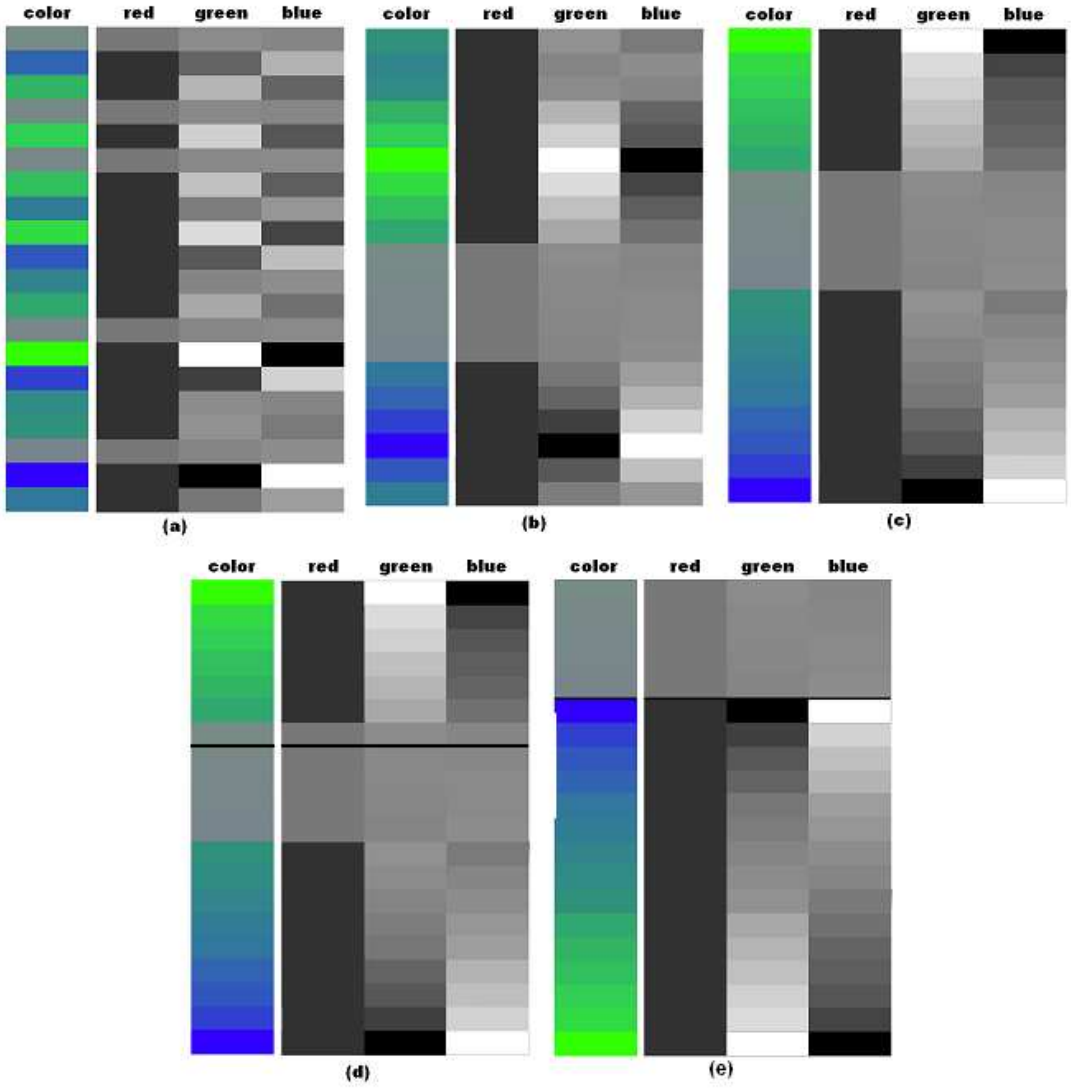


Figure 3: Rearrangement clustering of a set of color samples using their red, green, and blue components as their features. (To view this figure in color, please see the on-line version of this paper.) (a) The initial arrangement. (b) Rearrangement using BEA. (c) Rearrangement using TSP. This rearrangement is optimal for objective (2). (d) Restricted partitioning with $k = 2$. The black line indicates the cluster boundary. This algorithm yields the same ordering as TSP+k with $k = 1$. The gray cluster is split as the partitioning minimizes the maximum diameter. (e) Rearrangement using TSP+k with $k = 2$. The clusters are correctly identified as indicated by the black line.

Theorem 3 *Let*

$$d_{mean} = \frac{\sum_{i=1}^{k} \sum_{j=u_i}^{v_i-1} d(j, j+1)}{(n-k)}. \tag{5}$$

As k increases, d_{mean} is non-increasing.

Proof: d_{mean} is the average intra-cluster distance. In general, as k increases, the membership of clusters may be rearranged to provide the current optimal solution. Let us consider the special case in which the number of clusters is increased from k to $k+1$ and the only change in the TSP tour is that a single edge is replaced by two edges with costs C and the new dummy node. From Theorem 2, we know that the deleted edge must have the maximum distance. Thus, the average distance of the edges cannot increase with this change. Since the TSP finds the minimum tour distance, this property holds when the tour undergoes more than just this one minor change. Therefore, the average distance within clusters is non-increasing. □

The TSP+k algorithm guarantees an optimal rearrangement clustering for a given k. However, as shown by Theorem 3, we must consider desirable qualities other than average intra-cluster distance when determining the best value for k. One approach to handling this problem is to run the algorithm for each value of k in the desired range and use problem-specific information to determine the best clustering result. Another approach is based on the observation that a clustering in which the clusters are well-defined will tend to have large distances between clusters. Therefore, an analysis of the changes in inter-cluster distances may be useful in determining the best k.

When using TSP+k, the resulting clusters are randomly ordered. For some of our experiments, we applied TSP to the border cities to determine an ordering of the clusters. The resulting ordering minimizes the distances between clusters. While this is not necessary for identifying the clusters, it yields useful information about the average distance between clusters, may aid the evaluation of various k values, and may be advantageous for displaying the clustering result. It is also useful if it is desirable to merge small clusters in a post-processing step.

5. Implementation

Our code is composed of two programs and is available on-line.[3] The first program converts a data matrix into a TSP problem, and the second rearranges the rows of the data according to the TSP solution. Any TSP solver can be used. Our method is usable for large clustering problems thanks to recent advances in TSP research.

The TSP has been extensively studied for many decades. A plethora of papers have been written, books have been published (Gutin and Punnen, 2002; Lawler et al., 1985), and websites have been devoted to this problem (Cook, web; Moscato, web; Johnson, web).

There has been a vast amount of research devoted to solving TSPs to guaranteed optimality. For all of our experiments, we used Concorde (Applegate et al., 2001), an award winning TSP solver that has successfully solved a record 24,978-city TSP instance to optimality. The Concorde code is publicly available at *http://www.tsp.gatech.edu//concorde.html.*

There are many applications in which computation time is critical. Fortunately, a great deal of research has been devoted to quickly finding high-quality approximate solutions for the TSP, yielding a wealth of available code (Lodi and Punnen, 2002). These implementations vary drastically in

3. Please find the code at *http://www.climer.us* or *http://www.cse.wustl.edu/∼zhang/projects/software.html.*

running time and quality of solutions. The fastest compute solutions almost as quickly as the input can be read.

Others run more slowly but yield more accurate solutions. For instance, Helsgaun's method (2000), which is based on the Lin-Kernighan heuristic (Lin and Kernighan, 1973), has produced the optimal solution for every optimally solved problem Helsgaun has obtained, including a 15,112-city TSP instance. It has also improved upon the best known solutions for a number of large-scale instances, including a 1,904,711-city problem. Useful comparisons of time versus quality for a number of approximate algorithms are available (Johnson and McGeoch, 2002; Arora, 2002).

In some cases there are an extremely large number of objects that need to be clustered. Propitiously, TSP research has explored the problem of solving very large instances. For example, Johnson et al. presented an algorithm for solving TSP instances that are too large to fit into main memory (Johnson et al., 2004).

In some applications, there may exist a distance function that is not strictly symmetric. That is, $d(i, j)$ may not necessarily be equal to $d(j, i)$. For example, the affinities between amino acid sequences are frequently

asymmetric due to different lengths of the sequences and/or asymmetries in the amino acid substitution matrix. In these cases, TSP+k is still viable. Instead of solving a symmetric TSP, an asymmetric TSP (ATSP) would be computed. There are a number of approximation algorithms for the ATSP and comparisons of these algorithms are available (Johnson et al., 2002). Optimal solutions can be found using branch-and-bound (Carpaneto et al., 1995), branch-and-cut (Fischetti et al., 2002), or cut-and-solve (Climer and Zhang, 2006). Furthermore, symmetric TSP (STSP) codes such as Concorde could be used by converting the ATSP instance into an STSP instance. One way to make this conversion is the *2-node* transformation (Jonker and Volgenant, 1983), in which the number of cities is doubled.

6. Experimental Comparisons and Applications

In this section, we describe four benchmark problems as well as a 2,467-gene expression data clustering problem. The benchmark problems were previously solved using BEA with the ME metric. The clusters were manually identified by domain experts. We present comparisons with TSP+k using the ME metric. We did not compare these results with restricted partitioning (RP). Such a comparison would be misleading as RP minimizes with respect to cluster diameters. In the final part of this section, we present the results of clustering a yeast gene data set. Yeast genes have been extensively studied and functionally related groups have been identified. This research allows objective evaluation of cluster quality. We used these evaluations to compare results from hierarchical clustering, RP, and TSP+k.

6.1 Testbed

Four examples from diverse application domains have been previously presented in the literature. The first three were compared by McCormick et al. (1972) and Lenstra and Kan (1975). They include an airport design example, an aircraft types and functions example, and a marketing applications and techniques example. The fourth example was used by March (1983) for clustering personnel database records.

We also tackled rearrangement clustering of a large set of gene expression data. The data set consists of 2,467 genes in the budding yeast *Saccharomyces cerevisiae* that were studied during the

diauxic shift (DeRisi et al., 1997), mitotic cell division cycle (Spellman et al., 1998), sporulation (Chu et al., 1998), and temperature and reducing shocks (P.T. Spellman, P.O. Brown, and D. Botstein, unpublished results), yielding 79 measurements that are used as the features for the genes. These genes were previously clustered (Eisen et al., 1998) and the data is available at the PNAS website (*http://www.pnas.org*).

6.2 Results for Benchmark Problems

In this section, we first compare the old and new objective functions for four problems that have been presented in work by McCormick et al. (1972), Lenstra and Kan (1975), and March (1983). We based our comparisons on the quality of the individual clusters that are identified. Since McCormick et al. and March manually determined clusters based on the ME metric, we also used ME and compared cluster quality using the ME metric for these four problems.

The clusters identified by McCormick et al. (1972) do not strictly partition the objects. There is some overlapping and some objects are left unclustered. Overlapping of clusters is not allowed in most applications and is not addressed by our new objective. For these reasons, our comparisons are based on non-overlapping results.

The marketing example is shown in Figure 4. The rows represent applications and the columns represent various techniques. This is a binary matrix, where a one indicates that a technique has been shown to be useful for an application and a zero indicates that it has not been useful. This is the only example we present in which it is desirable to find clusters for both the columns and the rows. The ME for the entire matrix is equal whether approximately solved by BEA or optimally solved as a TSP with $k = 1$. When clustering was performed on the techniques (columns), TSP+k with $k = 17$ identified the same three clusters that were identified by McCormick et al. (1972). When TSP+k was used to cluster the applications (rows), it identified clusters with two, three, and four elements, respectively. Notice that McCormick *et al.* identified three clusters, which overlapped for the applications (rows) clustering. We used the four- and three-element clusters that were not overlapping for comparisons. The four-element cluster was the same for both algorithms. However, the three-element clusters differed by one application, *i.e.* BEA grouped together 'sales forecasting', 'brand strategy', and 'advertising research' while TSP+k substituted 'pricing strategy' for 'sales forecasting'. Computing the ME for the three applications yielded a value of 8 for BEA and 10 for TSP+k, revealing that, based on ME, the cluster identified by TSP+k is of higher quality.

The airport design example was presented by McCormick et al. (1972) to demonstrate how BEA can be used for problem decomposition, reducing a large project into a set of small projects with minimal interdependency. The values in the matrix were set to 0, 1, 2, or 3 to indicate no, weak, moderate, or strong dependencies respectively. Figure 5 shows the results for this data. The ME for the entire matrix is 577 for BEA and improved to 580 for TSP with $k = 1$. McCormick et al. (1972) identified eight clusters, with three pairs of clusters overlapping by one object. To make comparisons, we eliminated these overlaps by including the overlapped object in only one cluster, the one that increased the ME value the most. In order to compare the quality of the clusters, we only considered the intra-cluster similarities and ignored the similarities between adjacent clusters. The ME for each cluster was computed and the sum of ME values for the eight clusters is 464 for BEA and 503 for TSP+k with $k = 8$, yielding an improvement in the quality of the clusters.

Figure 6 shows the results for rearrangement clustering of aircraft types based on their functions. Values in the matrix were set from zero to two reflecting the extent that the aircraft can perform the

Figure 4: Marketing techniques and applications example. (a) Initial matrix. (b) BEA clustering. The gray rectangles indicate the clusters identified by McCormick et al. (1972). The black horizontal lines indicate the two non-overlapping clusters that are compared. (c) Optimal clustering with $k = 1$. (d) TSP+k solution. The horizontal lines indicate the two clusters that are compared with the BEA solution. The 4-element cluster is the same for both algorithms. The 3-element clusters differ by one item, yielding ME = 8 for BEA and ME = 10 for TSP+k.

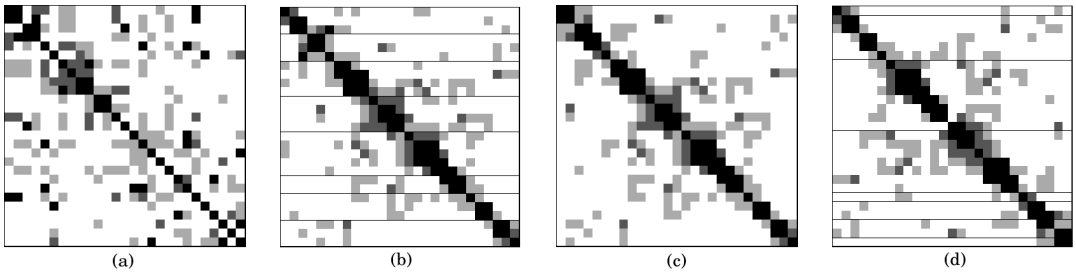


Figure 5: Airport design example. (a) Initial array. (b) BEA clustering with ME = 577 for the entire matrix. The sum of the ME values for the 8 clusters is 464. (c) Optimal clustering with $k = 1$, yielding ME = 580 for the entire matrix. (d) Optimal clustering with $k = 8$. The sum of the ME values for the 8 clusters is 503.

function. The ME for the entire matrix is 1930 for BEA and 1961 for TSP with $k = 1$. The clusters identified by McCormick et al. (1972) had substantial overlapping. We compared the two largest clusters, containing 24 and 14 aircraft, respectively. The two largest clusters for TSP+k with $k = 17$ also contained 24 and 14 aircraft. The sum of the ME values for the two clusters is 1545 for BEA and is 1616 for TSP+k.

Figure 7 shows the results for the personnel database records example from March (1983). The values in this matrix range from one to one hundred. The ME for the entire matrix is 1,791,870 for BEA and 1,836,260 for TSP with $k = 1$. March identified six clusters with no overlapping. The sum of the ME values for these clusters using BEA is 1,533,034 and for TSP+k with $k = 6$ is 1,645,207.

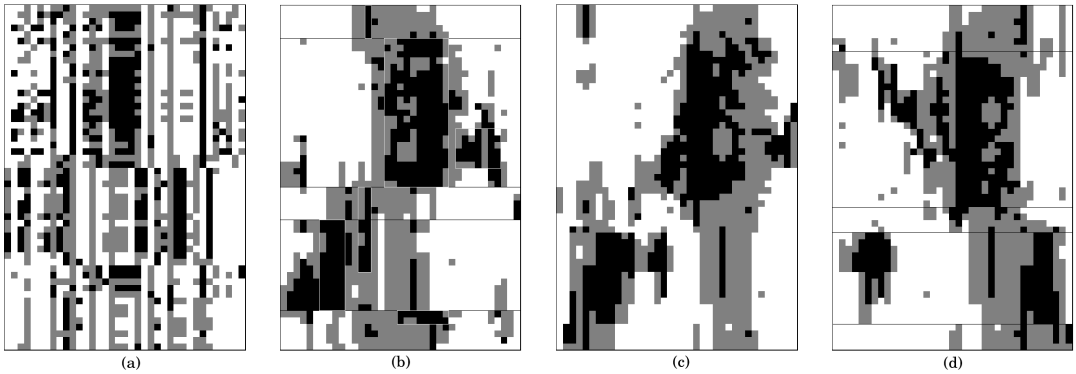


Figure 6: Aircraft types and functions example. (a) Initial array. (b) BEA clustering with ME = 1930 for the entire matrix. The sum of the ME values for the two largest clusters of aircraft is 1545. (c) Optimal clustering with $k = 1$, with ME = 1961 for the entire matrix. (d) Optimal clustering with $k = 17$. The sum of the ME values of the two largest clusters is 1616. These clusters contain the same number of aircraft as the BEA clusters with 24 and 14 aircraft respectively.

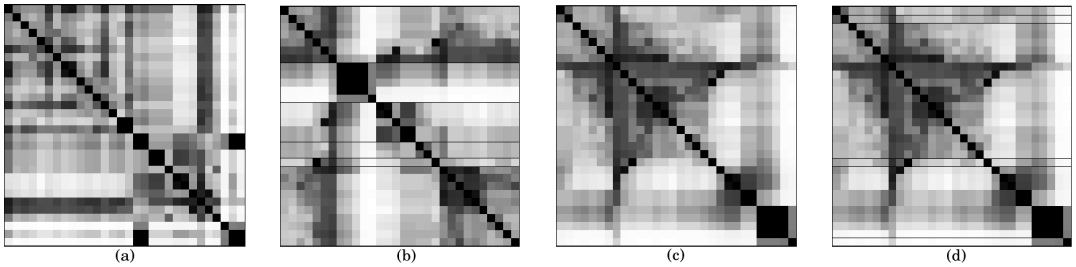


Figure 7: Personnel database records example. (a) Initial array. (b) BEA clustering with ME = 1,791,870 for the entire matrix. The sum of the ME values for the 6 clusters is 1,533,034. (c) Optimal clustering with $k = 1$ and ME = 1,836,260 for the entire matrix. (d) Optimal clustering with $k = 6$. The sum of the ME values for the 6 clusters is 1,645,207.

6.3 Gene Expression Data

In this section we compare rearrangement clustering methods for yeast gene expression data. Yeast genes have been extensively researched and annotated, allowing objective evaluation of the quality of clusters found by each method.

6.3.1 METRICS

In our gene clustering tests, we used the Pearson correlation coefficient (PCC) for the similarity measure. The PCC is defined as follows:

$$s(x,y) = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}} \tag{6}$$

where X and Y are the feature vectors for genes x and y, respectively, and N is the number of features for which both x and y have data tabulated. PCC has been extensively used for gene expression data clustering and was used for comparisons of gene clustering algorithms by Shamir and Sharan (2002). After finding the similarities, we scaled and applied an additive inverse to translate the similarities to nonnegative integral distances.

To objectively evaluate the performance of the various algorithms on gene expression data, we used Gene Ontology (GO) Term Finder (*http://www.yeastgenome.org/*), a tool for finding functionally related groups of yeast genes in a given cluster. This tool calculates a p-value that indicates the likelihood of observing a group of u genes with a particular functional annotation in a cluster containing v genes, given that M genes have this annotation in the total population of N genes. More specifically, the p-value is equal to

$$1 - \sum_{j=0}^{u-1} \frac{\binom{M}{j}\binom{N-M}{v-j}}{\binom{N}{v}}. \tag{7}$$

Notice that the size of the cluster is reflected in calculating the p-value. For instance, if a small and a large cluster both contain a group of u genes with a particular functional annotation, the p-value will be greater for the group in the large cluster as the probability of finding u genes with the given functional annotation is greater in a larger cluster.

6.3.2 RESULTS FOR GENE EXPRESSION DATA

In this section, the results of using three different algorithms for clustering the 2,467 yeast gene data set are presented. The first algorithm uses a hierarchical technique and was presented by Eisen et al. (1998). After applying hierarchical clustering, the results were illustrated in a linear fashion and ten clusters were identified by a domain expert. The identification of clusters was the same as is commonly used in rearrangement clustering. However, the rearrangement of the rows was not based on finding maximum similarity between adjacent rows. Out of the 2,467 genes, 263 were selected for the ten clusters that were identified. It was observed by Eisen et al. that each cluster contained genes that are functionally related.

We ran TSP+k with $k = 100$, $k = 200$, and $k = 300$ on the 2,467-gene data set. Our results are compared with restricted partitioning (RP) (Alpert and Kahng, 1997) and the results from Eisen et al. (1998). We adjusted the k value for RP so as to yield the same number of non-singleton clusters for comparisons. GO Term Finder (*http://www.yeastgenome.org/*) was run for each cluster found in each trial and on the ten clusters identified by Eisen et al. Functional groups found with p-values having orders of magnitude less than or equal to 10^{-7} were designated as "good" functional

	(a)	(b)	(c)	(d)	(e)	(f)
TSP+k	44	56	100	13	129.6	40
RP	44	0	44	13	101.4	28
TSP+k	77	123	200	13	81.8	44
RP	77	0	77	16	42.2	34
TSP+k	109	191	300	16	63.8	41
RP	109	1	110	18	33.1	39
Eisen et al.	10	-	-	9	26.3	49

Table 3: Results for 2,467 yeast gene clustering where "good" functional groups are defined as those with p-values with orders of magnitude $\leq 10^{-7}$. (a) Number of non-singleton clusters. (b) Number of singleton clusters. (c) Value of k used. (d) Number of clusters found containing "good" functional groups. (e) Average size of these "good" clusters. (f) Number of "good" functional groups.

groups. Tables 3 and 4 contain the results of these trials. Figure 8 displays the reordered matrices for TSP+k.

An interesting result of the TSP+k tests was the large number of singletons, as listed in Table 3. In all cases, more than half of the clusters contained singletons. Yet there was not a dominance of clusters containing only two or three genes. For instance, there were only six clusters containing two or three genes when $k = 100$. However, that trial had 56 singletons. Gene expression data is notoriously noisy, so many of the singletons that were found may correspond to outliers in the data. This result suggests that TSP+k may be useful for identifying outliers.

For all the values of k that we tested, the rearrangement clustering algorithms found more "good" clusters than the nine found by Eisen et al. RP found more "good" clusters than TSP+k for the two larger trials. However, the TSP+k clusters were larger in all three trials and a greater number of the 2,467 genes were placed into meaningful clusters. Note that the p-value essentially reflects the *concentration* of related genes within a cluster. Consequently, for a fixed p-value and a particular functional relationship, a larger cluster contains more of these related genes than a smaller cluster.

Table 3 lists the number of "good" functional groups found. TSP+k found more of these groups than RP for each run. Eisen *et al.* found more groups than any of the rearrangement clustering trials. However, the rearrangement clustering algorithms found more distinct functional groups than Eisen et al. when the results from the three trials are combined. Table 4 lists the combined results. Eisen et al. found 49 "good" functional groups. 25 of these groups were missed by RP and 18 were missed by TSP+k. RP found a total of 48 distinct functional groups. 24 of these were missed by Eisen et al. and 11 were missed by TSP+k. Finally, TSP+k found 61 distinct functional groups. Eisen et al. missed 37 of these and RP missed 33. Some of these functional groups were related and appeared in the same cluster. For example, in all but one trial, TSP+k identified a "good" cluster containing functionally related groups of genes involved in carbohydrate transporter activity and six related functions. All seven of these functional groups were overlooked by both RP and Eisen et al.

Tables listing the functional groups for each trial can be found on the web at *http://www.climer.us/cluster/TSPX.htm* and *http://www.climer.us/cluster/RPX.htm*, where X is replaced by the value of k. The results for Eisen et al. can be found at *http://www. climer.us/cluster/eisen.htm.*

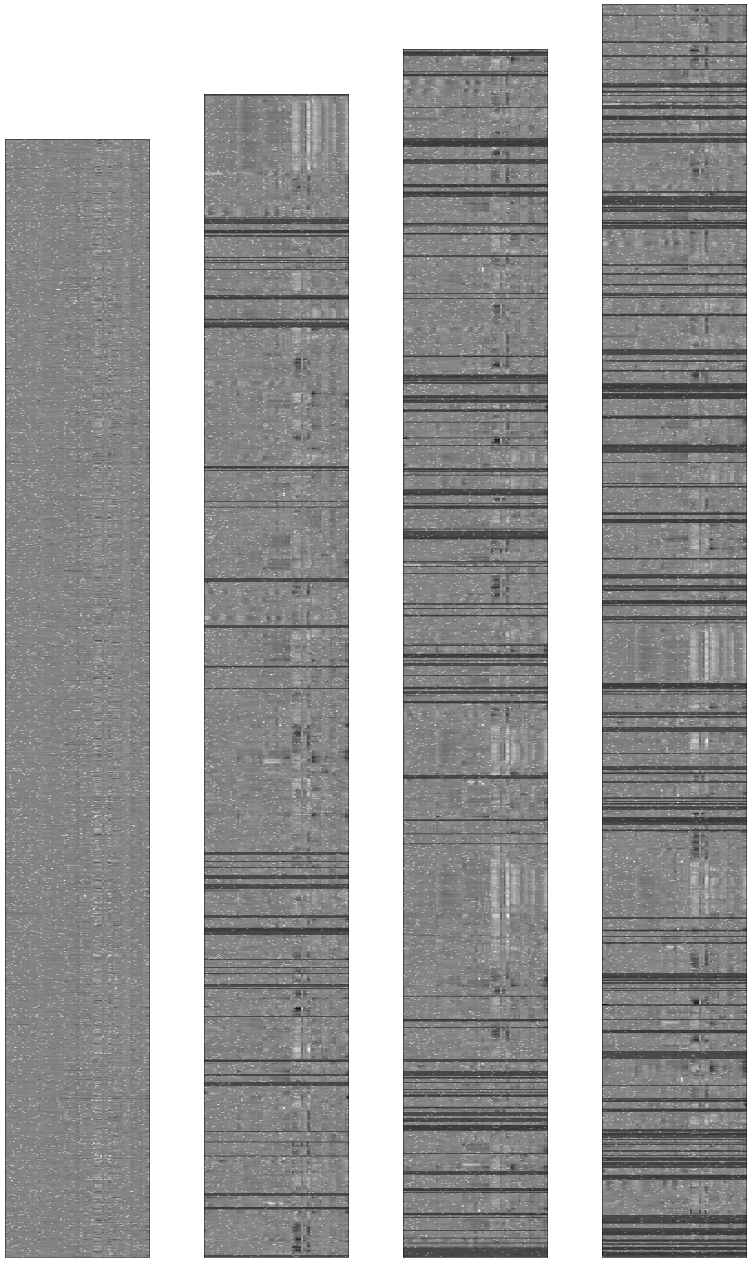


Figure 8: 2,467 yeast gene expression data randomly ordered (left) and rearranged using TSP+k with k equal to 100, 200, and 300. Cluster boundaries are marked by black lines. Missing data values are colored white.

6.3.3 CHANGES IN DOMAIN KNOWLEDGE

About a year ago, we ran GO Term Finder on the clusters found by Eisen et al. and those found using TSP+k. The results are listed in Table 5. It can be expected that a number of additional genes

	Total number of distinct groups	Number missed by Eisen et al.	Number missed by RP	Number missed by TSP+k
Eisen et al.	49	-	25	18
RP	48	24	-	11
TSP+k	61	37	33	-

Table 4: Total number of distinct "good" functional groups found by each algorithm. For each algorithm, the number of groups missed by the other algorithms are shown.

	k	old number of clusters	new number of clusters	old number of groups	new number of groups
TSP+k	100	13	13	39	40
TSP+k	200	12	13	42	44
TSP+k	300	16	16	38	41
Eisen et al.	-	10	9	48	49

Table 5: Comparisons of current GO Term Finder results with those found a year ago. This table lists the number of clusters containing "good" functional groups and the total number of "good" functional groups.

have been annotated during the year, resulting in changes in p-values. In other words, the clusters themselves have not changed during the year, but some of the p-values have, due to additional information found by other means. For Eisen et al., the number of "good" functional groups increased by one. For the three TSP+k trials, the number of "good" functional groups increased by one, two, and three groups respectively. For TSP+k, the number of "good" clusters remained the same for $k = 100$ and $k = 300$ and increased from 12 to 13 for $k = 200$. However, for Eisen et al., the number of "good" clusters decreased from 10 to 9.

Eisen's group published their work in 1998. If they were to redetermine the clusters today, they could use the additional information that has been found experimentally since that time to improve their results. RP and TSP+k do not rely on prior knowledge of functionally related groups to determine the clusters. If they were run in 1998, they would have yielded the same clusters as they do today. Indeed, if they were run before any knowledge of yeast functions was realized, they still would have produced these same clusters.

When using domain experts to determine cluster boundaries, the quality of the results is dependent on the current knowledge of the experts. As more knowledge is acquired in a domain, the clustering results found previously may become obsolete. Automated methods do not rely on current domain knowledge and consequently do not suffer from this antiquation. Moreover, automated methods can be used when there is little or no *a priori* knowledge or when the use of domain experts is impractical. The latter case can occur when the cost of a domain expert is too high or the number of objects is too large.

6.4 Computation Time

The time required to run either TSP+k or RP depends on the algorithm used to solve the TSP. Fast approximate TSP solvers can be used when computation time needs to be minimized.

In the experiments presented in this paper, Concorde was used to solve each instance optimally. (During these tests Concorde aborted early several times and required restarting.) For the 2,467-gene problem, the computation time ranged from 3 to 22 minutes on an Athlon 1.9 MHz processor with two gigabytes memory. An advantage of RP over TSP+k is that a single TSP solution can be used for various values of k. Each partitioning of the TSP path runs in $O(kn^2)$ time.

BEA and hierarchical clustering arrange objects quickly, but both techniques require a domain expert to identify the cluster boundaries. CPU time has become surprisingly inexpensive and a very large number of CPU hours would be equivalent in value to a single hour of a domain expert's time. Moreover, identifying clusters manually requires a fair amount of time. We can be certain that Eisen's group spent substantially more time identifying cluster boundaries than our computer spent solving TSPs. On the other hand, a domain expert simultaneously determines the number of clusters k while identifying cluster boundaries. For the 2,467-gene data set, we arbitrarily set k equal to 100, 200, and 300. Multiple solutions can be advantageous when attempting to maximize the number of functionally related groups as in this example. However, a single solution is frequently desired in many domains. Future work to automatically determine the "best" clustering for a set of k values would maximize the efficiency of rearrangement clustering for these cases. This determination could be based on inter-cluster distances (as discussed in Section 4) and/or other qualities of the clustering results.

7. Discussion

In this section, we examine a couple of considerations that may arise when using rearrangement clustering.

7.1 Number of Features

An interesting property of TSP+k is that the number of features has little impact on the computation time. More features may increase the time required to compute the distances between cities. However, the time required to actually solve the TSP is not directly dependent on the number of features.

While the number of features has little effect on the computation time, it may have bearing on the quality of the results. When the number of features is much greater than the number of objects, *transitivity* of the similarity measure might not be upheld. The transitive property requires that if object x is similar to object y, and y is similar to object z, then x and z are similar. In the previous work we have examined, transitivity is apparently assumed, though it is not explicitly stated. Given an appropriate similarity measure, transitivity might be expected when the number of objects is large in comparison to the number of features. However, care should be used in applying rearrangement clustering when the converse is the case.

7.2 Linearity Requirement

Rearrangement clustering requires a linear ordering of objects. Visualization of complex data is enhanced by arranging objects in this manner (Eisen et al., 1998; McCormick et al., 1972). For

some applications, the objects are actually placed in a linear manner, such as shelf space allocation (Lim et al., 2004). However, for many of the problems that have been previously solved using rearrangement clustering, linearity is not inherently necessary for identifying clusters.

When using rearrangement clustering, there is no quality assurance requiring that the diameters of clusters are less than a given value. This may be a concern for large clusters, in which the first and last objects may be quite dissimilar. On the other hand, this property may be advantageous when it is useful to identify elongated, but contiguous, clusters or irregularly shaped clusters as in Figure 1. Although rearrangement clustering requires the objects be linearly ordered, it doesn't suffer from the drawbacks that can arise when the objective is based on minimizing diameters or minimizing distances of objects from the centers of their respective clusters. In essence, rearrangement clustering yields solutions that tend to be contiguous as each object is a relatively short distance from at least one other object in the same cluster.

Furthermore, the addition of k dummy cities appears to increase the viability of the use of rearrangement clustering for general clustering problems. To gain some intuition about this, consider a traveling salesman who is given k free "jumps" and is required to visit n cities that fall into k distinct clusters. It is reasonable to expect that he will frequently find it most economical to use the free jumps for the long distances between clusters as opposed to using them for intra-cluster hops. When this is the case, TSP+k will correctly identify the k clusters. For example, when TSP+k with $k = 3$ is applied to the example in Figure 2, the three clusters are correctly identified.

As a final note, previous experiments have shown that rearrangement clustering, despite its pitfalls and linearity requirement, has outperformed non-linear-ordering clustering algorithms for applications that do not require linear ordering (Alpert, 1996; Alpert and Kahng, 1997; Liu et al., 2004).

8. Conclusion

Rearrangement clustering has been extensively used in a variety of domains over the last three decades. Yet, the previous approaches have overlooked two serious pitfalls: the summation in the objective function is dominated by inter-cluster distances and the ME metric can fail to appropriately quantify the quality of clustering. These pitfalls can be remedied by using the TSP+k algorithm and an alternate metric. As a bonus, TSP+k provides automatic identification of cluster boundaries.

By translating rearrangement clustering into the TSP, it is possible to take full advantage of the wealth of research that has been invested in optimally and approximately solving TSPs. Generally speaking, BEA is a relatively simple greedy approximation when compared to highly-refined TSP solvers. Moreover, rearrangement clustering can be solved *optimally* for many problems using TSP solvers such as Concorde (Applegate et al., 2001), as illustrated by arranging 2,467 genes in this paper.

Rearrangement clustering has been embraced in many diverse applications. Our new ability to overcome previous pitfalls should result in an even greater usefulness of this popular clustering technique. This usefulness is further enhanced by the fact that TSP+k does not require a domain expert to identify cluster boundaries, thus enabling its use in domains that are not well understood or when experts are unavailable.

Acknowledgments

This research was supported in part by NDSEG and Olin Fellowships and by NSF grants IIS-0196057, ITR/EIA-0113618, and IIS-0535257. We extend thanks to anonymous reviewers of the current paper as well as an earlier version (Climer and Zhang, 2004) who provided valuable comments and insights. We also thank David Applegate, Robert Bixby, Vašek Chvátal, and William Cook for the use of Concorde and Charles Alpert and Andrew Kahng for use of RP. Finally, we thank Michael Eisen for a useful discussion.

References

C. J. Alpert. *Multi-way Graph and Hypergraph Partitioning*. PhD thesis, UCLA, Los Angeles, CA, 1996.

C. J. Alpert and A. B. Kahng. Splitting an ordering into a partition to minimize diameter. *Journal of Classification*, 14:51–74, 1997.

D. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In M. Junger and D. Naddef, editors, *Computational Combinatorial Optimization*, pages 261–304. Springer, 2001.

P. Arabie and L. J. Hubert. The bond energy algorithm revisited. *IEEE Trans. Systems, Man, and Cybernetics*, 20(1):268–74, 1990.

P. Arabie, S. Schleutermann, J. Daws, and L. Hubert. Marketing applications of sequencing and partitioning of nonsymmetric and/or two-mode matrices. In W. Gaul and M. Schader, editors, *Data Analysis, Decision Support, and Expert Knowledge Representation in Marketing*, pages 215–24. Springer Verlag, 1988.

S. Arora. Approximation algorithms for geometric TSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

P. Baldi and G. W. Hatfield. *DNA Microarrays and Gene Expression*. Cambridge University Press, 2002.

G. Carpaneto, M. Dell'Amico, and P. Toth. Exact solution of large-scale, asymmetric Traveling Salesman Problems. *ACM Trans. on Mathematical Software*, 21:394–409, 1995.

H. M. Chan and D. A. Milner. Direct clustering algorithm for group formation in cellular manufacturing. *Journal of Manufacturing Systems*, 1:65–74, 1982.

S. Chu, J. L. DeRisi, M. B. Eisen, J. Mulholland, D. Botstein, P. O. Brown, and I. Herskowitz. The transcriptional program of sporulation in budding yeast. *Science*, 282:699–705, 1998.

S. Climer and W. Zhang. Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, 170:714–738, June 2006.

S. Climer and W. Zhang. Take a walk and cluster genes: A TSP-based approach to optimal rearrangement clustering. In *21st International Conference on Machine Learning (ICML'04)*, pages 169–176, Banff, Canada, July 2004.

W. Cook. Traveling Salesman Problem. *http://www.tsp.gatech.edu*, web.

J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice-Hall, Upper Saddle River, NJ, 2003.

M. B. Eisen, P. T. Spellman, P.O Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. of the Natl. Acad. of Sciences*, 95(25):14863–8, 1998.

M. Fischetti, A. Lodi, and P. Toth. Exact methods for the Asymmetric Traveling Salesman Problem. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

N. Gorla and K. Zhang. Deriving program physical structures using bond energy algorithm. In *Proc. 6th Asia Pacific Software Engineering Conference*, 1999.

G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and its variations*. Kluwer Academic Publishers, Norwell, MA, 2002.

H. A. Hoffer and D. G. Severance. The use of cluster analysis in physical data base design. In *Proceedings of the 1st International Conference on Very Large Data Bases*, pages 69–86, September 1975.

D. S. Johnson. 8th DIMACS implementation challenge. *http://www.research.att.com/ ˜dsj/chtsp/*, web.

D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic, Norwell, MA, 2002.

D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich. Experimental analysis of heuristics for the ATSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

D. S. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *30th Int. Conf. on Very Large Databases (VLDB)*, pages 13–23, 2004.

R. Jonker and T. Volgenant. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2:161–163, 1983.

R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

J. R. King. Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. *International Journal of Production Research*, 18(2):213–32, 1980.

A. Kusiak. Analysis of integer programming formulations of clustering problems. *Image and Vision Computing*, 2:35–40, 1984.

A. Kusiak. Flexible manufacturing systems: A structural approach. *Int. J. Production Res.*, 23: 1057–73, 1985.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem*. John Wiley & Sons, Essex, England, 1985.

J. K. Lenstra. Clustering a data array and the Traveling Salesman Problem. *Operations Research*, 22(2):413–4, 1974.

J. K. Lenstra and A. H. G. Rinnooy Kan. Some simple applications of the Travelling Salesman Problem. *Operational Research Quarterly*, 26(4):717–733, 1975.

A. Lim, B. Rodrigues, and X. Zhang. Metaheuristics with local search techniques for retail shelf-space optimization. *Informs*, 50(1):117 –131, 2004.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

Y. Liu, B. J. Ciliax, A. Pivoshenko, J. Civera, V. Dasigi, A. Ram, R. Dingledine, and S. B. Navathe. Evaluation of a new algorithm for keyword-based functional clustering of genes. In *8th International Conf. on Research in Computational Molecular Biology (RECOMB-04)*, San Diego, CA, March 2004. Poster paper.

A. Lodi and A. P. Punnen. TSP software. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

S. T. March. Techniques for structuring data base records. *Computing Surveys*, 15:45–79, 1983.

F. Marcotorchino. Block seriation problems: A unified approach. *Appl. Stochastic Models and Data Analysis*, 3:73–91, 1987.

W. T. McCormick, P. J. Schweitzer, and T. W. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20:993–1009, 1972.

P. Moscato. TSPBIB. *http://www.densis.fee.unicamp .br/˜ moscato/TSPBIB_home.html*, web.

S. Navathe, S. Ceri, G. Wiederhold, and J. Dou. Vertical partitioning of algorithms for database design. *ACM Trans. Database Syst.*, 9(4):680–710, 1984.

M. T. Ozsu and P. Valduriez. *Principles of distributed database systems*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

A. P. Punnen. The Traveling Salesman Problem: applications, formulations, and variations. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer Academic, Norwell, MA, 2002.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9(12): 3273–97, 1998.

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, 2000.

R. Torres-Velzquez and V. Estivill-Castro. Local search for hamiltonian path with applications to clustering visitation paths. *Journal of the Operational Research Society*, 55(7):737–748, 2004.

Journal of Machine Learning Research 7 (2006) 945–969 Submitted 10/05; Revised 3/06; Published 10/06

Segmental Hidden Markov Models with Random Effects for Waveform Modeling

Seyoung Kim SYKIM@ICS.UCI.EDU
Department of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Padhraic Smyth SMYTH@ICS.UCI.EDU
Department of Computer Science
University of California, Irvine
Irvine, CA 92697-3425, USA

Editor: Sam Roweis

Abstract

This paper proposes a general probabilistic framework for shape-based modeling and classification of waveform data. A segmental hidden Markov model (HMM) is used to characterize waveform shape and shape variation is captured by adding random effects to the segmental model. The resulting probabilistic framework provides a basis for learning of waveform models from data as well as parsing and recognition of new waveforms. Expectation-maximization (EM) algorithms are derived and investigated for fitting such models to data. In particular, the "expectation conditional maximization either" (ECME) algorithm is shown to provide significantly faster convergence than a standard EM procedure. Experimental results on two real-world data sets demonstrate that the proposed approach leads to improved accuracy in classification and segmentation when compared to alternatives such as Euclidean distance matching, dynamic time warping, and segmental HMMs without random effects.

Keywords: waveform recognition, random effects, segmental hidden Markov models, EM algorithm, ECME

1. Introduction

Automatically parsing and recognizing waveforms based on their shape has broad applications, including interpretation and classification of heartbeats in ECG data analysis (Koski, 1996), analysis of waveforms from turbulent flow experiments (Bruun, 1995), and discrimination of nuclear events and earthquakes in seismograph data (Bennett and Murphy, 1986). Waveform analysis has also attracted attention in information retrieval and data mining, with a focus on algorithms that can take a waveform as an input query and search a large database to find similar waveforms that match the query waveform (e.g., Yi and Faloutsos, 2000). Applications include finding temporal patterns in retail time-series data (Agrawal et al., 1993) and fault diagnosis in complex systems (Keogh and Smyth, 1997).

While the human visual system can easily recognize the characteristic signature of a particular waveform shape (a heartbeat waveform for example) the problem can be quite difficult for automated methods. For example, Figure 1 shows a set of time-series waveforms collected during turbulent fluid-flow experiments where the shape of each waveform is determined by the nature of

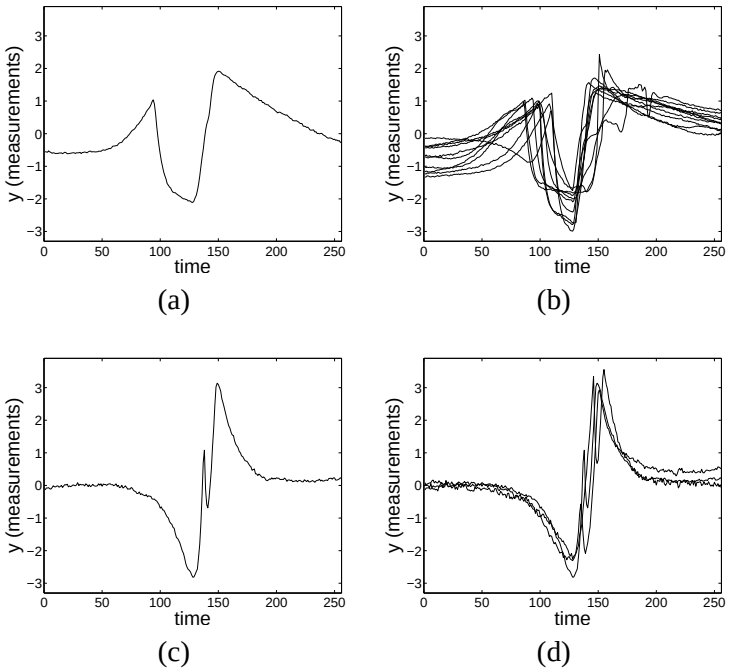


Figure 1: Fluid-flow waveform data: (a) a waveform from the class *splitting* (where the probe splits a bubble), (b) a set of such waveforms, (c) a waveform from the class *glance*, and (d) a set of such waveforms.

interactions between a probe and bubbles in the fluid. Figure 1(a) shows an example waveform from a particular type of interaction. Figure 1(b) shows a whole set of such waveforms that have all been classified (by human experts) as being of the same interaction type. Although all of these waveforms belong to the same interaction class, there is significant variability in shape among those waveforms. The sources of variability include shifts of the locations of prominent features such as peaks, valleys, and plateaus, scaling along the time and amplitude axes, and measurement noise. An example waveform from a different class is shown in Figure 1(c), and a set of such waveforms are shown in Figure 1(d). Again there is significant within-class variability.

In this paper we address the problem of detecting and classifying general classes of waveforms based on their shape and propose a new statistical model that directly addresses within-class shape variability. We will assume in the paper that the waveforms to be analyzed are in the form of "snippets" that have already been extracted from the "background" time-series, e.g., in the form of Figures 1(b) and (d). This assumption can be relaxed—we outline a method for detection of waveforms that are embedded in a time-series in Section 6. We will also assume that the waveforms are being analyzed offline, i.e., that all of the waveform measurements are available at the time of analysis rather than arriving sequentially in an online fashion. The online sequential detection problem can be addressed by generalizing the methods we propose, but we do not pursue online algorithms in this paper.

We will assume that a set of one or more waveforms from a single class are provided a priori (e.g., the data in Figures 1(b) or (d)) and from this data we wish to learn a model for recognition. Hidden Markov models (HMMs) are a broadly useful class of generative models for waveform

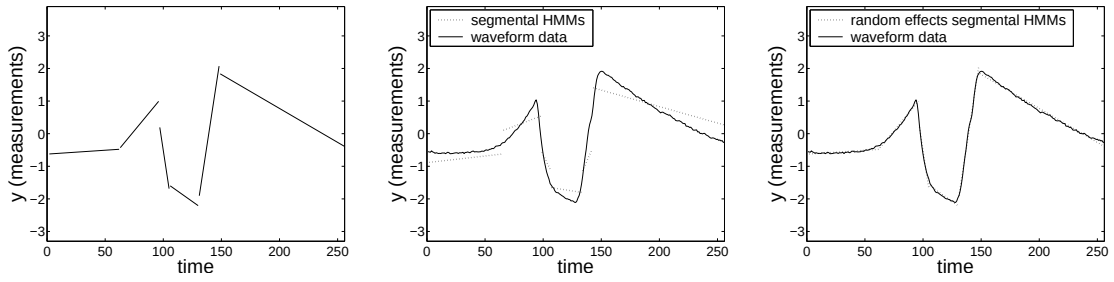


Figure 2: Waveform models: (a) a piecewise linear approximation of the waveform in Figure 1(a), (b) a segmental HMM best fit, and (c) a random effects segmental HMM best fit as described in this paper.

modeling, finding application (for example) in heartbeat monitoring of ECG data (Koski, 1996; Hughes et al., 2003). These models are characterized by (a) a discrete-time finite-state Markov process which is unobserved, and (b) a set of observed measurements at each time t which only depend (stochastically) on the state value at time t. From a shape-modeling viewpoint the standard version of the model generates noisy versions of piecewise constant shapes over time, since the observations within a sequence of states of the same value have constant mean. For waveform modeling, a useful extension is the so-called segmental HMM, originally introduced in the speech recognition (Russell, 1993) and more recently used for more general waveform modeling (Ge and Smyth, 2000). The segmental model allows for the observed data within each segment (a sequence of states with the same value) to follow a general parametric regression form, such as a linear function of time with additive noise. This allows us to model the shape of the waveform directly, in this case as a sequence of piecewise linear components—Figure 2(a) shows a piecewise linear representation of the waveform in Figure 1(a).

A limitation of this particular model is that it assumes that the parameters of the model are fixed. Thus, the only source of variability in an observed waveform arises from variation in the lengths of the segments and observation noise added to the functional form in each segment. The limitation of this can clearly be seen in Figure 2(b), where a segmental HMM has been trained on the data in Figure 1(b) and then used to generate maximum-likelihood estimates of segment boundaries, slopes, and intercepts for the new waveform in Figure 2(b). We can see that the best-fit slopes and intercepts provided by the model do not match the observed data particularly well in each segment, e.g., in the first segment the intercept is clearly too low on the y-axis, in the second segment the slope is too small, and so forth. By using the same fixed parameters for all waveforms, the model cannot fully account for variability in waveform shapes (e.g., as seen in Figure 1(b)).

To address this limitation, in this paper we combine segmental HMMs with random effects models. The general idea of random effects is to allow each group of observations (or each waveform) to have its own parameters that are still coupled together by an overall population prior (Laird and Ware, 1982). By extending the segmental HMM to include random effects, we can allow the slopes and intercepts within each segment of each waveform to vary according to a prior distribution. As illustrated in Figure 3, in the hierarchical setup of our model each waveform (at the bottom level) has its own slope and intercept parameters (as shown at the middle level) that come from a shape

template (at the top level) modeled as a population prior. The parameters of this prior can be learned in an unsupervised manner from data in the form of sets of waveforms. The resulting model can be viewed as a directed graphical model, allowing for application of standard methods for inference and learning (Jordan, 1999; Murphy, 2002). For example, we can in principle learn that the slopes across multiple waveforms for the first segment in Figure 1(b) tend to have a characteristic mean slope and standard deviation. The random effects approach provides a systematic mechanism for allowing variation in "shape space" in a manner that can be parametrized. Figure 2(c) shows a visual example of how a random effects model (constructed from the training data in Figure 1(b)) is used to generate maximum-likelihood estimates of segment boundaries and segment slopes and intercepts for the waveform in Figure 1(a).

Kim et al. (2004) described preliminary results using random effects segmental HMMs for waveform parsing and recognition. A drawback of this earlier approach is the relatively slow convergence of the expectation-maximization (EM) algorithm in learning. This is a result of the large amount of missing information present (due to the random effects component of the model), compared to a standard segmental HMM. In this paper we use the "expectation conditional maximization either" (ECME) algorithm (Liu and Rubin, 1994) for parameter estimation of random effects segmental HMMs. This dramatically speeds up convergence relative to the EM algorithm, making the model much more practical to use for real-world waveform recognition problems.

We begin our discussion by reviewing related work on segmental HMMs and random effects models in Section 2. We introduce segmental HMMs in Section 3. In Section 4, we extend this model to incorporate random effects models, and describe the inference procedure and the EM algorithm for parameter estimation. We also show that the ECME algorithm can be used to significantly speed up the convergence of the EM algorithm. In Section 5, we evaluate our model on two applications involving bubble-probe interaction data and ECG data, and compare random effects segmental HMMs to other waveform-matching algorithms such as Euclidean distance matching, dynamic time warping, and segmental HMMs without random effects. Section 6 contains a brief discussion of possible extensions of the model and final conclusions.

2. Related Work and Contributions

A general approach to waveform recognition is to extract characteristic features from the waveform in the time-domain or the frequency-domain, and then perform classification in the resulting feature-vector space. Examples of this approach include the work of Shimshoni and Intrator (1998) who used neural networks to classify seismic waveforms, and Jankowski and Oreziak (2003) who used support vector machines to classify heartbeats in ECG data. Using classifiers in this manner requires training data from both positive and negative classes as well as the extraction of reliable discriminative features from the raw waveform data. In the approach described in this paper we avoid these requirements by learning models from the positive class only and by modeling the waveform directly in the time-domain without any need for feature extraction. Other techniques have been pursued in the area of waveform query-matching for information retrieval involving time-series data (e.g., Agrawal et al., 1993; Chan and Fu, 1999; Keogh and Pazzani, 2000; Yi and Faloutsos, 2000). These approaches generally focus on the investigation of robust and computationally efficient similarity measures. In contrast, in this paper, we focus on a generative model approach, allowing techniques from statistical learning to be brought to bear. This allows us (for example)

to learn models from data, to handle within-class waveform variability, and to generate maximum-likelihood segmentations of waveforms.

As mentioned in Section 1, standard discrete-time finite-state HMMs are not ideal for modeling waveform shapes since the generative model implicitly assumes a geometric distribution on segment lengths and a piecewise constant shape model. Segmental HMMs relax these modeling constraints by allowing (a) arbitrary distributions on run lengths, and (b) "segment models" (regression models) that allow the mean to be a function of time within each segment. HMMs that allow arbitrary distributions on run lengths (the semi-Markov property in a HMM context) were introduced in the work of Ferguson (1980), Russell and Moore (1985), and Levinson (1986). Deng et al. (1994) and Russell (1993) extended these models to segmental HMMs by modeling dependencies between observations from the same state with a parametric trajectory model that changes over time. Ostendorf et al. (1996) reviewed variations of segmental HMMs from a speech recognition perspective. More recent work includes Achan et al. (2005) and Yun and Oh (2000). Ge and Smyth (2000) introduced the idea of using segmental HMMs for general waveform recognition problems.

The idea of using random effects with segmental HMMs to model parameter variability across waveforms is implicit in the speech recognition work of both Gales and Young (1993) and, later, Holmes and Russell (1999). This work can be viewed as precursors to the more general random effects segmental HMM framework we present in this paper. Gales and Young (1993) used a model with a constant mean per segment, but where the mean values themselves come from a distribution, allowing modeling of variability across different individual speakers. Holmes and Russell (1999) extended this idea to use a linear regression function instead of a constant mean for each segment with a Gaussian prior on the regression parameters (slope and intercept) for each segment. In earlier work (Kim et al., 2004), we noted that Holmes and Russell's model could be formalized within a random effects framework, and derived a more general EM framework for such models, taking advantage of ideas developed separately in speech recognition and in statistics.

In the statistical literature there is a significant body of work on modeling a hierarchical data-generating process with a random effects model and estimating the parameters of this model (Searle et al., 1992). Dempster et al. (1977) sketched the EM algorithm for finding maximum-likelihood estimates for parameters of random effects models. This algorithm was further developed by Dempster et al. (1981), Laird and Ware (1982), and Laird et al. (1987). There appears to be no work in the statistical literature on applying random effects to segmental HMMs.

In this context, the primary contribution of this paper is a general framework for random-effects segmental hidden Markov models. We demonstrate how such models can be used for waveform modeling, recognition, and segmentation, with experimental comparisons of the random effects approach with alternative methods such as dynamic time warping, using two real-world data sets. We extend earlier approaches for learning the parameters of random effects segmental HMMs by deriving a provably correct EM algorithm with monotonic convergence. Both Gales and Young (1993) and Holmes and Russell (1999) derived EM-like optimization algorithms, but their M steps are not in a closed form and use approximate solutions—thus, the monotonic convergence property of EM is not guaranteed in general using their approaches.

We further extend the standard EM algorithm to develop an ECME algorithm for fitting random effects segmental HMMs. The ECME approach significantly reduces the number of iterations required for convergence, relative to EM, while increasing the time complexity per iteration only slightly. For example, as we will discuss later, ECME led to a time-savings of 3 orders of magnitude over the standard EM approach in our experiments. We derive a computationally efficient

inference algorithm (applicable to both EM and ECME) that reduces the time complexity of the forward-backward algorithm by a factor of T^2, where T is the length of a waveform. We also show that this inference algorithm can be applied to full covariance models rather than assuming (as in Holmes and Russell, 1999) that the intercept and slope in the segment distribution are conditionally independent. Since the inference algorithm is used in each iteration of the E step in the EM and ECME iterations, this significantly reduces the overall time complexity of each iteration of EM and ECME.

3. Segmental HMMs

A segmental HMM with M states is described by an $M \times M$ transition matrix, a probability distribution over duration for each state, and a segment model for each state. The transition matrix $\mathbf{A}$ (assumed here to be stationary in time) has entries a_{kl}, namely, the probability of being in state l at time $t+1$ given state k at time t. The initial state distribution can be included in $\mathbf{A}$ as transitions from a special state 0 to each state $k = 1, \ldots, M$. In waveform modeling, we typically constrain the transition matrix to allow only left-to-right transitions and do not allow self-transitions. Thus, there is an ordering on states, each state can be visited at most once, and states can be skipped.

In this paper, we model the duration distribution of state k using a Poisson distribution,

$$P(d-1|\lambda_k) = \frac{e^{-\lambda_k}{\lambda_k}^{d-1}}{(d-1)!} \qquad d = 1, 2, \ldots$$

(shifted to start at $d = 1$ to prevent a silent state). Other choices for the duration distribution could also be used (e.g., Ferguson, 1980; Levinson, 1986). Once the process enters state k, a duration d is drawn, and state k produces a segment of observations of length d from the segment distribution model. In this paper we focus on models with linear functional forms within each segment. We model the rth segment of observations of length d, $\mathbf{y}_r$, generated by state k, as a linear function of time,

$$\mathbf{y}_r = \mathbf{X}_r \beta_k + \mathbf{e}_r \qquad \mathbf{e}_r \sim N_d(\mathbf{0}, \sigma^2 \mathbf{I}_d), \tag{1}$$

where β_k is a 2×1 vector of regression coefficients for the intercept and slope, $\mathbf{e}_r$ is a $d \times 1$ vector of Gaussian noise with variance σ^2 for each component, and $\mathbf{X}_r$ is a $d \times 2$ design matrix consisting of a column of 1's (for the intercept term) and a column of x values representing discrete time values.

In speech recognition using the mid-point of a segment as a parameter in the model instead of intercept has been shown to lead to better speech recognition performance (Holmes and Russell, 1999). Nonetheless, parametrization of the model via the intercept worked well in our experiments, and for this reason we use the intercept in the models discussed in this paper. For simplicity, σ^2 is assumed to be common across all states; again this can be relaxed. We do not enforce continuity of the mean functions (Equation (1)) across segments in the probabilistic model. However, as reported in Section 5, the model without continuity constraints worked well on real-world data in our recognition experiments.

Treating the unobserved state sequences as missing, we can estimate the parameters, $\theta = \{\mathbf{A}, \Lambda = \{\lambda_k | k = 1, \ldots, M\}, \theta_f = \{\beta_k, (\sigma^2) | k = 1, \ldots, M\}\}$, using the EM algorithm, with the forward-backward (F-B) algorithm as a subroutine for inference in the E step (Deng et al., 1994). The F-B algorithm for segmental HMMs, modified from that of standard HMMs to take into account the duration dis-

tribution, recursively computes

$$\begin{aligned}\alpha_t(k) &= P(y_{1:t}, \text{state } k \text{ ends at } t|\theta)\\ \alpha^*_t(k) &= P(y_{1:t}, \text{state } k \text{ starts at } t+1|\theta)\end{aligned} \tag{2}$$

in the forward pass, and

$$\begin{aligned}\beta_t(k) &= P(y_{t+1:T}|\text{state } k \text{ ends at } t, \theta)\\ \beta^*_t(k) &= P(y_{t+1:T}|\text{state } k \text{ starts at } t+1, \theta)\end{aligned} \tag{3}$$

in the backward pass, and returns the results to the M step as a set of sufficient statistics (Rabiner and Juang, 1993).

Inference algorithms for segmental HMMs provide a natural way to evaluate the performance of the model on test data. The F-B algorithm scores a previously unseen waveform $\mathbf{y}$ by calculating the likelihood

$$p(\mathbf{y}|\theta) = \sum_{\mathbf{s}} p(\mathbf{y}, \mathbf{s}|\theta) = \sum_k \alpha_T(k), \tag{4}$$

where $\mathbf{s}$ represents a sequence of unknown state labels for observations $\mathbf{y}$. The Viterbi algorithm can provide a segmentation of a waveform by computing the most likely state sequence (e.g., Figure 2(b)). The addition of duration distributions in segmental HMMs increases the time complexity of both the F-B and Viterbi algorithms from $O(M^2T)$ for standard HMMs to $O(M^2T^2)$, where T is the length of the waveform (i.e., the number of observations).

4. Segmental HMMs with Random Effects

A random effects model is a general statistical framework when the data generation process has a hierarchical structure, coupling a population-level model with individual-level variation. At each level of the generative process, the model defines a prior distribution over the individual group parameters, called random effects, of one level below. The observed data are generated at the bottom of the hierarchy, given parameters drawn from the prior distribution one level above. Typically, the random effects are not observable, so the EM algorithm is a popular approach to learning model parameters from the observed data (Dempster et al., 1981; Laird and Ware, 1982). By combining segmental HMMs and random effects models we can take advantage of the strength of each in waveform modeling. Random effects models add one level of hierarchy to the probabilistic structure of segmental HMMs, defining a population distribution over the possible shapes of waveform segments. Instead of requiring all waveforms to be modeled with a single set of parameters, individual waveforms are allowed to have their own parameters but coupled by a common population prior across all waveforms.

4.1 The Model

Beginning with the segmental HMMs described in Section 3, we add random effects via a new variable $\mathbf{u}^i_r$ to the segment distribution part of the model as follows. Consider the rth segment $\mathbf{y}^i_r$ of length d from the ith individual waveform $\mathbf{y}^i$ generated by state k. Following the discussion in Laird

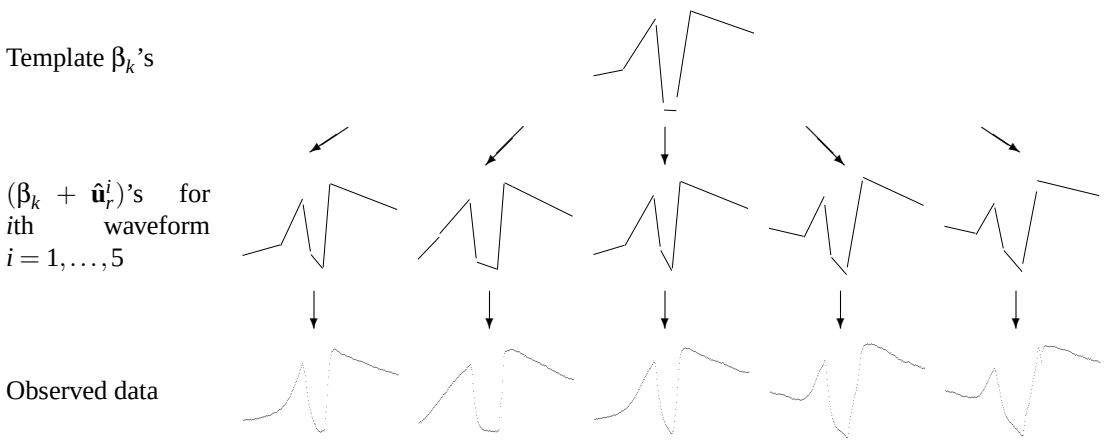


Figure 3: A visual illustration of the random effects segmental HMM, using fluid-flow waveform data as an example (as described in Section 5.1). The top level shows the population level parameters β_k's for the waveform shape. The plots at the bottom level consist of observed data. The plots in the middle level show the posterior estimates (combining both the data and the prior) of $\hat{\mathbf{u}}^i$ and $\hat{\mathbf{s}}^i$, using Equation (8) and the Viterbi algorithm respectively.

and Ware (1982), we describe the generative model as a two-level process. At the bottom level, we model the observed data $\mathbf{y}_r^i$ as

$$\mathbf{y}_r^i = \mathbf{X}_r^i \beta_k + \mathbf{X}_r^i \mathbf{u}_r^i + \mathbf{e}_r^i \qquad \mathbf{e}_r^i \sim N_d(\mathbf{0}, \sigma^2 \mathbf{I}_d), \tag{5}$$

where $\mathbf{e}_r^i$ is the measurement noise, $\mathbf{X}_r^i$ is a $d \times 2$ design matrix for the time measurements corresponding to $\mathbf{y}_r^i$, $(\beta_k + \mathbf{u}_r^i)$ are the regression coefficients, and $1 \le i \le N$ (for N waveforms). β_k represents the mean regression parameters for segment k, and $\mathbf{u}_r^i$ represents the variation in regression (or shape) parameters for the ith individual waveform. At this level, the individual random effects $\mathbf{u}_r^i$ as well as β_k and σ^2 are viewed as parameters. At the top level, $\mathbf{u}_r^i$ is viewed as a random variable with distribution

$$\mathbf{u}_r^i \sim N_2(\mathbf{0}, \Psi_k), \tag{6}$$

where Ψ_k is a 2×2 covariance matrix, and $\mathbf{u}_r^i$ is independent of $\mathbf{e}_r^i$. Notice that this model described by Equations (5) and (6) is equivalent to having $\mathbf{y}_r^i = \mathbf{X}_r^i \beta_r^i + \mathbf{e}_r^i$ with $\beta_r^i \sim N_2(\beta_k, \Psi_k)$. It can be shown that $\mathbf{y}_r^i$ and $\mathbf{u}_r^i$ have the following joint distribution:

$$\begin{pmatrix} \mathbf{y}_r^i \\ \mathbf{u}_r^i \end{pmatrix} \sim N_{d+2}\left(\begin{pmatrix} \mathbf{X}_r^i \beta_k \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{X}_r^i \Psi_k \mathbf{X}_r^{i\prime} + \sigma^2 \mathbf{I}_d & \mathbf{X}_r^i \Psi_k \\ \Psi_k \mathbf{X}_r^{i\prime} & \Psi_k \end{pmatrix} \right). \tag{7}$$

Also, from Equation (7), the posterior distribution of $\mathbf{u}_r^i$ can be written as

$$\mathbf{u}_r^i | \mathbf{y}_r^i, \beta_k, \Psi_k, \sigma^2 \sim N_2\left(\hat{\mathbf{u}}_r^i, \Psi_{\hat{\mathbf{u}}_r^i}\right), \tag{8}$$

where

$$\hat{\mathbf{u}}_r^i = (\mathbf{X}_r^{i\prime} \mathbf{X}_r^i + \sigma^2 (\Psi_k)^{-1})^{-1} \mathbf{X}_r^{i\prime} (\mathbf{y}_r^i - \mathbf{X}_r^i \beta_k), \tag{9}$$

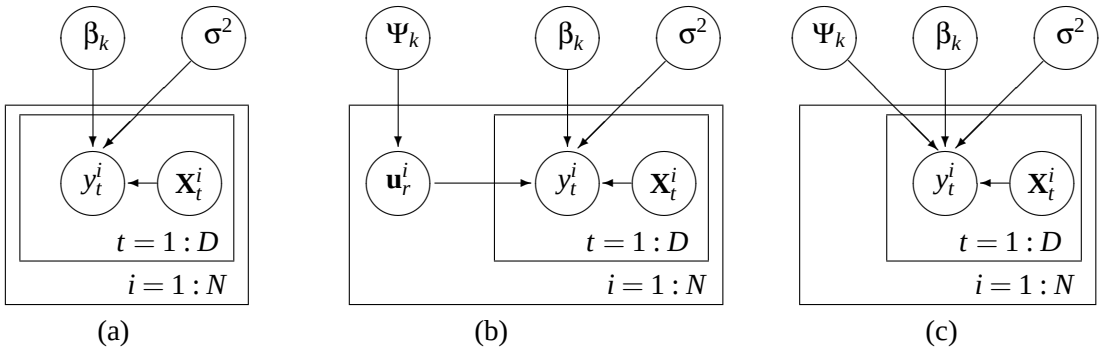


Figure 4: Plate diagrams for the segment distribution part of segmental HMMs and random effects segmental HMMs. (a) segment model in segmental HMMs, (b) a two-stage model with random effects parameters in random effects segmental HMMs, and (c) the model after integrating out random effects parameters from (b).

and

$$\Psi_{\hat{\mathbf{u}}_r^i} = \sigma^2(\mathbf{X}_r^{i\prime}\mathbf{X}_r^i + \sigma^2(\Psi_k)^{-1})^{-1}. \tag{10}$$

In the discussion that follows we use $\mathbf{u}^i$ to denote $\{\mathbf{u}_r^i | r = 1,\ldots,R\}$ given the segmentation $\mathbf{s}^i$ of waveform $\mathbf{y}^i$ into R segments. Similarly, $\hat{\mathbf{u}}^i$ represents $\{\hat{\mathbf{u}}_r^i | r = 1,\ldots,R\}$, given the segmentation $\hat{\mathbf{s}}^i$ of waveform $\mathbf{y}^i$ found by the Viterbi algorithm.

Figure 3 conceptually illustrates the hierarchical setup of the model. The shape template described by the population parameters β_k's is shown at the top of the hierarchy. The plots at the bottom level consist of observed data. The plots at the middle level show the posterior estimates (combining both the data and the prior) of $\hat{\mathbf{u}}^i$ and $\hat{\mathbf{s}}^i$, using Equation (8) and the Viterbi algorithm respectively. From a generative model perspective, the shape templates in the middle row, $(\beta_k + \mathbf{u}_r^i)$'s, $i = 1,\ldots,5$, are generated from the mean shape at the top level by Equation (6). The observed data at the bottom of the hierarchy are modeled as noisy realizations of these individual shape templates. This final data generation process is modeled in Equation (5).

Figure 4 shows plate diagrams for the segment distribution part of segmental HMMs and random effects segmental HMMs, illustrating the generative process for N waveforms, $\mathbf{y}^1,\ldots,\mathbf{y}^N$, under the simplifying assumption that each waveform comes from a single segment of length D corresponding to state k.

4.2 Inference

To handle the random effects component in the F-B and Viterbi algorithms for segmental HMMs, we notice from Equation (7) that the marginal distribution of a segment $\mathbf{y}_r^i$ generated by state k is $N_d(\mathbf{X}_r^i\beta_k, \mathbf{X}_r^i\Psi_k\mathbf{X}_r^{i\prime} + \sigma^2\mathbf{I}_d)$, and that this corresponds to Equation (1) with the covariance matrix $\sigma^2\mathbf{I}_d$ replaced by $(\mathbf{X}_r^i\Psi_k\mathbf{X}_r^{i\prime} + \sigma^2\mathbf{I}_d)$. Replacing the two-level segment distribution with this marginal distribution, and collapsing the hierarchy into a single level, we can use the same F-B and Viterbi algorithm as in segmental HMMs in the marginalized space over the random effects parameters $\mathbf{u}^i$.

The F-B algorithm recursively computes the quantities in Equations (2) and (3). These are then used in the M step of the EM algorithm. The likelihood of a waveform $\mathbf{y}$, given fixed parameters $\theta = \{\mathbf{A}, \Lambda, \theta_f = \{\beta_k, \Psi_k, (\sigma^2) | k = 1, \ldots, M\}\}$, but with states $\mathbf{s}$ and random effects $\mathbf{u}$ unknown, is evaluated as

$$\begin{aligned} p(\mathbf{y}|\theta) &= \sum_{\mathbf{s}} \int p(\mathbf{y}, \mathbf{s}, \mathbf{u}|\theta) d\mathbf{u} \\ &= \sum_{\mathbf{s}} p(\mathbf{y}, \mathbf{s}|\theta) = \sum_{k} \alpha_T(k). \end{aligned} \tag{11}$$

As in segmental HMMs, the Viterbi algorithm can be used as a method to segment a waveform by computing the most likely state sequence.

What appears to make the inference in random effects segmental HMMs computationally much more expensive than in segmental HMMs is the inversion of the $d \times d$ covariance matrix of the marginal segment distribution, $\mathbf{X}_r^i \Psi_k \mathbf{X}_r^{i\prime} + \sigma^2 \mathbf{I}_d$, during the evaluation of the likelihood of a segment. For example, in the F-B algorithm, the likelihood of a segment $\mathbf{y}_r^i$ of length d given state k, $p(\mathbf{y}_r^i | \beta_k, \Psi_k, \sigma^2)$, needs to be calculated for all possible durations d in each of the $\alpha_t(k)$ and $\beta_t(k)$ expressions at each recursion. Naive computation of a segment likelihood, using direct inversion of the $d \times d$ covariance matrix, requires $O(T^3)$ computations, where T is the upper bound for d, leading to an overall time complexity of $O(M^2 T^5)$. This can be computationally impractical for long waveforms with a large value of T (for example, $T = 256$ for the fluid-flow data shown in Figure 1(a)).

In the case of a simpler model with a diagonal covariance matrix for Ψ_k, Holmes and Russell (1999) derived a method for computing the segment likelihood with time complexity $O(M^2 T^3)$. We obtain the same complexity for a more general case with an arbitrary covariance matrix as follows. In discussing computational issues for random effects models, Dempster et al. (1981) suggested an expression for the likelihood that is simple to evaluate. Applying their method to the segment distribution of our model, we rewrite, using Bayes' rule, the likelihood of a segment $\mathbf{y}_r^i$ generated by state k as

$$p(\mathbf{y}_r^i | \beta_k, \Psi_k) = \frac{p(\mathbf{y}_r^i, \mathbf{u}_r^i | \beta_k, \Psi_k, \sigma^2)}{p(\mathbf{u}_r^i | \mathbf{y}_r^i, \beta_k, \Psi_k, \sigma^2)},$$

where the numerator and the denominator of the right-hand side are given as Equations (7) and (8), respectively. The right-hand side of the above equation holds for all values of $\mathbf{u}_r^i$. By setting $\mathbf{u}_r^i$ to $\hat{\mathbf{u}}_r^i$ as in Equation (9), we can simplify the expression for the segment likelihood to

$$p(\mathbf{y}_r^i | \beta_k, \Psi_k) = (2\pi)^{-d/2} \sigma^{-d} |\Psi_{\hat{\mathbf{u}}_r^i}|^{1/2} / |\Psi_k|^{1/2} \exp(-\mathbf{S}_r^i / (2\sigma^2)), \tag{12}$$

where

$$\mathbf{S}_r^i = (\mathbf{y}_r^i - \mathbf{X}_r^i \beta_k - \mathbf{X}_r^i \hat{\mathbf{u}}_r^i)' (\mathbf{y}_r^i - \mathbf{X}_r^i \beta_k - \mathbf{X}_r^i \hat{\mathbf{u}}_r^i) + \sigma^2 \hat{\mathbf{u}}_r^{i\prime} \Psi_k^{(-1)} \hat{\mathbf{u}}_r^i.$$

This can be further simplified using Equation (9):

$$\mathbf{S}_r^i = (\mathbf{y}_r^i - \mathbf{X}_r^i \beta_k)' (\mathbf{y}_r^i - \mathbf{X}_r^i \beta_k - \mathbf{X}_r^i \hat{\mathbf{u}}_r^i).$$

Equation (12) has a form that involves only $O(d)$ computations for each step, where previously this involved $O(d^3)$ computations in the case of the naive approach with matrix inversions. Thus, the

time complexities of the F-B and Viterbi algorithms are reduced to $O(M^2T^3)$. For segmental HMMs this computational complexity can be further reduced to $O(M^2T^2)$ by precomputing the segment likelihood and storing the values in a table (Mitchell et al., 1995). However, this precomputation is not possible with random effects models, leading to the additional factor of T in the complexity term.

4.3 Parameter Estimation

In this section, we describe how to obtain maximum-likelihood estimates of the parameters from a training set of multiple waveforms for a random effects segmental HMM using the EM algorithm. We augment the observed waveform data with both (a) state sequences and (b) random effects parameters (both are considered to be hidden). The log likelihood of the complete data of N waveforms, $D_{\text{complete}} = (\mathbf{Y},\mathbf{S},\mathbf{U}) = \{(\mathbf{y}^1,\mathbf{s}^1,\mathbf{u}^1),\ldots,(\mathbf{y}^N,\mathbf{s}^N,\mathbf{u}^N)\}$, where the state sequence $\mathbf{s}^i$ implies R^i segments in waveform $\mathbf{y}^i$, is defined as:

$$\begin{aligned}
\log L(\theta|D_{\text{complete}}) &= \sum_{i=1}^{N} \log p(\mathbf{y}^i,\mathbf{s}^i,\mathbf{u}^i|\mathbf{A},\Lambda,\theta_f) \\
&= \sum_{i=1}^{N}\sum_{r=1}^{R^i} \log P(s_r^i|s_{r-1}^i,\mathbf{A}) && \text{(13a)} \\
&\quad + \sum_{i=1}^{N}\sum_{r=1}^{R^i} \log P(d_r^i|\lambda_k,k=s_r^i) && \text{(13b)} \\
&\quad + \sum_{i=1}^{N}\sum_{r=1}^{R^i} \log p(\mathbf{y}_r^i|\mathbf{u}_r^i,\beta_k,\sigma^2,k=s_r^i,d_r^i) && \text{(13c)} \\
&\quad + \sum_{i=1}^{N}\sum_{r=1}^{R^i} \log p(\mathbf{u}_r^i|\Psi_k,k=s_r^i). && \text{(13d)}
\end{aligned}$$

As we can see from the above equation, given the complete data, the log-likelihood decouples into four parts Equations (13a)-(13d), where the transition matrix, the duration distribution parameters, the bottom level parameters β_k,σ^2, and the top level random effect parameters $\mathbf{u}_r^i$ appear in each of the four terms. If we had complete data, we could optimize the four sets of parameters independently. When only parts of the data are observed, by iterating between the E step and the M step in the EM algorithm as described below, we can find a solution that locally maximizes the likelihood of the observed data.

4.3.1 E STEP

In the E step, we find the expected log likelihood of the complete data,

$$\mathrm{Q}(\theta^{(t)},\theta) = \mathrm{E}[\log L(\theta|D_{\text{complete}})], \tag{14}$$

with respect to

$$\begin{aligned}
p(\mathbf{S},\mathbf{U}|\mathbf{Y},\theta^{(t)}) &= p(\mathbf{U}|\mathbf{S},\mathbf{Y},\theta^{(t)})P(\mathbf{S}|\mathbf{Y},\theta^{(t)}) \\
&= \prod_{i=1}^{N}\prod_{r=1}^{R^i} p(\mathbf{u}_r^i|s_r^i=k,\mathbf{y}_r^i,\theta^{(t)})P(s_r^i=k|\mathbf{y}_r^i,\theta^{(t)}), && \text{(15)}
\end{aligned}$$

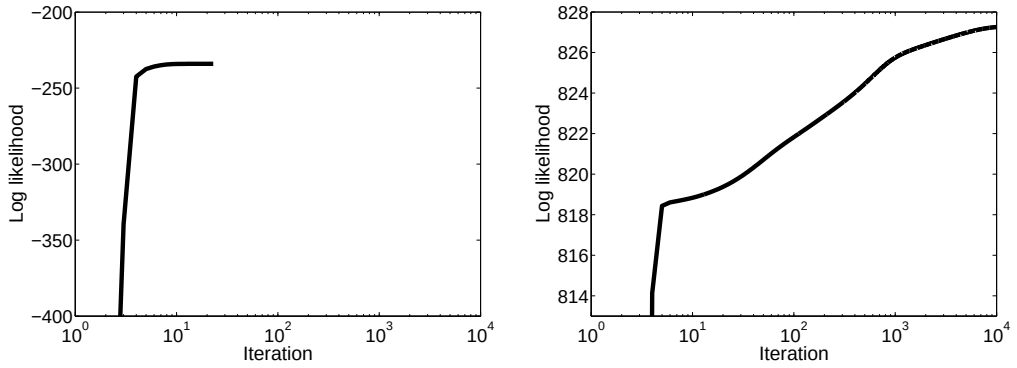


Figure 5: Example of training data log-likelihood convergence as a function of the number of EM iterations, for fluid-flow waveform data, comparing segmental HMMs (on the left) and random effects segmental HMMs (on the right), both using the EM algorithm, *x*-axis on a log-scale.

where $\theta^{(t)}$ is the estimate of the parameter vector from the previous M step of the tth EM iteration. $P(s_r^i = k|\mathbf{y}_r^i, \theta^{(t)})$ in Equation (15) can be obtained from the F-B algorithm. The sufficient statistics, $\mathrm{E}\left[\mathbf{u}_r^i|s_r^i = k, \mathbf{Y}, \theta^{(t)}\right]$ and $\mathrm{E}\left[\mathbf{u}_r^i\mathbf{u}_r^{i\prime}|s_r^i = k, \mathbf{Y}, \theta^{(t)}\right]$, for $P(\mathbf{u}_r^i|s_r^i = k, \mathbf{y}_r^i, \theta^{(t)})$ in Equation (15) can be directly obtained from Equations (9) and (10). The time complexity for an E step is $O(M^2T^3N)$ where N is the number of waveforms (and assuming each waveform is of length T).

4.3.2 M STEP

In the M step, we find the values of the parameters that maximize Equation (14). As we can see from Equations (13a)-(13d) and (14), the optimization problem decouples into four parts, each of which involves a distinct set of parameters. Closed form solutions exist for all of the parameters (the equations are included in Appendix A). The time complexity for each M step is $O(MT^3N)$.

In practice, the algorithm often converges relatively slowly, compared to segmental HMMs, due to the additional missing information in random effects parameters $\mathbf{U}$. Figure 5 shows a typical run of the algorithm. The segmental HMM converges much faster but converges to a lower log-likelihood value. The iterations were halted when the increase of the log-likelihood from one iteration to the next was less than 10^{-5}.

Holmes and Russell (1999) augmented the observed waveform data with state sequences after integrating out the random effects parameters, and used $D_{\text{complete}} = \{\mathbf{Y}, \mathbf{S}\}$ in the E step. In this case the parameters for the segment distribution $\{\beta_k, \sigma^2, \Psi_k\}$ do not decouple in the complete data log-likelihood and there is no closed form solution for those parameters in the M step. Using the approximate solutions proposed in Holmes and Russell means that the monotonic convergence property of EM is no longer guaranteed. In contrast, if we use $D_{\text{complete}} = \{\mathbf{Y}, \mathbf{S}, \mathbf{U}\}$ in the E step as in Equation (14), we can ensure that the algorithm is a proper EM algorithm that always converges to a local maximum of log likelihood.

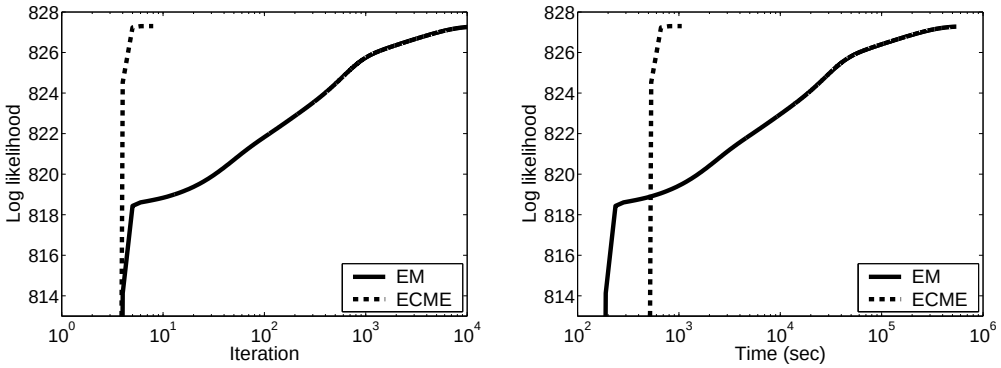


Figure 6: Example of training data log-likelihood convergence as a function of the number of iterations (on the left) and as a function of computation time (on the right), for fluid-flow waveform data (the same data set as in Figure 5), comparing EM vs. ECME for the random effects segmental HMM, x-axis on a log-scale.

4.4 Faster Learning with ECME

As mentioned above, the convergence of the EM algorithm can be very slow especially in the estimation of random effects models. Various extensions of the algorithm have been proposed to speed up the convergence. In the expectation conditional maximization (ECM) algorithm Meng and Rubin (1993) replaced the M step of the EM algorithm with a sequence of $W > 1$ constrained or conditional maximizations (the CM steps). This does not necessarily decrease the number of EM iterations but can significantly reduce the total computation time. Liu and Rubin (1994) further extended the ECM algorithm to the ECME algorithm, reducing both the number of iterations and the total computation time. Both the ECM and the ECME algorithms preserve the property of monotone convergence of the EM algorithm.

More specifically, the CM step of the tth iteration of the ECM algorithm consists of W CM steps. The wth CM step maximizes $Q(\theta^{(t)},\theta)$ under the constraint

$$g_w(\theta) = g_w(\theta^{(t+(w-1)/W)}),$$

where $\theta^{(t+w/W)}$ denotes the value of θ in the wth CM step of the $(t+1)$th iteration and $C = \{g_w(\theta), w = 1,\ldots,W\}$ is a set of W preselected vector functions. These constraints are set so that the maximization is over the full parameter space of θ. In a typical application of the ECM algorithm the set of parameters θ is divided into W subvectors $\theta_1,\ldots,\theta_W$ and in the wth CM step of the tth iteration $Q(\theta^{(t)},\theta)$ is maximized over θ_w. In this case $g_w(\theta)$ is equal to θ_{-w}, the vector of all parameters except for θ_w. In all of the following discussion we assume $g_w(\theta)$ has this particular form.

In the ECME algorithm some of the CM steps of the ECM algorithm are replaced by a maximization of the actual log likelihood subject to the same constraint instead of the expected complete data log likelihood. The large amount of missing information present in the expected complete data log likelihood leads to slow convergence of the EM algorithm (Dempster et al., 1977). The ECME algorithm often speeds up the convergence dramatically by removing the missing information altogether and maximizing the actual log likelihood in some of the CM steps.

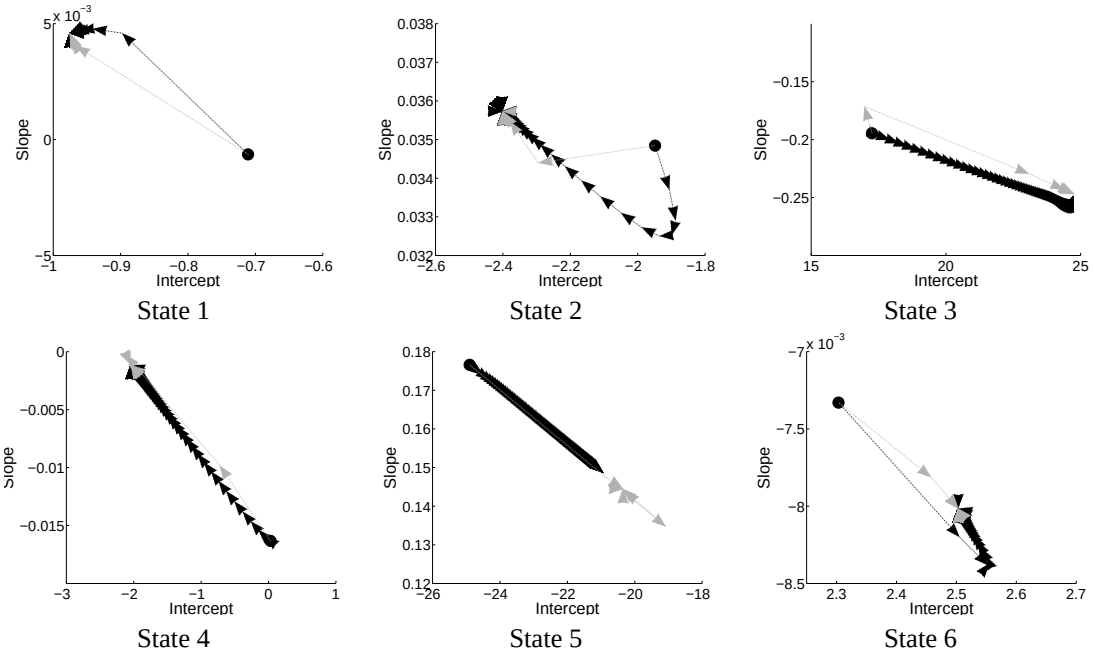


Figure 7: Convergence of β (*x*-axis is intercept, *y*-axis is slope) for fluid-flow data. The starting point is indicated by a circle. Gray arrows represent ECME, black arrows represent EM. An arrow for the parameter values is drawn for each iteration in ECME and for every 100 iterations in EM.

Laird and Ware (1982) first derived an ECME algorithm for random effects models but mistakenly thought it was the EM algorithm. Liu and Rubin (1994) gave a formal description of the ECME algorithm and introduced two different versions of the algorithm for random effects models. The first version has a closed form solution in the CM steps. The other requires an iterative algorithm for one of CM steps, and loses the monotone convergence property of the EM algorithm. Liu and Rubin report slightly faster convergence from the latter, but in our application of the ECME algorithm to random effects segmental HMMs we use the first version with closed form CM steps, thus, retaining the monotone convergence property of EM.

For random effects segmental HMMs we partition the parameters θ into $\theta_1 = \{\mathbf{A}, \Lambda, \Psi_k, \sigma^2 | k = 1,\ldots,M\}$ and $\theta_2 = \{\beta_k | k = 1,\ldots,M\}$ and consider the ECME algorithm with two CM steps for each of the two partitions as follows.

CM step 1: Compute $\mathbf{A}^{(t+1)}$, $\Lambda^{(t+1)}$, $\Psi_k^{(t+1)}$, $k = 1,\ldots,M$, and $(\sigma^2)^{(t+1)}$ as in the M step of the EM algorithm.

CM step 2: Given $\Psi_k^{(t+1)}$, $k = 1,\ldots,M$, and $(\sigma^2)^{(t+1)}$ obtained from CM Step 1, we can integrate out $\mathbf{u}^i$ from Equations (13c)-(13d), and maximize $\sum_{i=1}^{N}\sum_{r=1}^{R^i} \log p(\mathbf{y}_r^i | \beta_k, \Psi_k^{(t+1)}, (\sigma^2)^{(t+1)}, k = s_r^i, d_r^i)$, where $p(\mathbf{y}_r^i | \beta_k, \Psi_k^{(t+1)}, (\sigma^2)^{(t+1)}, k = s_r^i, d_r^i)$ is given as

$$N_d(\mathbf{X}_r^i \beta_k, \mathbf{X}_r^i \Psi_k^{(t+1)} \mathbf{X}_r^{i\prime} + (\sigma^2)^{(t+1)} \mathbf{I}_d).$$

The update equations for $\beta_k^{(t+1)}$, $k = 1, \ldots, M$ are

$$\beta_k^{(t+1)} = \left(\sum_{i=1}^{N} \frac{\sum_t \sum_{d<t} C_{iktd} (\mathbf{X}_{td}^{i\prime} \mathbf{Z}_{td}^{i} \mathbf{X}_{td}^{i})}{P(\mathbf{y}^i | \theta^{(t)})} \right)^{-1} \cdot \left(\sum_{i=1}^{N} \frac{\sum_t \sum_{d<t} C_{iktd} (\mathbf{X}_{td}^{i\prime} \mathbf{Z}_{td}^{i} \mathbf{y}_{td}^{i})}{P(\mathbf{y}^i | \theta^{(t)})} \right),$$

where $\mathbf{X}_{td}^i = \mathbf{X}_{t-d+1:t}^i$ and

$$\mathbf{Z}_{td}^i = (\mathbf{X}_{td}^i \Psi_k^{(t+1)} \mathbf{X}_{td}^{i\prime} + (\sigma^2)^{(t+1)} \mathbf{I}_d)^{-1}.$$

When d is large we can avoid inverting a $d \times d$ matrix to obtain $\mathbf{Z}_{td}^i$ by rewriting this as

$$\mathbf{Z}_{td}^i = \{\mathbf{I}_d - \mathbf{X}_{td}^i ((\sigma^2)^{(t+1)} (\Psi_k)^{-1} + \mathbf{X}_{td}^{i\prime} \mathbf{X}_{td}^i)^{-1} \mathbf{X}_{td}^{i\prime}\} / (\sigma^2)^{(t+1)}.$$

CM step 1 maximizes the expected complete data log likelihood where both state sequences **S** and random effects parameters **U** are considered missing. In CM step 2 the incomplete data log likelihood is augmented only with **S** and then maximized. The computational complexity of the update equation for $\beta_k^{(t+1)}$ in CM step 2 is $O(MT^4N)$ compared to $O(MT^3N)$ for the same parameter in the M step of the EM algorithm. Thus, the overall asymptotic complexity for the CM steps is $O(MT^4N)$, and the ECME algorithm is computationally more expensive in time complexity per iteration than the EM algorithm.

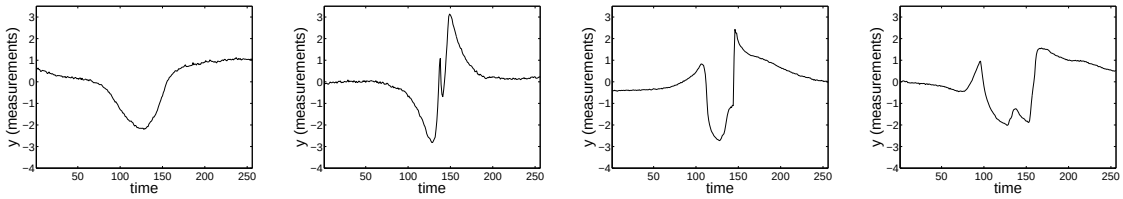
The convergence of the EM and the ECME algorithms for a random effects segmental HMM with six states is shown in Figure 6 for the fluid-flow waveform data described in Section 5.1. The parameters were initialized to the same values for both algorithms and the convergence criterion was set to 10^{-5}. In Figure 6(a) the EM algorithm takes 11506 iterations to converge to roughly the same log-likelihood that the ECME algorithm converges to in only 8 iterations. Each iteration takes 133.3s in the ECME algorithm, versus 47.4s in the EM algorithm, but the overall time to convergence of ECME is still over 3 orders of magnitude faster than EM (as shown in Figure 6(b)).

The convergence trajectories of the 2-dimensional parameters β_k for both algorithms are shown in Figure 7 for each of the six states. The starting values are shown as black circles. Black arrows represent the parameter values of every 100 iterations in the EM algorithm and grey arrows represent the parameters in every iteration of the ECME algorithm. Both Figure 6 and Figure 7 show a dramatic improvement in the speed of convergence of ECME over EM: they both converge to the same solutions in parameter space but ECME converges much more quickly.

5. Experiments

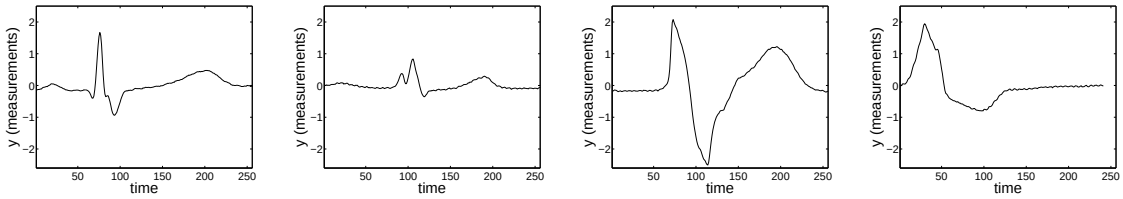
We apply our model to two real-world data sets: (a) hot-film anemometry data in turbulent bubbly fluid-flow and (b) ECG heartbeat data: both are described in more detail below in Section 5.1. In all of our experiments we compare the results from our new segmental HMM with random effects to those obtained using segmental HMMs without random effects. We use several methods to evaluate the models:

Average LogP Score: We compute $\log p(\mathbf{y}|\theta)$ scores (Equations (4) and (11) for each model) for test waveforms **y** to compare how much probability is assigned to new test data by different models. Higher logP scores indicate better predictive power.



(a) (b) (c) (d)

Figure 8: Negative examples in bubble-probe interaction data. (a) no interaction (b) glancing (c) bouncing (d) penetrating.



(a) (b) (c) (d)

Figure 9: Negative examples in ECG data (a) right bundle branch block beat (b) left bundle branch block beat (c) paced beat (d) premature ventricular contraction beat.

Segmentation Quality: To evaluate how well the model can segment test waveforms, we first obtain the segmentations of test waveforms with the Viterbi algorithm, estimate the regression coefficients $\hat{\gamma}$ of each segment, and calculate the mean squared difference between the observed data and $\mathbf{X}\hat{\gamma}$. Good segmentations should produce low scores.

Recognition Accuracy: We use the model learned from a training set of positive examples to recognize waveforms of interest from a test set with both positive and negative exemplars. We compare the results from random effects segmental HMMs with those from dynamic time warping (Keogh and Pazzani, 2000), Euclidean distance matching, and segmental HMMs.

All of the experiments were conducted using cross-validation. The number of segments M for each data set was determined by visual inspection prior to training the models. All waveforms were shifted to have zero mean amplitude before training and testing.

In all experiments reported below, we use the ECME algorithm for training random effects segmental HMMs. The convergence criterion is set to 10^{-5}. We found in our experiments that providing one manually-segmented example is useful in initialization of both EM and ECME—details on initialization are described in Appendix B.

5.1 Data Sets

Below we describe two different data sets that were used as the basis for our experiments.

	Bubble-probe interaction data		ECG data	
	Avg. LogP Score	Avg. Segmentation Error	Avg. LogP Score	Avg. Segmentation Error
Segmental HMMs	-3.25	15.39	-3.12	2.55
Random Effects Segmental HMMs	4.50	1.43	19.63	0.39

Table 1: Average logP scores and segmentation errors for bubble-probe interaction data and ECG data.

5.1.1 BUBBLE-PROBE INTERACTION DATA

Hot-film anemometry is a technique commonly used in turbulent bubbly flow measurements in fluid physics. Different types of interactions between the bubbles and the probe in turbulent gas flow, such as splitting, bouncing, and penetration, lead to characteristic waveform shapes. Automatically detecting the occurrence and types of interactions from such waveforms is a problem of active interest (Bruun, 1995). This recognition task is difficult because of the large variability in the shapes of waveforms within a given class of interactions (e.g., Figure 1(b)), caused by various factors such as velocity fluctuations and different gas fractions during measurement.

We applied our method to individual bubble-probe interaction data. Our data set consisted of 7 waveforms in the class *no interaction* (Figure 8(a)), 5 waveforms in the class *glancing* (Figure 8(b)), 52 waveforms in the class *bouncing* (Figure 8(c)), 8 waveforms in the class *penetration* (Figure 8(d)) and 48 waveforms in the class *splitting* (Figures 1(a) and (b)). Class labels were determined for each interaction based on expert examination of high-speed image recordings of the event obtained simultaneously with the interaction signal (Luther, 2004). Each waveform had 256 data points sampled at 5kHz. We built waveform models for the class of *splitting* interactions, where the probe splits the bubble, and ran a 9-fold cross-validation with 5 waveforms in the training set and 43 waveforms in the test set for each run. The 72 waveforms from the other interactions were used as negative examples in the test set. Given that Figure 2(a) is a reasonable piecewise linear approximation of the general shape, we subjectively chose $M = 6$ as the number of states for both segmental HMMs and random effects segmental HMMs.

5.1.2 ECG DATA

The shape of heartbeat cycles in ECG data can be used to diagnose the heart condition of a patient (Koski, 1996; Hughes et al., 2003). For example, Figure 11 shows the typical shape of normal heartbeats, whereas Figures 9(a)-(d) are taken from a heart experiencing various abnormal conditions. Heartbeats of the same type can vary significantly across individuals in terms of the heights and locations of peaks in the shape. Variability can also be found among heartbeats from the same individual although it is lower than across individuals.

For our experiments we used the ECG recordings with a sampling rate of 360 samples per second from the MIT-BIH Arrhythmia database[1]. We selected hour long recordings from 23 subjects and manually extracted two heartbeats of the same type from each subject. Normal heartbeats were

1. http://www.physionet.org/physiobank/database/mitdb/

	Top 10	Top 20
Euclidean distance (using mean distance)	86.7	81.7
Euclidean distance (using minimum distance)	82.2	80.0
Dynamic time warping (using mean distance)	85.6	82.2
Dynamic time warping (using minimum distance)	92.2	82.8
Segmental HMMs	86.7	82.2
Random Effects Segmental HMMs	100.0	95.0

Table 2: Cross-validated recognition accuracy for bubble-probe interaction data on test set. The numbers represent the true positive rates in percentages (%) among the top k waveforms selected by each algorithm.

taken from each of twelve subjects, and similarly, left bundle branch block beats from three subjects, right bundle branch block beats from two subjects, premature ventricular contraction beats from three subjects, and paced beats from three subjects. The lengths of heartbeats varied approximately from 210 to 410 samples. We modeled each normal heartbeat with $M = 9$ segments. We performed a 4-fold cross-validation with 6 normal waveforms from three individuals as a training set for each cross-validation run and the remainder as a test set. Note that the test set contained heartbeats from a different set of individuals than the individuals used to train the model. Segmental HMMs could not be learned for one of the cross-validation runs due to numerical instability (a problem that did not occur with random effects HMMs), so we report results from the remaining three runs of cross-validations for segmental HMMs. The 22 abnormal heartbeats were used as negative examples for the evaluation of recognition accuracy in the test sets.

5.2 Results

In Table 1 we compare the average logP scores of positive test waveforms for segmental HMMs with those for random effects segmental HMMs. The new model produces significantly higher scores for both data sets, indicating that random effects allow segmental HMMs to capture both the typical shape and shape variability.

Table 1 also shows the average segmentation errors for the test waveforms from both models. Adding the random effects component to segmental HMMs reduces the segmentation error roughly by a factor of 10 on both data sets. Segmentation examples are shown in Figure 10 for the bubble-probe interaction data and Figure 11 for the ECG data, where it is apparent that random effects segmental HMMs are more consistent in locating segment boundaries.

To evaluate the recognition accuracy we score both pattern and non-pattern waveforms in the test set using the model for the pattern waveform learned from the training set, and rank the waveforms according to their log probability scores. We also compare probabilistic methods with non-probabilistic scoring methods such as Euclidean distance and dynamic time warping. For non-probabilistic methods we compute the distance between a test waveform and each of the N training waveforms, and use both the average and minimum of the N distances as a score for that test waveform. The percentages of the true positives in the top 10 and 20 waveforms from bubble-probe interaction data are reported in Table 2. Random effects segmental HMMs give a substantially

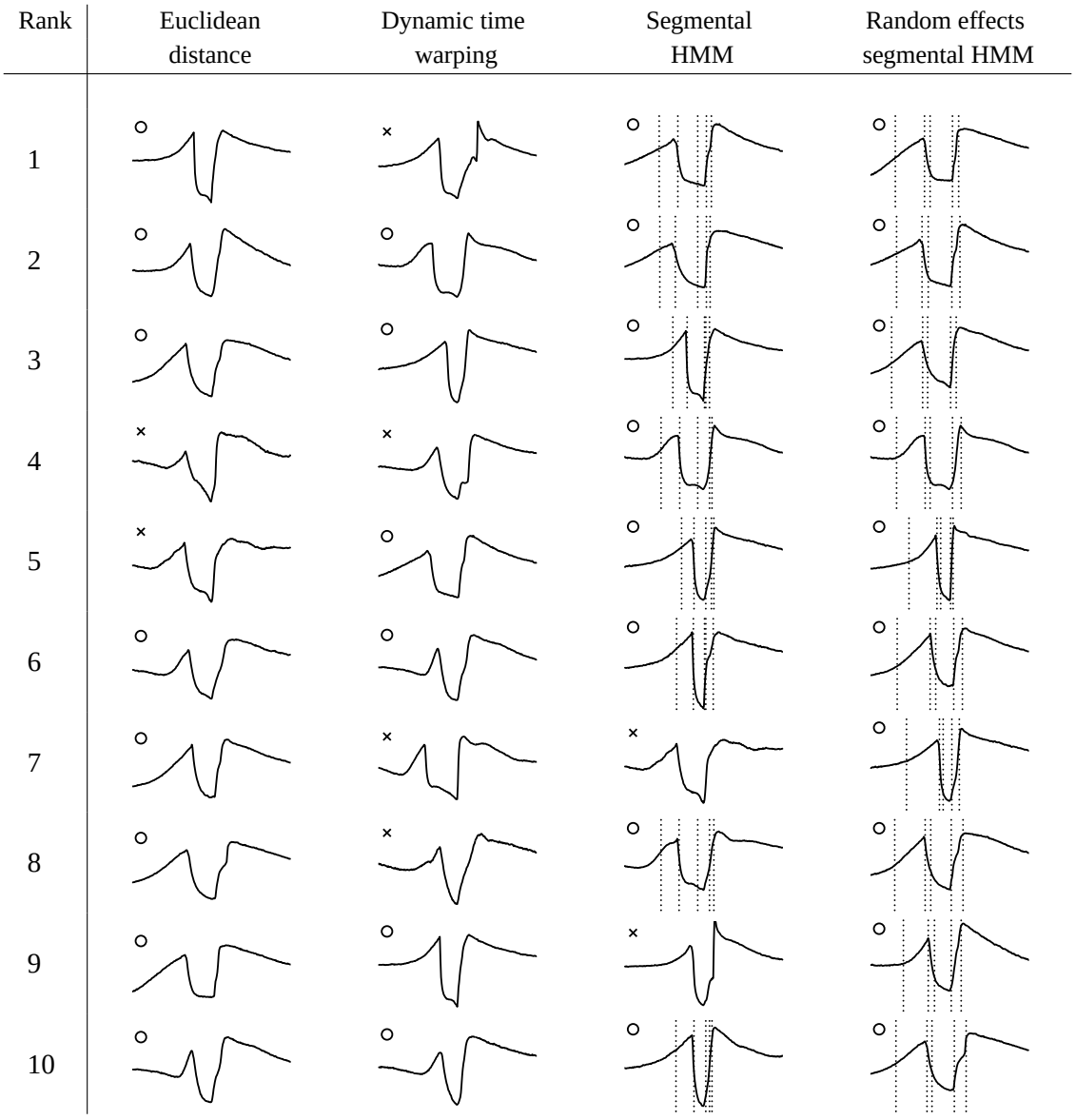


Figure 10: Top 10 waveforms found by four different algorithms in bubble-probe interaction data. ‘o’s are true positives and ‘x’s are false positives. Segmentations by the Viterbi algorithm are overlaid on top of the waveforms in the case of true positives for segmental HMMs and random effects segmental HMMs. Segmentations are not produced by the Euclidean distance method or by dynamic time warping.

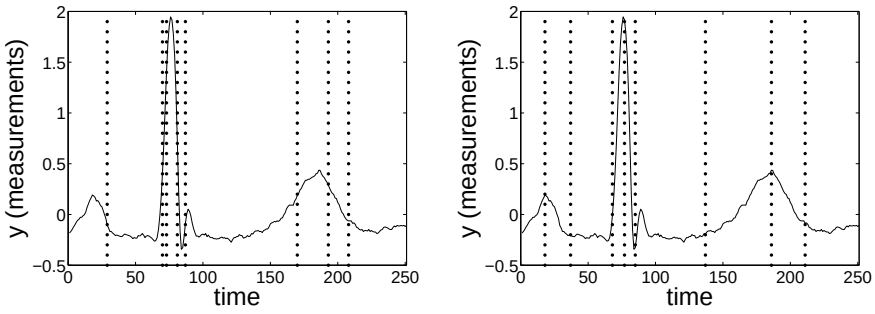


Figure 11: Segmentation of a normal ECG heartbeat by the Viterbi algorithm for segmental HMMs (left) and for random effects segmental HMMs (right).

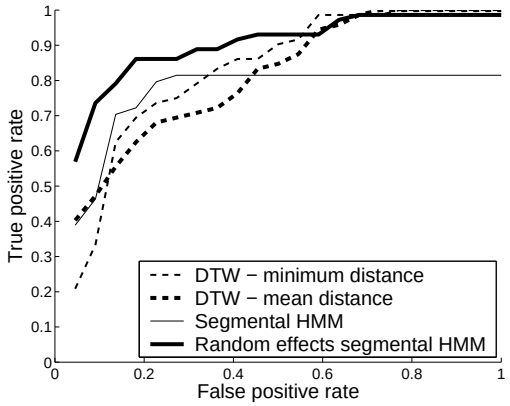


Figure 12: ROC plot for ECG data.

higher accuracy than any of the other methods. Figure 10 shows the top 10 waveforms found by the different methods. All of the false positives are from the interaction class *bouncing*, which is more similar in shape to the class *splitting* than other interaction types. Random effects segmental HMMs can effectively distinguish subtle differences in shape between the pattern that we are modeling and the non-pattern waveforms. Segmentations are overlaid in Figure 10 on the waveforms as found by probabilistic models using the Viterbi algorithm. Such segmentations are not available for dynamic time warping and Euclidean distance methods, providing an additional advantage of using probabilistic models in applications where segmentation is useful.

Figure 12 shows the ROC curves for the ECG data. The results from Euclidean distance are not available for ECG data because the method as implemented requires the length of each waveform sequence to be the same. Random effects segmental HMMs have the highest accuracy, particularly over the range from 0 to 0.5 in terms of fraction of false positives (x-axis) which is typically the range of interest when ranking objects by similarity to a target. A similar result was obtained for bubble-probe interaction data as can be seen in Figure 13.

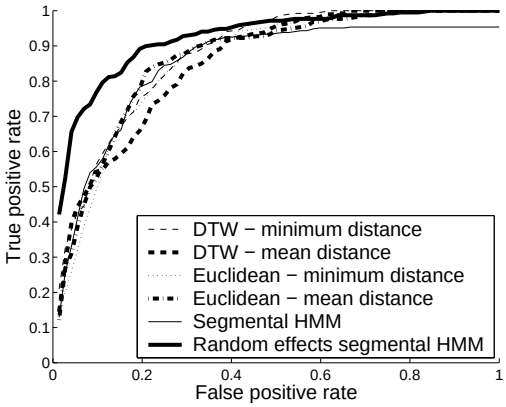


Figure 13: ROC plot for bubble-probe interaction data.

6. Discussions and Conclusions

As noted elsewhere in the paper, the random effects segmental HMM proposed in this paper can be extended in multiple different ways. For example, the parametrization of the segment models as linear functions of time can be generalized directly to any functional form that is linear in the parameters without altering the underlying time complexity of the learning and inference algorithms.

In the results reported here we applied our model to score relatively short waveform "snippets" to detect waveforms that are similar in shape to a query waveform. In order to parse online time-series data and detect "embedded" waveforms relative to a target, a two-state HMM with a pattern state and a background state can be used, where the random effects segmental HMM is embedded inside the pattern state. Each instance of the pattern waveform is allowed to have its own parameters via the random effects mechanism. The background state models any measurements that do not belong to pattern waveforms. A long time-series can then be parsed via the Viterbi algorithm (for example) to segment the series into background and pattern states, where the segments that belong to the pattern state correspond to predicted waveform locations according to the model.

In conclusion, we have proposed a probabilistic model that extends segmental HMMs to include random effects. This model allows an individual waveform to vary its shape in a constrained manner via a prior distribution over individual waveform parameters. The ECME algorithm for learning this model greatly improved the speed of convergence of parameter estimation compared to a standard EM approach. Experimental results support the hypothesis that random effects segmental HMMs perform better in modeling, segmentation, and recognition of waveforms compared both to probabilistic models without random effects and to non-probabilistic methods.

Acknowledgments

This material is based upon work supported by the National Science Foundation under award numbers SCI-0225642 and IIS-0431085. We also thank David Van Dyk for discussions relating to random effects models and EM, Stefan Luther for providing the fluid-flow waveform data, and the referees for providing useful comments that improved the presentation of the paper.

Appendix A: Re-estimation Formulas for EM

The re-estimation formula for the transition probabilities and the duration distribution parameters can be shown to be:

$$a_{kl}^{(t+1)} = \frac{\sum_{i=1}^{N} \frac{1}{P(\mathbf{y}^i|\theta^{(t)})} \sum_t \alpha_t^i(k) a_{kl}^{(t)} \beta_t^{i*}(l)}{\sum_{i=1}^{N} \frac{1}{P(\mathbf{y}^i|\theta^{(t)})} \sum_m \sum_t \alpha_t^i(k) a_{km}^{(t)} \beta_t^{i*}(l)},$$

$$\lambda_k^{(t+1)} = \frac{\sum_{i=1}^{N} \frac{1}{P(\mathbf{y}^i|\theta^{(t)})} \sum_t \sum_d C_{iktd} \cdot (d-1)}{\sum_{i=1}^{N} \frac{1}{P(\mathbf{y}^i|\theta^{(t)})} \sum_t \sum_d C_{iktd}},$$

where

$$C_{iktd} = \alpha_t^{i*}(k) P(d|\lambda_k^{(t)}) p(\mathbf{y}_{t+1:t+d}^i|\theta_{f_k}^{(t)}) \beta_{t+d}^i(k).$$

Using the notation of $\mathbf{X}_{td}^i = \mathbf{X}_{t-d+1:t}^i$ and $\mathbf{y}_{td}^i = \mathbf{y}_{t-d+1:t}^i$, we update the covariance matrix of the top level of the segment distribution model according to

$$\Psi_k^{(t+1)} = \frac{\sum_{i=1}^{N} \frac{\sum_t \sum_{d<t} C_{iktd} \mathrm{E}[\mathbf{u}_k^i \mathbf{u}_k^{i\prime}|\mathbf{Y}, \theta^{(t)}]}{P(\mathbf{y}^i|\theta^{(t)})}}{\sum_{i=1}^{N} \frac{\sum_t \sum_{d<t} C_{iktd}}{P(\mathbf{y}^i|\theta^{(t)})}},$$

and for the bottom level, we re-estimate the parameters using

$$\beta_k^{(t+1)} = \left(\sum_{i=1}^{N} \frac{\sum_t \sum_{d<t} C_{iktd} (\mathbf{X}_{td}^{i\prime} \mathbf{X}_{td}^i)}{P(\mathbf{y}^i|\theta^{(t)})} \right)^{-1} \cdot \left(\sum_{i=1}^{N} \frac{\sum_t \sum_{d<t} C_{iktd} (\mathbf{X}_{td}^{i\prime} (\mathbf{y}_{td}^i - \mathbf{X}_{td}^i \mathrm{E}[\mathbf{u}_k^i|\mathbf{Y}, \theta^{(t)}])}{P(\mathbf{y}^i|\theta^{(t)})} \right)$$

$$(\sigma^2)^{(t+1)} = \frac{\sum_{i=1}^{N} \frac{\sum_{k=1}^{M} \sum_t \sum_{d<t} C_{iktd} \mathrm{E}[\mathbf{D}_k^{i\prime} \mathbf{D}_k^i|\mathbf{Y}, \theta^{(t)}]}{P(\mathbf{y}^i|\theta^{(t)})}}{\sum_{i=1}^{N} \frac{\sum_{k=1}^{M} \sum_t \sum_{d<t} C_{iktd} d}{P(\mathbf{y}^i|\theta^{(t)})}},$$

where

$$\begin{aligned} \mathrm{E}[\mathbf{D}_k^{i\prime} \mathbf{D}_k^i|\mathbf{Y}, \theta^{(t)}] = {} & (\mathbf{y}_{t+1:t+d}^i - \mathbf{X}_{td}^i \beta_k - \mathbf{X}_{td}^i \mathrm{E}[\mathbf{u}_k^i|\mathbf{Y}, \theta^{(t)}])' \cdot (\mathbf{y}_{td}^i - \mathbf{X}_{td}^i \beta_k - \mathbf{X}_{td}^i \mathrm{E}[\mathbf{u}_k^i|\mathbf{Y}, \theta^{(t)}]) \\ & + \mathrm{tr}[\mathbf{X}_{td}^{i\prime} \mathbf{X}_{td}^i \mathrm{Var}(\mathbf{u}_k^i|\mathbf{Y}, \theta^{(t)})]. \end{aligned}$$

Appendix B: Initialization of the EM and ECME Algorithms

Initialization of the EM and ECME algorithms is based on manual segmentation of a single waveform in the training data. The manual segmentation is only used to determine initial values for the parameters (for use in the first E-step), and is not used in any further manner by EM or ECME after this initialization.

Given the manually segmented waveform, the parameters $\mathbf{A}$, θ_d, and β_k's are set to their maximum-likelihood values as estimated from this waveform. The 2×2 covariance matrices Ψ_k's of the random effects component of the model require more than two segmented waveforms in order to obtain maximum-likelihood estimates—thus, their values are initialized in a different manner as follows. The variance term for the slope in Ψ_k's is set to a value generated from a uniform distribution over $[z_{\min}, z_{\max}]$. From preliminary inspection of data $z_{\min}$ and $z_{\max}$ are set to 1 and 10 respectively for bubble-probe interaction data, and 1 and 5 for ECG data. As the state index increases, the values of the intercept parameters in β_k's tend to increase and a small variability in slope leads to a more significant variability in intercept values. To take into account this we initialize the variance for the intercept by sampling a value from the same uniform distribution $[z_{\min}, z_{\max}]$ and multiplying this value by the state index i for that intercept. Given that a positive change in the slope leads to a decreased value of the intercept we initialize the covariance between the slope and intercept to a negative value generated from a uniform distribution over $[z_{\min} \times (-0.1), z_{\max} \times (-0.1)]$. Multiplying $z_{\min}$ and $z_{\max}$ by 0.1 makes the covariance relatively small compared to variances in Ψ_k's and also ensures that the covariance matrices Ψ_k's are positive definite. Finally, we sample the initial value for the noise parameter σ^2 from a uniform distribution over $[1,6]$ for both data sets. This initialization strategy essentially sets the variance parameters Ψ_k's and σ^2 to relatively large initial values and then lets them adjust to the training data.

References

Kannan Achan, Sam Roweis, Aaron Hertzmann, and Brendan Frey. A segment-based probabilistic generative model of speech. In *Proc. of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 221–224, Philadelphia, PA, 2005. IEEE.

Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Efficient similarity search in sequence databases. In *Proc. of the 4th International Conference of Foundations of Data Organization and Algorithms*, pages 69–84, Chicago, IL, 1993. Springer Verlag.

Theron Bennett and John Murphy. Analysis of seismic discrimination using regional data from western United States events. *Bull. Seis. Soc. Am.*, 76:1069–1086, 1986.

Hans Bruun. *Hot Wire Anemometry: Principles and Signal Analysis*. Oxford University Press, Oxford, 1995.

King-pong Chan and Ada Wai-chee Fu. Efficient time series matching by wavelets. In *Proc. of the 15th International Conference on Data Engineering*, pages 126–133, Sydney, Australia, 1999. IEEE Computer Society.

Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39:1–38, 1977.

Arthur Dempster, Donald Rubin, and Robert Tsutakawa. Estimation in covariance components models. *Journal of the American Statistical Association*, 76(374):341–353, 1981.

Li Deng, Mike Aksmanovic, Xiaodong Sun, and Jeff Wu. Speech recognition using hidden Markov models with polynomial regression functions as nonstationary states. *IEEE Trans. Speech Audio Processing*, 2(4):507–520, 1994.

James Ferguson. Variable duration models for speech. In *Proc. of the Symposium on the Application of Hidden Markov Models to Text and Speech*, pages 143–179, Princeton, NJ, 1980. IDA-CRD.

Mark Gales and Steve Young. The theory of segmental hidden Markov models. Technical Report CUED/F-INFENG/TR 133, Cambridge University Engineering Department, 1993.

Xianping Ge and Padhraic Smyth. Deformable Markov model templates for time-series pattern matching. In *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 81–90, Boston, MA, 2000. ACM Press.

Wendy Holmes and Martin Russell. Probabilistic-trajectory segmental HMMs. *Computer Speech and Language*, 13(1):3–37, 1999.

Nicholas Hughes, Lionel Tarassenko, and Stephen Roberts. Markov models for automated ECG interval analysis. In *Advances in Neural Information Processing Systems 16*, pages 611–618, Cambridge, MA, 2003. MIT Press.

Stanislaw Jankowski and Artur Oreziak. Learning system for computer-aided ECG analysis based on support vector machines. *International Journal of Bioelectromagnetism*, 5(1):175–176, 2003.

Michael Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

Eamonn Keogh and Michael Pazzani. Scaling up dynamic time warping for datamining applications. In *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 285–289, Boston, MA, 2000. ACM Press.

Eamonn Keogh and Padhraic Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Proc. of the 3rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 24–30, Newport Beach, CA, 1997. AAAI Press.

Seyoung Kim, Padhraic Smyth, and Stefan Luther. Modeling waveform shapes with random effects segmental hidden Markov models. In *Proc. of the 20th International Conference on Uncertainty in AI*, pages 309–316, Banff, Canada, 2004. AUAI Press.

Antti Koski. Modelling ECG signals with hidden Markov models. *Artificial Intelligence in Medicine*, 8(5):453–471, 1996.

Nan Laird and James Ware. Random-effects models for longitudinal data. *Biometrics*, 38(4):963–974, 1982.

Nan Laird, Nicholas Lange, and Daniel Stram. Maximum likelihood computations with repeated measures: application of the EM algorithm. *Journal of the American Statistical Association*, 82 (397):97–105, 1987.

Stephen Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, 1(1):29–45, 1986.

Chuanhai. Liu and Donald Rubin. The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81(4):633–648, 1994.

Stefan Luther, 2004. personal correspondence.

Xiao-Li Meng and Donald Rubin. Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika*, 80:267–278, 1993.

Carl Mitchell, Mary Harper, and Leah Jamieson. On the computational complexity of explicit duration HMMs. *IEEE Trans. on Speech and Audio Processing*, 3(3):213–217, 1995.

Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference, and Learning*. PhD thesis, University of California, Berkeley, 2002.

Mari Ostendorf, Vassilios Digalakis, and Owen Kimball. From HMMs to segmental models: a unified view of stochastic modeling for speech recognition. *IEEE Trans. on Speech and Audio Processing*, 4(5):360–378, 1996.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, 1993.

Martin Russell. A segmental HMM for speech pattern matching. In *Proc. of the 1993 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 499–502, Minneapolis, MN, 1993. IEEE.

Martin Russell and Roger Moore. Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition. In *Proc. of the 1985 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2376–2379, Tampa, FL, 1985. IEEE.

Shayle Searle, George Casella, and Charles McCulloch. *Variance Components*. Wiley, New York, 1992.

Yair Shimshoni and Nathan Intrator. Classification of seismic signals by integrating ensembles of neural networks. *IEEE Trans. on Signal Processing*, 46:1194–1201, 1998.

Byoung-Kee Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary L_p norms. In *Proc. of the 26th Very Large Data Bases Conference*, pages 385–394, Cairo, Egypt, 2000. Morgan Kaufmann.

Young-Sun Yun and Yung-Hwan Oh. A segmental-feature HMM for speech pattern modeling. *IEEE Signal Processing Letters*, 7(6):135–137, 2000.

Journal of Machine Learning Research 7 (2006) 971–981 Submitted 11/05; Revised 5/06; Published 6/06

Lower Bounds and Aggregation in Density Estimation

Guillaume Lecué LECUE@CCR.JUSSIEU.FR
Laboratoire de Probabilités et Modèles Aléatoires
Université Paris 6
4 place Jussieu, BP 188
75252 Paris, France

Editor: Gábor Lugosi

Abstract

In this paper we prove the optimality of an aggregation procedure. We prove lower bounds for aggregation of model selection type of M density estimators for the Kullback-Leibler divergence (KL), the Hellinger's distance and the L_1-distance. The lower bound, with respect to the KL distance, can be achieved by the on-line type estimate suggested, among others, by Yang (2000a). Combining these results, we state that $\log M/n$ is an optimal rate of aggregation in the sense of Tsybakov (2003), where n is the sample size.

Keywords: aggregation, optimal rates, Kullback-Leibler divergence

1. Introduction

Let $(\mathcal{X},\mathcal{A})$ be a measurable space and ν be a σ-finite measure on $(\mathcal{X},\mathcal{A})$. Let $D_n=(X_1,\dots,X_n)$ be a sample of n i.i.d. observations drawn from an unknown probability of density f on $\mathcal{X}$ with respect to ν. Consider the estimation of f from D_n.

Suppose that we have $M\geq 2$ different estimators $\hat{f}_1,\dots,\hat{f}_M$ of f. Catoni (1997), Yang (2000a), Yang (2000b), Nemirovski (2000), Juditsky and Nemirovski (2000), Yang (2001), Tsybakov (2003), Catoni (2004) and Rigollet and Tsybakov (2004) have studied the problem of model selection type aggregation. It consists in construction of a new estimator $\tilde{f}_n$ (called *aggregate*) which is approximatively at least as good as the best among $\hat{f}_1,\dots,\hat{f}_M$. In most of these papers, this problem is solved by using a kind of cross-validation procedure. Namely, the aggregation is based on splitting the sample in two independent subsamples D_m^1 and D_l^2 of sizes m and l respectively, where $m\gg l$ and $m+l=n$. The size of the first subsample has to be greater than the one of the second because it is used for the true estimation, that is for the construction of the M estimators $\hat{f}_1,\dots,\hat{f}_M$. The second subsample is used for the adaptation step of the procedure, that is for the construction of an aggregate $\tilde{f}_n$, which has to mimic, in a certain sense, the behavior of the best among the estimators $\hat{f}_i$. Thus, $\tilde{f}_n$ is measurable w.r.t. the whole sample D_n unlike the first estimators $\hat{f}_1,\dots,\hat{f}_M$. Actually, Nemirovski (2000) and Juditsky and Nemirovski (2000) did not focus on model selection type aggregation. These papers give a bigger picture about the general topic of procedure aggregation and Yang (2004) complemented their results. Tsybakov (2003) improved these results and formulated the three types of aggregation problems (cf. Tsybakov (2003)).

One can suggest different aggregation procedures and the question is how to look for an optimal one. A way to define optimality in aggregation in a minimax sense for a regression problem is suggested in Tsybakov (2003). Based on the same principle we can define optimality for density

aggregation. In this paper we will not consider the sample splitting and concentrate only on the adaptation step, i.e. on the construction of aggregates (following Nemirovski (2000), Juditsky and Nemirovski (2000), Tsybakov (2003)). Thus, the first subsample is fixed and instead of estimators $\hat{f}_1,\dots,\hat{f}_M$, we have fixed functions $f_1,\dots,f_M$. Rather than working with a part of the initial sample we will use, for notational simplicity, the whole sample D_n of size n instead of a subsample D_l^2.

The aim of this paper is to prove the optimality, in the sense of Tsybakov (2003), of the aggregation method proposed by Yang, for the estimation of a density on $(\mathbb{R}^d,\lambda)$ where λ is the Lebesgue measure on $\mathbb{R}^d$. This procedure is a convex aggregation with weights which can be seen in two different ways. Yang's point of view is to express these weights in function of the likelihood of the model, namely

$$\tilde{f}_n(x)=\sum_{j=1}^{M}\tilde{w}_j^{(n)}f_j(x),\quad \forall x\in\mathcal{X}, \tag{1}$$

where the weights are $\tilde{w}_j^{(n)}=(n+1)^{-1}\sum_{k=0}^{n}w_j^{(k)}$ and

$$w_j^{(k)}=\frac{f_j(X_1)\dots f_j(X_k)}{\sum_{l=1}^{M}f_l(X_1)\dots f_l(X_k)},\ \forall k=1,\dots,n \text{ and } w_j^{(0)}=\frac{1}{M}. \tag{2}$$

And the second point of view is to write these weights as exponential ones, as used in Augustin et al. (1997), Catoni (2004), Hartigan (2002), Bunea and Nobel (2005), Juditsky et al. (2005) and Lecué (2005), for different statistical models. Define the empirical Kullback loss $K_n(f)=-(1/n)\sum_{i=1}^{n}\log f(X_i)$ (keeping only the term independent of the underlying density to estimate) for all density f. We can rewrite these weights as exponential weights:

$$w_j^{(k)}=\frac{\exp(-kK_k(f_j))}{\sum_{l=1}^{M}\exp(-kK_k(f_l))},\quad \forall k=0,\dots,n.$$

Most of the results on convergence properties of aggregation methods are obtained for the regression and the gaussian white noise models. Nevertheless, Catoni (1997, 2004), Devroye and Lugosi (2001), Yang (2000a), Zhang (2003) and Rigollet and Tsybakov (2004) have explored the performances of aggregation procedures in the density estimation framework. Most of them have established upper bounds for some procedure and do not deal with the problem of optimality of their procedures. Nemirovski (2000), Juditsky and Nemirovski (2000) and Yang (2004) state lower bounds for aggregation procedure in the regression setup. To our knowledge, lower bounds for the performance of aggregation methods in density estimation are available only in Rigollet and Tsybakov (2004). Their results are obtained with respect to the mean squared risk. Catoni (1997) and Yang (2000a) construct procedures and give convergence rates w.r.t. the KL loss. One aim of this paper is to prove optimality of one of these procedures w.r.t. the KL loss. Lower bounds w.r.t. the Hellinger's distance and L_1-distance (stated in Section 3) and some results of Birgé (2004) and Devroye and Lugosi (2001) (recalled in Section 4) suggest that the rates of convergence obtained in Theorem 2 and 4 are optimal in the sense given in Definition 1. In fact, an approximate bound can be achieved, if we allow the leading term in the RHS of the oracle inequality (i.e. in the upper bound) to be multiplied by a constant greater than one.

The paper is organized as follows. In Section 2 we give a Definition of optimality, for a rate of aggregation and for an aggregation procedure, and our main results. Lower bounds, for different loss functions, are given in Section 3. In Section 4, we recall a result of Yang (2000a) about an exact oracle inequality satisfied by the aggregation procedure introduced in (1).

2. Main Definition and Main Results

To evaluate the accuracy of a density estimator we use the Kullback-Leibler (KL) divergence, the Hellinger's distance and the L_1-distance as loss functions. The *KL divergence* is defined for all densities f, g w.r.t. a $\sigma-$finite measure ν on a space $\mathcal{X}$, by

$$K(f|g) = \begin{cases} \int_{\mathcal{X}} \log\left(\frac{f}{g}\right) f d\nu & \text{if } P_f \ll P_g; \\ +\infty & \text{otherwise,} \end{cases}$$

where P_f (respectively P_g) denotes the probability distribution of density f (respectively g) w.r.t. ν. *Hellinger's distance* is defined for all non-negative measurable functions f and g by

$$H(f,g) = \left\| \sqrt{f} - \sqrt{g} \right\|_2,$$

where the L_2-norm is defined by $\|f\|_2 = \left(\int_{\mathcal{X}} f^2(x) d\nu(x)\right)^{1/2}$ for all functions $f \in L_2(\mathcal{X}, \nu)$. The L_1*-distance* is defined for all measurable functions f and g by

$$v(f,g) = \int_{\mathcal{X}} |f-g| d\nu.$$

The main goal of this paper is to find optimal rate of aggregation in the sense of the definition given below. This definition is an analog, for the density estimation problem, of the one in Tsybakov (2003) for the regression problem.

Definition 1 *Take $M \geq 2$ an integer, $\mathcal{F}$ a set of densities on $(\mathcal{X}, \mathcal{A}, \nu)$ and $\mathcal{F}_0$ a set of functions on $\mathcal{X}$ with values in $\mathbb{R}$ such that $\mathcal{F} \subseteq \mathcal{F}_0$. Let d be a loss function on the set $\mathcal{F}_0$. A sequence of positive numbers $(\psi_n(M))_{n\in\mathbb{N}^*}$ is called* ***optimal rate of aggregation of M functions in*** $(\mathcal{F}_0, \mathcal{F})$ ***w.r.t. the loss*** *d if :*

(i) There exists a constant $C < \infty$, depending only on $\mathcal{F}_0, \mathcal{F}$ and d, such that for all functions $f_1, \ldots, f_M$ in $\mathcal{F}_0$ there exists an estimator $\tilde{f}_n$ (aggregate) of f such that

$$\sup_{f\in\mathcal{F}} \left[\mathbb{E}_f \left[d(f, \tilde{f}_n) \right] - \min_{i=1,\ldots,M} d(f, f_i) \right] \leq C\psi_n(M), \quad \forall n \in \mathbb{N}^*. \tag{3}$$

(ii) There exist some functions $f_1, \ldots, f_M$ in $\mathcal{F}_0$ and $c > 0$ a constant independent of M such that for all estimators $\hat{f}_n$ of f,

$$\sup_{f\in\mathcal{F}} \left[\mathbb{E}_f \left[d(f, \hat{f}_n) \right] - \min_{i=1,\ldots,M} d(f, f_i) \right] \geq c\psi_n(M), \quad \forall n \in \mathbb{N}^*. \tag{4}$$

Moreover, when the inequalities (3) and (4) are satisfied, we say that the procedure $\tilde{f}_n$, appearing in (3), is an **optimal aggregation procedure w.r.t. the loss** *d***.**

Let $A > 1$ be a given number. In this paper we are interested in the estimation of densities lying in

$$\mathcal{F}(A) = \{\text{densities bounded by } A\} \tag{5}$$

and, depending on the used loss function, we aggregate functions in $\mathcal{F}_0$ which can be:

1. $\mathcal{F}_K(A) = \{\text{densities bounded by } A\}$ for KL divergence,
2. $\mathcal{F}_H(A) = \{\text{non-negative measurable functions bounded by } A\}$ for Hellinger's distance,
3. $\mathcal{F}_v(A) = \{\text{measurable functions bounded by } A\}$ for the L_1-distance.

The main result of this paper, obtained by using Theorem 5 and assertion (6) of Theorem 3, is the following Theorem.

Theorem 1 *Let $A > 1$. Let M and n be two integers such that $\log M \leq 16(\min(1, A-1))^2 n$. The sequence*

$$\psi_n(M) = \frac{\log M}{n}$$

is an optimal rate of aggregation of M functions in $(\mathcal{F}_K(A), \mathcal{F}(A))$ (introduced in (5)) w.r.t. the KL divergence loss. Moreover, the aggregation procedure with exponential weights, defined in (1), achieves this rate. So, this procedure is an optimal aggregation procedure w.r.t. the KL-loss.

Moreover, if we allow the leading term "$\min_{i=1,\dots,M} d(f, f_i)$", in the upper bound and the lower bound of Definition 1, to be multiplied by a constant greater than one, then the rate $(\psi_n(M))_{n\in\mathbb{N}^*}$ is said "near optimal rate of aggregation". Observing Theorem 6 and the result of Devroye and Lugosi (2001) (recalled at the end of Section 4), the rates obtained in Theorems 2 and 4:

$$\left(\frac{\log M}{n}\right)^{\frac{q}{2}}$$

are near optimal rates of aggregation for the Hellinger's distance and the L_1-distance to the power q, where $q > 0$.

3. Lower Bounds

To prove lower bounds of type (4) we use the following lemma on minimax lower bounds which can be obtained by combining Theorems 2.2 and 2.5 in Tsybakov (2004). We say that d is a semi-distance on Θ if d is symmetric, satisfies the triangle inequality and $d(\theta, \theta) = 0$.

Lemma 1 *Let d be a semi-distance on the set of all densities on $(\mathcal{X}, \mathcal{A}, \nu)$ and w be a non-decreasing function defined on $\mathbb{R}_+$ which is not identically 0. Let $(\psi_n)_{n\in\mathbb{N}}$ be a sequence of positive numbers. Let $\mathcal{C}$ be a finite set of densities on $(\mathcal{X}, \mathcal{A}, \nu)$ such that $card(\mathcal{C}) = M \geq 2$,*

$$\forall f, g \in \mathcal{C}, f \neq g \Longrightarrow d(f, g) \geq 4\psi_n > 0,$$

and the KL divergences $K(P_f^{\otimes n} | P_g^{\otimes n})$, between the product probability measures corresponding to densities f and g respectively, satisfy, for some $f_0 \in \mathcal{C}$,

$$\forall f \in \mathcal{C}, K(P_f^{\otimes n} | P_{f_0}^{\otimes n}) \leq (1/16)\log(M).$$

Then,

$$\inf_{\hat{f}_n} \sup_{f\in\mathcal{C}} \mathbb{E}_f \left[w(\psi_n^{-1} d(\hat{f}_n, f)) \right] \geq c_1,$$

where $\inf_{\hat{f}_n}$ denotes the infimum over all estimators based on a sample of size n from an unknown distribution with density f and $c_1 > 0$ is an absolute constant.

Now, we give a lower bound of the form (4) for the three different loss functions introduced in the beginning of the section. Lower bounds are given in the problem of estimation of a density on $\mathbb{R}^d$, namely we have $\mathcal{X} = \mathbb{R}^d$ and ν is the Lebesgue measure on $\mathbb{R}^d$.

Theorem 2 *Let M be an integer greater than 2, $A > 1$ and $q > 0$ be two numbers. We have for all integers n such that $\log M \leq 16(\min(1, A-1))^2 n$,*

$$\sup_{f_1,\dots,f_M \in \mathcal{F}_H(A)} \inf_{\hat{f}_n} \sup_{f \in \mathcal{F}(A)} \left[\mathbb{E}_f \left[H(\hat{f}_n, f)^q \right] - \min_{j=1,\dots,M} H(f_j, f)^q \right] \geq c \left(\frac{\log M}{n} \right)^{q/2},$$

where c is a positive constant which depends only on A and q. The sets $\mathcal{F}(A)$ and $\mathcal{F}_H(A)$ are defined in (5) when $\mathcal{X} = \mathbb{R}^d$ and the infimum is taken over all the estimators based on a sample of size n.

Proof : For all densities $f_1, \dots, f_M$ bounded by A we have,

$$\sup_{f_1,\dots,f_M \in \mathcal{F}_H(A)} \inf_{\hat{f}_n} \sup_{f \in \mathcal{F}(A)} \left[\mathbb{E}_f \left[H(\hat{f}_n, f)^q \right] - \min_{j=1,\dots,M} H(f_j, f)^q \right] \geq \inf_{\hat{f}_n} \sup_{f \in \{f_1,\dots,f_M\}} \mathbb{E}_f \left[H(\hat{f}_n, f)^q \right].$$

Thus, to prove Theorem 1, it suffices to find M appropriate densities bounded by A and to apply Lemma 1 with a suitable rate.

We consider D the smallest integer such that $2^{D/8} \geq M$ and $\Delta = \{0,1\}^D$. We set $h_j(y) = h(y - (j-1)/D)$ for all $y \in \mathbb{R}$, where $h(y) = (L/D)g(Dy)$ and $g(y) = \mathbb{1}_{[0,1/2]}(y) - \mathbb{1}_{(1/2,1]}(y)$ for all $y \in \mathbb{R}$ and $L > 0$ will be chosen later. We consider

$$f_\delta(x) = \mathbb{1}_{[0,1]^d}(x) \left(1 + \sum_{j=1}^{D} \delta_j h_j(x_1) \right), \quad \forall x = (x_1, \dots, x_d) \in \mathbb{R}^d,$$

for all $\delta = (\delta_1, \dots, \delta_D) \in \Delta$. We take L such that $L \leq D\min(1, A-1)$ thus, for all $\delta \in \Delta$, f_δ is a density bounded by A. We choose our densities $f_1, \dots, f_M$ in $\mathcal{B} = \{f_\delta : \delta \in \Delta\}$, but we do not take all of the densities of $\mathcal{B}$ (because they are too close to each other), but only a subset of $\mathcal{B}$, indexed by a separated set (this is a set where all the points are separated from each other by a given distance) of Δ for the *Hamming distance* defined by $\rho(\delta^1, \delta^2) = \sum_{i=1}^{D} I(\delta_i^1 \neq \delta_i^2)$ for all $\delta^1 = (\delta_1^1, \dots \delta_D^1), \delta^2 = (\delta_1^2, \dots, \delta_D^2) \in \Delta$. Since $\int_{\mathbb{R}} h d\lambda = 0$, we have

$$\begin{aligned} H^2(f_{\delta^1}, f_{\delta^2}) &= \sum_{j=1}^{D} \int_{\frac{j-1}{D}}^{\frac{j}{D}} I(\delta_j^1 \neq \delta_j^2) \left(1 - \sqrt{1 + h_j(x)} \right)^2 dx \\ &= 2\rho(\delta^1, \delta^2) \int_0^{1/D} \left(1 - \sqrt{1 + h(x)} \right) dx, \end{aligned}$$

for all $\delta^1 = (\delta_1^1, \dots, \delta_D^1), \delta^2 = (\delta_1^2, \dots, \delta_D^2) \in \Delta$. On the other hand the function $\varphi(x) = 1 - \alpha x^2 - \sqrt{1+x}$, where $\alpha = 8^{-3/2}$, is convex on $[-1,1]$ and we have $|h(x)| \leq L/D \leq 1$ so, according to Jensen, $\int_0^1 \varphi(h(x))dx \geq \varphi\left(\int_0^1 h(x)dx \right)$. Therefore $\int_0^{1/D} \left(1 - \sqrt{1+h(x)} \right) dx \geq \alpha \int_0^{1/D} h^2(x)dx = (\alpha L^2)/D^3$, and we have

$$H^2(f_{\delta^1}, f_{\delta^2}) \geq \frac{2\alpha L^2}{D^3} \rho(\delta^1, \delta^2),$$

for all $\delta^1, \delta^2 \in \Delta$. According to Varshamov-Gilbert, cf. Tsybakov (2004, p. 89) or Ibragimov and Hasminskii (1980), there exists a $D/8$-separated set, called $N_{D/8}$, on Δ for the Hamming distance such that its cardinal is higher than $2^{D/8}$ and $(0,\dots,0) \in N_{D/8}$. On the separated set $N_{D/8}$ we have,

$$\forall \delta^1, \delta^2 \in N_{D/8}\,, H^2(f_{\delta^1}, f_{\delta^2}) \geq \frac{\alpha L^2}{4D^2}.$$

In order to apply Lemma 1, we need to control the KL divergences too. Since we have taken $N_{D/8}$ such that $(0,\dots,0) \in N_{D/8}$, we can control the KL divergences w.r.t. P_0, the Lebesgue measure on $[0,1]^d$. We denote by P_δ the probability of density f_δ w.r.t. the Lebesgue's measure on $\mathbb{R}^d$, for all $\delta \in \Delta$. We have,

$$\begin{aligned} K(P_\delta^{\otimes n}|P_0^{\otimes n}) &= n\int_{[0,1]^d} \log\left(f_\delta(x)\right) f_\delta(x)dx \\ &= n\sum_{j=1}^{D} \int_{\frac{j-1}{D}}^{j/D} \log\left(1+\delta_j h_j(x)\right)\left(1+\delta_j h_j(x)\right)dx \\ &= n\left(\sum_{j=1}^{D}\delta_j\right)\int_0^{1/D} \log(1+h(x))(1+h(x))dx, \end{aligned}$$

for all $\delta = (\delta_1,\dots,\delta_D) \in N_{D/8}$. Since $\forall u > -1, \log(1+u) \leq u$, we have,

$$K(P_\delta^{\otimes n}|P_0^{\otimes n}) \leq n\left(\sum_{j=1}^{D}\delta_j\right)\int_0^{1/D}(1+h(x))h(x)dx \leq nD\int_0^{1/D} h^2(x)dx = \frac{nL^2}{D^2}.$$

Since $\log M \leq 16(\min(1,A-1))^2 n$, we can take L such that $(nL^2)/D^2 = \log(M)/16$ and still having $L \leq D\min(1,A-1)$. Thus, for $L = (D/4)\sqrt{\log(M)/n}$, we have for all elements δ^1, δ^2 in $N_{D/8}$, $H^2(f_{\delta^1}, f_{\delta^2}) \geq (\alpha/64)(\log(M)/n)$ and $\forall \delta \in N_{D/8}\,, K(P_\delta^{\otimes n}|P_0^{\otimes n}) \leq (1/16)\log(M)$.

Applying Lemma 1 when d is H, the Hellinger's distance, with M densities $f_1,\dots,f_M$ in $\left\{f_\delta : \delta \in N_{D/8}\right\}$ where $f_1 = \mathbb{1}_{[0,1]^d}$ and the increasing function $w(u) = u^q$, we get the result.

■

Remark 1 *The construction of the family of densities $\left\{f_\delta : \delta \in N_{D/8}\right\}$ is in the same spirit as the lower bound of Tsybakov (2003), Rigollet and Tsybakov (2004). But, as compared to Rigollet and Tsybakov (2004), we consider a different problem (model selection aggregation) and as compared to Tsybakov (2003), we study in a different context (density estimation). Also, our risk function is different from those considered in these papers.*

Now, we give a lower bound for KL divergence. We have the same result as for square of Hellinger's distance.

Theorem 3 *Let $M \geq 2$ be an integer, $A > 1$ and $q > 0$. We have, for any integer n such that $\log M \leq 16(\min(1,A-1))^2 n$,*

$$\sup_{f_1,\dots,f_M \in \mathcal{F}_K(A)} \inf_{\hat{f}_n} \sup_{f\in\mathcal{F}(A)} \left[\mathbb{E}_f\left[(K(f|\hat{f}_n))^q\right] - \min_{j=1,\dots,M}(K(f|f_j))^q\right] \geq c\left(\frac{\log M}{n}\right)^q, \qquad (6)$$

and

$$\sup_{f_1,\dots,f_M\in\mathcal{F}_K(A)}\inf_{\hat{f}_n}\sup_{f\in\mathcal{F}(A)}\left[\mathbb{E}_f\left[(K(\hat{f}_n|f))^q\right]-\min_{j=1,\dots,M}(K(f_j|f))^q\right]\geq c\left(\frac{\log M}{n}\right)^q, \tag{7}$$

where c is a positive constant which depends only on A. The sets $\mathcal{F}(A)$ and $\mathcal{F}_K(A)$ are defined in (5) for $\mathcal{X}=\mathbb{R}^d$.

Proof : Proof of the inequality (7) of Theorem 3 is similar to the one for (6). Since we have for all densities f and g,

$$K(f|g)\geq H^2(f,g),$$

(a proof is given in Tsybakov, 2004, p. 73), it suffices to note that, if $f_1,\dots,f_M$ are densities bounded by A then,

$$\sup_{f_1,\dots,f_M\in\mathcal{F}_K(A)}\inf_{\hat{f}_n}\sup_{f\in\mathcal{F}(A)}\left[\mathbb{E}_f\left[(K(f|\hat{f}_n))^q\right]-\min_{j=1,\dots,M}(K(f|f_i))^q\right]$$
$$\geq\inf_{\hat{f}_n}\sup_{f\in\{f_1,\dots,f_M\}}\left[\mathbb{E}_f\left[(K(f|\hat{f}_n))^q\right]\right]\geq\inf_{\hat{f}_n}\sup_{f\in\{f_1,\dots,f_M\}}\left[\mathbb{E}_f\left[H^{2q}(f,\hat{f}_n)\right]\right],$$

to get the result by applying Theorem 2.

■

With the same method as Theorem 1, we get the result below for the L_1-distance.

Theorem 4 *Let $M\geq 2$ be an integer, $A>1$ and $q>0$. We have for any integers n such that $\log M\leq 16(\min(1,A-1))^2 n$,*

$$\sup_{f_1,\dots,f_M\in\mathcal{F}_v(A)}\inf_{\hat{f}_n}\sup_{f\in\mathcal{F}(A)}\left[\mathbb{E}_f\left[v(f,\hat{f}_n)^q\right]-\min_{j=1,\dots,M}v(f,f_i)^q\right]\geq c\left(\frac{\log M}{n}\right)^{q/2}$$

where c is a positive constant which depends only on A. The sets $\mathcal{F}(A)$ and $\mathcal{F}_v(A)$ are defined in (5) for $\mathcal{X}=\mathbb{R}^d$.

Proof : The only difference with Theorem 2 is in the control of the distances. With the same notations as the proof of Theorem 2, we have,

$$v(f_{\delta^1},f_{\delta^2})=\int_{[0,1]^d}|f_{\delta^1}(x)-f_{\delta^2}(x)|dx=\rho(\delta^1,\delta^2)\int_0^{1/D}|h(x)|dx=\frac{L}{D^2}\rho(\delta^1,\delta^2),$$

for all $\delta^1,\delta^2\in\Delta$. Thus, for $L=(D/4)\sqrt{\log(M)/n}$ and $N_{D/8}$, the $D/8$-separated set of Δ introduced in the proof of Theorem 2, we have,

$$v(f_{\delta^1},f_{\delta^2})\geq\frac{1}{32}\sqrt{\frac{\log(M)}{n}},\quad\forall\delta^1,\delta^2\in N_{D/8}\text{ and }K(P_\delta^{\otimes n}|P_0^{\otimes n})\leq\frac{1}{16}\log(M),\quad\forall\delta\in\Delta.$$

Therefore, by applying Lemma 1 to the L_1-distance with M densities $f_1,\dots,f_M$ in $\left\{f_\delta:\delta\in N_{D/8}\right\}$ where $f_1=\mathbb{1}_{[0,1]^d}$ and the increasing function $w(u)=u^q$, we get the result.

■

4. Upper Bounds

In this section we use an argument in Yang (2000a) (see also Catoni, 2004) to show that the rate of the lower bound of Theorem 3 is an optimal rate of aggregation with respect to the KL loss. We use an aggregate constructed by Yang (defined in (1)) to attain this rate. An upper bound of the type (3) is stated in the following Theorem. Remark that Theorem 5 holds in a general framework of a measurable space $(\mathcal{X},\mathcal{A})$ endowed with a σ-finite measure ν.

Theorem 5 (Yang) *Let $X_1,\dots,X_n$ be n observations of a probability measure on $(\mathcal{X},\mathcal{A})$ of density f with respect to ν. Let $f_1,\dots,f_M$ be M densities on $(\mathcal{X},\mathcal{A},\nu)$. The aggregate $\tilde{f}_n$, introduced in (1), satisfies, for any underlying density f,*

$$\mathbb{E}_f\left[K(f|\tilde{f}_n)\right] \leq \min_{j=1,\dots,M} K(f|f_j) + \frac{\log(M)}{n+1}. \tag{8}$$

Proof : Proof follows the line of Yang (2000a), although he does not state the result in the form (3), for convenience we reproduce the argument here. We define $\hat{f}_k(x;X^{(k)}) = \sum_{j=1}^M w_j^{(k)} f_j(x)$, $\forall k = 1,\dots,n$ (where $w_j^{(k)}$ is defined in (2) and $x^{(k)} = (x_1,\dots,x_k)$ for all $k\in\mathbb{N}$ and $x_1,\dots,x_k\in\mathcal{X}$) and $\hat{f}_0(x;X^{(0)}) = (1/M)\sum_{j=1}^M f_j(x)$ for all $x\in\mathcal{X}$. Thus, we have

$$\tilde{f}_n(x;X^{(n)}) = \frac{1}{n+1}\sum_{k=0}^n \hat{f}_k(x;X^{(k)}).$$

Let f be a density on $(\mathcal{X},\mathcal{A},\nu)$. We have

$$\begin{aligned}
\sum_{k=0}^n \mathbb{E}_f\left[K(f|\hat{f}_k)\right] &= \sum_{k=0}^n \int_{\mathcal{X}^{k+1}} \log\left(\frac{f(x_{k+1})}{\hat{f}_k(x_{k+1};x^{(k)})}\right)\prod_{i=1}^{k+1} f(x_i)d\nu^{\otimes(k+1)}(x_1,\dots,x_{k+1}) \\
&= \int_{\mathcal{X}^{n+1}} \left(\sum_{k=0}^n \log\left(\frac{f(x_{k+1})}{\hat{f}_k(x_{k+1};x^{(k)})}\right)\right)\prod_{i=1}^{n+1} f(x_i)d\nu^{\otimes(n+1)}(x_1,\dots,x_{n+1}) \\
&= \int_{\mathcal{X}^{n+1}} \log\left(\frac{f(x_1)\dots f(x_{n+1})}{\prod_{k=0}^n \hat{f}_k(x_{k+1};x^{(k)})}\right)\prod_{i=1}^{n+1} f(x_i)d\nu^{\otimes(n+1)}(x_1,\dots,x_{n+1}),
\end{aligned}$$

but $\prod_{k=0}^n \hat{f}_k(x_{k+1};x^{(k)}) = (1/M)\sum_{j=1}^M f_j(x_1)\dots f_j(x_{n+1}), \forall x_1,\dots,x_{n+1}\in\mathcal{X}$ thus,

$$\sum_{k=0}^n \mathbb{E}_f\left[K(f|\hat{f}_k)\right] = \int_{\mathcal{X}^{n+1}} \log\left(\frac{f(x_1)\dots f(x_{n+1})}{\frac{1}{M}\sum_{j=1}^M f_j(x_1)\dots f_j(x_{n+1})}\right)\prod_{i=1}^{n+1} f(x_i)d\nu^{\otimes(n+1)}(x_1,\dots,x_{n+1}),$$

moreover $x\longmapsto \log(1/x)$ is a decreasing function so,

$$\begin{aligned}
&\sum_{k=0}^n \mathbb{E}_f\left[K(f|\hat{f}_k)\right] \\
&\leq \min_{j=1,\dots,M}\left\{\int_{\mathcal{X}^{n+1}} \log\left(\frac{f(x_1)\dots f(x_{n+1})}{\frac{1}{M}f_j(x_1)\dots f_j(x_{n+1})}\right)\prod_{i=1}^{n+1} f(x_i)d\nu^{\otimes(n+1)}(x_1,\dots,x_{n+1})\right\} \\
&\leq \log M + \min_{j=1,\dots,M}\left\{\int_{\mathcal{X}^{n+1}} \log\left(\frac{f(x_1)\dots f(x_{n+1})}{f_j(x_1)\dots f_j(x_{n+1})}\right)\prod_{i=1}^{n+1} f(x_i)d\nu^{\otimes(n+1)}(x_1,\dots,x_{n+1})\right\},
\end{aligned}$$

finally we have,

$$\sum_{k=0}^{n} \mathbb{E}_f\left[K(f|\hat{f}_k)\right] \leq \log M + (n+1) \inf_{j=1,\ldots,M} K(f|f_j). \tag{9}$$

On the other hand we have,

$$\mathbb{E}_f\left[K(f|\tilde{f}_n)\right] = \int_{\mathcal{X}^{n+1}} \log\left(\frac{f(x_{n+1})}{\frac{1}{n+1}\sum_{k=0}^{n}\hat{f}_k(x_{n+1};x^{(k)})}\right) \prod_{i=1}^{n+1} f(x_i) d\nu^{\otimes(n+1)}(x_1,\ldots,x_{n+1}),$$

and $x \longmapsto \log(1/x)$ is convex, thus,

$$\mathbb{E}_f\left[K(f|\tilde{f}_n)\right] \leq \frac{1}{n+1}\sum_{k=0}^{n} \mathbb{E}_f\left[K(f|\hat{f}_k)\right]. \tag{10}$$

Theorem 5 follows by combining (9) and (10).

■

Birgé constructs estimators, called *T-estimators* (the "T" is for "test"), which are adaptive in aggregation selection model of M estimators with a residual proportional at $(\log M/n)^{q/2}$ when Hellinger and L_1-distances are used to evaluate the quality of estimation (cf. Birgé (2004)). But it does not give an optimal result as Yang, because there is a constant greater than 1 in front of the main term $\min_{i=1,\ldots,M} d^q(f,f_i)$ where d is the Hellinger distance or the L_1 distance. Nevertheless, observing the proof of Theorem 2 and 4, we can obtain

$$\sup_{f_1,\ldots,f_M\in\mathcal{F}(A)} \inf_{\hat{f}_n} \sup_{f\in\mathcal{F}(A)} \left[\mathbb{E}_f\left[d(f,\hat{f}_n)^q\right] - C(q) \min_{i=1,\ldots,M} d(f,f_i)^q\right] \geq c\left(\frac{\log M}{n}\right)^{q/2},$$

where d is the Hellinger or L_1-distance, $q>0$ and $A>1$. The constant $C(q)$ can be chosen equal to the one appearing in the following Theorem. The same residual appears in this lower bound and in the upper bounds of Theorem 6, so we can say that

$$\left(\frac{\log M}{n}\right)^{q/2}$$

is near optimal rate of aggregation w.r.t. the Hellinger distance or the L_1-distance to the power q, in the sense given at the end of Section 2. We recall Birgé's results in the following Theorem.

Theorem 6 (Birgé) *If we have n observations of a probability measure of density f w.r.t. ν and $f_1,\ldots,f_M$ densities on $(\mathcal{X},\mathcal{A},\nu)$, then there exists an estimator $\tilde{f}_n$ (T-estimator) such that for any underlying density f and $q>0$, we have*

$$\mathbb{E}_f\left[H(f,\tilde{f}_n)^q\right] \leq C(q)\left(\min_{j=1,\ldots,M} H(f,f_j)^q + \left(\frac{\log M}{n}\right)^{q/2}\right),$$

and for the L_1-distance we can construct an estimator $\tilde{f}_n$ which satisfies :

$$\mathbb{E}_f\left[v(f,\tilde{f}_n)^q\right] \leq C(q)\left(\min_{j=1,\ldots,M} v(f,f_j)^q + \left(\frac{\log M}{n}\right)^{q/2}\right),$$

where $C(q)>0$ is a constant depending only on q.

Another result, which can be found in Devroye and Lugosi (2001), states that the minimum distance estimate proposed by Yatracos (1985) (cf. Devroye and Lugosi (2001, p. 59)) achieves the same aggregation rate as in Theorem 6 for the L_1-distance with $q=1$. Namely, for all $f, f_1, \dots, f_M \in \mathcal{F}(A)$,

$$\mathbb{E}_f\left[v(f,\check{f}_n)\right] \leq 3 \min_{j=1,\dots,M} v(f,f_j) + \sqrt{\frac{\log M}{n}},$$

where $\check{f}_n$ is the estimator of Yatracos defined by

$$\check{f}_n = \arg \min_{f\in\{f_1,\dots,f_M\}} \sup_{A\in\mathcal{A}} \left| \int_A f - \frac{1}{n}\sum_{i=1}^n \mathbb{1}_{\{X_i\in A\}} \right|,$$

and $\mathcal{A} = \left\{ \{x : f_i(x) > f_j(x)\} : 1 \leq i,j \leq M \right\}$.

References

N. H. Augustin, S. T. Buckland, and K. P. Burnham. Model selection: An integral part of inference. *Biometrics*, 53:603–618, 1997.

A. Barron and G. Leung. Information theory and mixing least-square regressions. 2004. manuscript.

L. Birgé. Model selection via testing: an alternative to (penalized) maximum likelihood estimators. *To appear in Annales of IHP*, 2004. Available at http://www.proba.jussieu.fr/mathdoc/textes/PMA-862.pdf.

F. Bunea and A. Nobel. Online prediction algorithms for aggregation of arbitrary estimators of a conditional mean. 2005. Submitted to IEEE Transactions in Information Theory.

O. Catoni. A mixture approach to universal model selection. 1997. preprint LMENS-97-30, available at http://www.dma.ens.fr/EDITION/preprints/.

O. Catoni. *Statistical Learning Theory and Stochastic Optimization*. Ecole d'été de Probabilités de Saint-Flour 2001, Lecture Notes in Mathematics. Springer, N.Y., 2004.

L. Devroye and G. Lugosi. *Combinatorial methods in density estimation*. 2001. Springer, New-York.

J.A. Hartigan. Bayesian regression using akaike priors. 2002. Yale University, New Haven, Preprint.

I.A. Ibragimov and R.Z. Hasminskii. An estimate of density of a distribution. Studies in mathematical stat. IV. Zap. Nauchn. Semin., LOMI, 98(1980),61–85.

A. Juditsky, A. Nazin, A.B. Tsybakov and N. Vayatis. Online aggregation with mirror-descent algorithm. 2005. Preprint n.987, Laboratoire de Probabilités et Modèle aléatoires, Universités Paris 6 and Paris 7 (available at http://www.proba.jussieu.fr/mathdoc/preprints/index.html#2005).

A. Juditsky and A. Nemirovski. Functionnal aggregation for nonparametric estimation. *Ann. of Statist.*, 28:681–712, 2000.

G. Lecué. Simultaneous adaptation to the marge and to complexity in classification. 2005. Available at http://hal.ccsd.cnrs.fr/ccsd-00009241/en/.

A. Nemirovski. *Topics in Non-parametric Statistics*. Springer, N.Y., 2000.

P. Rigollet and A. B. Tsybakov. Linear and convex aggregation of density estimators. 2004. Manuscript.

A.B. Tsybakov. Optimal rates of aggregation. *Computational Learning Theory and Kernel Machines. B.Schölkopf and M.Warmuth, eds. Lecture Notes in Artificial Intelligence*, 2777:303–313, 2003. Springer, Heidelberg.

A.B. Tsybakov. *Introduction à l'estimation non-paramétrique*. Springer, 2004.

Y. Yang. Mixing strategies for density estimation. *Ann. Statist.*, 28(1):75–87, 2000a.

Y. Yang. Combining Different Procedures for Adaptive Regression. *Journal of Multivariate Analysis*, 74:135–161, 2000b.

Y. Yang. Adaptive regression by mixing. *Journal of American Statistical Association*, 96:574–588, 2001.

Y. Yang. Aggregating regression procedures to impose performance. *Bernoulli*, 10(1):25–47, 2004.

T. Zhang. From epsilon-entropy to KL-complexity: analysis of minimum information complexity density estimation. 2003. Tech. Report RC22980, IBM T.J.Watson Research Center.

Journal of Machine Learning Research 7 (2006) 983–999 Submitted 10/05; Revised 2/06; Published 6/06

Quantile Regression Forests

Nicolai Meinshausen NICOLAI@STAT.MATH.ETHZ.CH
Seminar für Statistik
ETH Zürich
8092 Zürich, Switzerland

Editor: Greg Ridgeway

Abstract

Random forests were introduced as a machine learning tool in Breiman (2001) and have since proven to be very popular and powerful for high-dimensional regression and classification. For regression, random forests give an accurate approximation of the conditional mean of a response variable. It is shown here that random forests provide information about the full conditional distribution of the response variable, not only about the conditional mean. Conditional quantiles can be inferred with quantile regression forests, a generalisation of random forests. Quantile regression forests give a non-parametric and accurate way of estimating conditional quantiles for high-dimensional predictor variables. The algorithm is shown to be consistent. Numerical examples suggest that the algorithm is competitive in terms of predictive power.

Keywords: quantile regression, random forests, adaptive neighborhood regression

1. Introduction

Let Y be a real-valued response variable and X a covariate or predictor variable, possibly high-dimensional. A standard goal of statistical analysis is to infer, in some way, the relationship between Y and X. Standard regression analysis tries to come up with an estimate $\hat{\mu}(x)$ of the conditional mean $E(Y|X = x)$ of the response variable Y, given $X = x$. The conditional mean minimizes the expected squared error loss,

$$E(Y|X = x) = \arg\min_z E\{(Y - z)^2|X = x\},$$

and approximation of the conditional mean is typically achieved by minimization of a squared error type loss function over the available data.

Beyond the Conditional Mean The conditional mean illuminates just one aspect of the conditional distribution of a response variable Y, yet neglects all other features of possible interest. This led to the development of quantile regression; for a good summary see e.g. Koenker (2005). The conditional distribution function $F(y|X = x)$ is given by the probability that, for $X = x$, Y is smaller than $y \in \mathbb{R}$,

$$F(y|X = x) = P(Y \leq y|X = x).$$

For a continuous distribution function, the α-quantile $Q_\alpha(x)$ is then defined such that the probability of Y being smaller than $Q_\alpha(x)$ is, for a given $X = x$, exactly equal to α. In

general,

$$Q_\alpha(x) = \inf\{y : F(y|X = x) \geq \alpha\}. \quad (1)$$

The quantiles give more complete information about the distribution of Y as a function of the predictor variable X than the conditional mean alone.

As an example, consider the predictions of next day ozone levels, as in Breiman and Friedman (1985). Least-squares regression tries to estimate the conditional mean of ozone levels. It gives little information about the fluctuations of ozone levels around this predicted conditional mean. It might for example be of interest to find an ozone level that is -with high probability- not surpassed. This can be achieved with quantile regression, as it gives information about the spread of the response variable. For some other examples see Le et al. (2005), which is to the best of our knowledge the first time that quantile regression is mentioned in the Machine Learning literature.

Prediction Intervals How reliable is a prediction for a new instance? This is a related question of interest. Consider again the prediction of next day ozone levels. Some days, it might be possible to pinpoint next day ozone levels to a higher accuracy than on other days (this can indeed be observed for the ozone data, see the section with numerical results). With standard prediction, a single point estimate is returned for each new instance. This point estimate does not contain information about the dispersion of observations around the predicted value.

Quantile regression can be used to build prediction intervals. A 95% prediction interval for the value of Y is given by

$$I(x) = [Q_{.025}(x), Q_{.975}(x)]. \quad (2)$$

That is, a new observation of Y, for $X = x$, is with high probability in the interval $I(x)$. The width of this prediction interval can vary greatly with x. Indeed, going back to the previous example, next day ozone level can on some days be predicted five times more accurately than on other days. This effect is even more pronounced for other data sets. Quantile regression offers thus a principled way of judging the reliability of predictions.

Outlier Detection Quantile regression can likewise be used for outlier detection (for surveys on outlier detection see e.g. Barnett and Lewis, 1994; Hodge and Austin, 2004). A new observation (X, Y) would be regarded as an outlier if its observed value Y is extreme, in some sense, with regard to the predicted conditional distribution function.

There is, however, no generally applicable rule of what precisely constitutes an "extreme" observation. One could possibly flag observations as outliers if the distance between Y and the median of the conditional distribution is large; "large" being measured in comparison to some robust measure of dispersion like the conditional median absolute deviation or the conditional interquartile range (Huber, 1973). Both quantities are made available by quantile regression.

Note that only anomalies in the conditional distribution of Y can be detected in this way. Outliers of X itself cannot be detected. Other research has focused on detecting anomalies for unlabelled data (e.g. Markou and Singh, 2003; Steinwart et al., 2005).

Estimating Quantiles from Data Quantile regression aims to estimate the conditional quantiles from data. Quantile regression can be cast as an optimization problem, just as estimation of the conditional mean is achieved by minimizing a squared error loss function. Let the loss function L_α be defined for $0 < \alpha < 1$ by the weighted absolute deviations

$$L_\alpha(y, q) = \begin{cases} \alpha \, |y - q| & y > q \\ (1 - \alpha) \, |y - q| & y \leq q \end{cases} . \tag{3}$$

While the conditional mean minimizes the expected squared error loss, conditional quantiles minimize the expected loss $E(L_\alpha)$,

$$Q_\alpha(x) = \arg\min_q \; E\{L_\alpha(Y, q)|X = x\}.$$

A parametric quantile regression is solved by optimizing the parameters so that the empirical loss is minimal. This can be achieved efficiently due to the convex nature of the optimization problem (Portnoy and Koenker, 1997). Non-parametric approaches, in particular quantile Smoothing Splines (He et al., 1998; Koenker et al., 1994), involve similar ideas. Chaudhuri and Loh (2002) developed an interesting tree-based method for estimation of conditional quantiles which gives good performance and allows for easy interpretation, being in this respect similar to CART (Breiman et al., 1984).

In this manuscript, a different approach is proposed, which does not directly employ minimization of a loss function of the sort (3). Rather, the method is based on random forests (Breiman, 2001). Random forests grows an ensemble of trees, employing random node and split point selection, inspired by Amit and Geman (1997). The prediction of random forests can then be seen as an adaptive neighborhood classification and regression procedure (Lin and Jeon, 2002). For every $X = x$, a set of weights $w_i(x)$, $i = 1, \ldots, n$ for the original n observations is obtained. The prediction of random forests, or estimation of the conditional *mean*, is equivalent to the *weighted mean of the observed response variables*. For quantile regression forests, trees are grown as in the standard random forests algorithm. The conditional *distribution* is then estimated by the *weighted distribution of observed response variables*, where the weights attached to observations are identical to the original random forests algorithm.

In Section 2, necessary notation is introduced and the mechanism of random forests is briefly explained, using the interpretation of Lin and Jeon (2002), which views random forests as an adaptive nearest neighbor algorithm, a view that is later supported in Breiman (2004). Using this interpretation, quantile regression forests are introduced in Section 3 as a natural generalisation of random forests. A proof of consistency is given in Section 4, while encouraging numerical results for popular machine learning data sets are presented in Section 5.

2. Random Forests

Random forests grows an ensemble of trees, using n independent observations

$$(Y_i, X_i), \quad i = 1, \ldots, n.$$

A large number of trees is grown. For each tree and each node, random forests employs randomness when selecting a variable to split on. For each tree, a bagged version of the

training data is used. In addition, only a random subset of predictor variables is considered for splitpoint selection at each node. The size of the random subset, called *mtry*, is the single tuning parameter of the algorithm, even though results are typically nearly optimal over a wide range of this parameter. The value of *mtry* can be fine-tuned on the out-of-bag samples. For regression, the prediction of random forests for a new data point $X = x$ is the averaged response of all trees. For details see Breiman (2001). The algorithm is somewhat related to boosting (Schapire et al., 1998), with trees as learners. Yet, with random forests, each tree is grown using the original observations of the response variable, while boosting tries to fit the residuals after taking into account the prediction of previously generated trees (Friedman et al., 2000).

Some Notation Following the notation of Breiman (2001), call θ the random parameter vector that determines how a tree is grown (e.g. which variables are considered for splitpoints at each node). The corresponding tree is denoted by $T(\theta)$. Let $\mathcal{B}$ be the space in which X lives, that is $X : \Omega \mapsto \mathcal{B} \subseteq \mathbb{R}^p$, where $p \in \mathbb{N}_+$ is the dimensionality of the predictor variable. Every leaf $\ell = 1, \ldots, L$ of a tree corresponds to a rectangular subspace of $\mathcal{B}$. Denote this rectangular subspace by $R_\ell \subseteq \mathcal{B}$ for every leaf $\ell = 1, \ldots, L$. For every $x \in \mathcal{B}$, there is one and only one leaf ℓ such that $x \in R_\ell$ (corresponding to the leaf that is obtained when dropping x down the tree). Denote this leaf by $\ell(x, \theta)$ for tree $T(\theta)$.

The prediction of a single tree $T(\theta)$ for a new data point $X = x$ is obtained by averaging over the observed values in leaf $\ell(x, \theta)$. Let the weight vector $w_i(x, \theta)$ be given by a positive constant if observation X_i is part of leaf $\ell(x, \theta)$ and 0 if it is not. The weights sum to one, and thus

$$w_i(x, \theta) = \frac{1_{\{X_i \in R_{\ell(x,\theta)}\}}}{\#\{j : X_j \in R_{\ell(x,\theta)}\}}. \tag{4}$$

The prediction of a single tree, given covariate $X = x$, is then the weighted average of the original observations $Y_i, i = 1, \ldots, n$,

$$\text{single tree:} \quad \hat{\mu}(x) = \sum_{i=1}^{n} w_i(x, \theta)\, Y_i.$$

Using random forests, the conditional mean $E(Y|X = x)$ is approximated by the averaged prediction of k single trees, each constructed with an i.i.d. vector θ_t, $t = 1, \ldots, k$. Let $w_i(x)$ be the average of $w_i(\theta)$ over this collection of trees,

$$w_i(x) = k^{-1} \sum_{t=1}^{k} w_i(x, \theta_t). \tag{5}$$

The prediction of random forests is then

$$\text{Random Forests:} \quad \hat{\mu}(x) = \sum_{i=1}^{n} w_i(x) Y_i.$$

The approximation of the conditional mean of Y, given $X = x$, is thus given by a weighted sum over all observations. The weights vary with the covariate $X = x$ and tend to be large for those $i \in \{1, \ldots, n\}$ where the conditional distribution of Y, given $X = X_i$, is similar to the conditional distribution of Y, given $X = x$ (Lin and Jeon, 2002).

3. Quantile Regression Forests

It was shown above that random forests approximates the conditional mean $E(Y|X = x)$ by a weighted mean over the observations of the response variable Y. One could suspect that the weighted observations deliver not only a good approximation to the conditional mean but to the full conditional distribution. The conditional distribution function of Y, given $X = x$, is given by

$$F(y|X = x) = P(Y \leq y|X = x) = E(1_{\{Y \leq y\}}|X = x).$$

The last expression is suited to draw analogies with the random forest approximation of the conditional mean $E(Y|X = x)$. Just as $E(Y|X = x)$ is approximated by a weighted mean over the observations of Y, define an approximation to $E(1_{\{Y \leq y\}}|X = x)$ by the weighted mean over the observations of $1_{\{Y \leq y\}}$,

$$\hat{F}(y|X = x) = \sum_{i=1}^{n} w_i(x)\ 1_{\{Y_i \leq y\}}, \tag{6}$$

using the same weights $w_i(x)$ as for random forests, defined in equation (5). This approximation is at the heart of the quantile regression forests algorithm.

The Algorithm The algorithm for computing the estimate $\hat{F}(y|X = x)$ can be summarized as:

a) Grow k trees $T(\theta_t)$, $t = 1, \ldots, k$, as in random forests. However, for every leaf of every tree, take note of all observations in this leaf, not just their average.

b) For a given $X = x$, drop x down all trees. Compute the weight $w_i(x, \theta_t)$ of observation $i \in \{1, \ldots, n\}$ for every tree as in (4). Compute weight $w_i(x)$ for every observation $i \in \{1, \ldots, n\}$ as an average over $w_i(x, \theta_t)$, $t = 1, \ldots, k$, as in (5).

c) Compute the estimate of the distribution function as in (6) for all $y \in \mathbb{R}$, using the weights from Step b).

Estimates $\hat{Q}_\alpha(x)$ of the conditional quantiles $Q_\alpha(x)$ are obtained by plugging $\hat{F}(y|X = x)$ instead of $F(y|X = x)$ into (1). Other approaches for estimating quantiles from empirical distribution functions are discussed in Hyndman and Fan (1996).

The key difference between quantile regression forests and random forests is as follows: for each node in each tree, random forests keeps only the mean of the observations that fall into this node and neglects all other information. In contrast, quantile regression forests keeps the value of all observations in this node, not just their mean, and assesses the conditional distribution based on this information.

Software is made available as a package `quantregForest` for R (R Development Core Team, 2005). The package builds upon the excellent R-package `randomForest` (Liaw and Wiener, 2002).

4. Consistency

Consistency of the proposed method is shown. Consistency for random forests (when approximating the conditional mean) has been shown for a simplified model of random forests in (Breiman, 2004), together with an analysis of convergence rates. The conditions for growing individual trees are less stringent in the current analysis but no attempt is made to analyze convergence rates.

Assumptions Three assumptions are needed for the proof of consistency. First, an assumption is made about the distribution of covariates.

Assumption 1 *$\mathcal{B} = [0,1]^p$ and X uniform on $[0,1]^p$.*

This assumption is just made for notational convenience. Alternatively, one could assume that the density of X is positive and bounded from above and below by positive constants.

Next, two assumptions are made about the construction of individual trees. Denote the node-sizes of the leaves ℓ of a tree constructed with parameter vector θ by $k_\theta(\ell)$, that is $k_\theta(\ell) = \#\{i \in \{1,\ldots,n\} : X_i \in R_{\ell(x,\theta)}\}$.

Assumption 2 *The proportion of observations in a node, relative to all observations, is vanishing for large n, $\max_{\ell,\theta} k_\theta(\ell) = o(n)$, for $n \to \infty$. The minimal number of observations in a node is growing for large n, that is $1/\min_{\ell,\theta} k_\theta(\ell) = o(1)$, for $n \to \infty$.*

The first part of this assumption is necessary. The second part could possibly be dropped, with a more involved proof of consistency.

The following assumption concerns the actual construction of trees. An attempt has been made to keep these assumptions as minimal as possible.

Assumption 3 *When finding a variable for a splitpoint, the probability that variable $m = 1,\ldots,p$ is chosen for the splitpoint is bounded from below for every node by a positive constant. If a node is split, the split is chosen so that each of the resulting sub-nodes contains at least a proportion γ of the observations in the original node, for some $0 < \gamma \leq 0.5$.*

Next, The conditional distribution function is assumed to be Lipschitz continuous.

Assumption 4 *There exists a constant L so that $F(y|X = x)$ is Lipschitz continuous with parameter L, that is for all $x, x' \in \mathcal{B}$,*

$$\sup_y |F(y|X = x) - F(y|X = x')| \;\leq\; L\|x - x'\|_1.$$

Lastly, positive density is assumed.

Assumption 5 *The conditional distribution function $F(y|X = x)$ is, for every $x \in \mathcal{B}$, strictly monotonously increasing in y.*

This assumption is necessary to derive consistency of quantile estimates from the consistency of distribution estimates.

Consistency Under the made assumptions, consistency of quantile regression forests is shown.

Theorem 1 *Let Assumptions 1-5 be fulfilled. It holds pointwise for every $x \in \mathcal{B}$ that*

$$\sup_{y \in \mathbb{R}} |\hat{F}(y|X = x) - F(y|X = x)| \rightarrow_p 0 \qquad n \rightarrow \infty.$$

In other words, the error of the approximation to the conditional distribution converges uniformly in probability to zero for $n \rightarrow \infty$. Quantile regression forests is thus a consistent way of estimating conditional distributions and quantile functions.

Proof Let the random variables U_i, $i = 1, \ldots, n$ be defined as the quantiles of the observations Y_i, conditional on $X = X_i$,

$$U_i = F(Y_i|X = X_i).$$

Note that U_i, $i = 1, \ldots, n$ are i.i.d. uniform on $[0, 1]$. For a given $X = X_i$, the event $\{Y_i \leq y\}$ is identical to $\{U_i \leq F(y|X = X_i)\}$ under Assumption 5. The approximation $\hat{F}(y|x) = \hat{F}(y|X = x)$ can then be written as a sum of two parts,

$$\begin{aligned} \hat{F}(y|x) &= \sum_{i=1}^{n} w_i(x)\ 1_{\{Y_i \leq y\}} = \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \leq F(y|X_i)\}} \\ &= \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \leq F(y|x)\}} + \\ &\quad \sum_{i=1}^{n} w_i(x)\ (1_{\{U_i \leq F(y|X_i)\}} - 1_{\{U_i \leq F(y|x)\}}). \end{aligned}$$

The absolute difference between the approximation and the true value is hence bounded by

$$\begin{aligned} |F(y|x) - \hat{F}(y|x)| &\leq |F(y|x) - \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \leq F(y|x)\}}| + \\ &\quad |\sum_{i=1}^{n} w_i(x)(1_{\{U_i \leq F(y|X_i)\}} - 1_{\{U_i \leq F(y|x)\}})|. \end{aligned}$$

The first term is a variance-type part, while the second term reflects the change in the underlying distribution (if the distribution would be constant as a function of x, the second term would vanish). Taking supremum over y in the first part leads to

$$\sup_{y \in \mathbb{R}} |F(y|x) - \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \leq F(y|x)\}}| = \sup_{z \in [0,1]} |z - \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \leq z\}}|.$$

Note that $E(1_{\{U_i \leq z\}}) = z$, as U_i are i.i.d. uniform random variables on $[0, 1]$. Furthermore $0 \leq w_i(x) \leq (\min_{\ell,\theta} k_\theta(\ell))^{-1}$. As the weights add to one, $\sum_{i=1}^{n} w_i(x) = 1$, and $(\min_{\ell,\theta} k_\theta(\ell))^{-1} = o(1)$ by Assumption 2, it follows that, for every $x \in \mathcal{B}$,

$$\sum_{i=1}^{n} w_i(x)^2 \rightarrow 0 \qquad n \rightarrow \infty. \tag{7}$$

and hence, for every $z \in [0,1]$ and $x \in \mathcal{B}$,

$$|z - \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \le z\}}| = o_p(1) \qquad n \to \infty.$$

By Bonferroni's inequality, the above still holds true if, on the left hand side, the supremum over a finite set of z-values is taken, where the cardinality of this set can grow to infinity for $n \to \infty$. By straightforward yet tedious calculations, it can be shown that the supremum can be extended not only to such a set of z-values, but also to the whole interval $z \in [0,1]$, so that

$$\sup_{z \in [0,1]} |z - \sum_{i=1}^{n} w_i(x)\ 1_{\{U_i \le z\}}| = o_p(1) \qquad n \to \infty.$$

It thus remains to be shown that, for every $x \in \mathcal{B}$,

$$|\sum_{i=1}^{n} w_i(x)(1_{\{U_i \le F(y|X_i)\}} - 1_{\{U_i \le F(y|x)\}})| \to_p 0 \qquad n \to \infty.$$

As U_i, $i = 1, \ldots, n$ are uniform over $[0,1]$, it holds that

$$E(1_{\{U_i \le F(y|X_i)\}} - 1_{\{U_i \le F(y|x)\}}) = F(y|X_i) - F(y|x).$$

Using (7) and independence of all U_i, $i = 1, \ldots, n$, for $n \to \infty$,

$$|\sum_{i=1}^{n} w_i(x)(1_{\{U_i \le F(y|X_i)\}} - 1_{\{U_i \le F(y|x)\}})| \to_p \sum_{i=1}^{n} w_i(x)\{F(y|X_i) - F(y|x)\}.$$

Using Assumption 4 about Lipschitz continuity of the distribution function, it thus remains to show that

$$\sum_{i=1}^{n} w_i(x)\ \|x - X_i\|_1 = o_p(1) \qquad n \to \infty.$$

Note that $w_i(x) = k^{-1} \sum_{t=1,\ldots,k} w_i(x, \theta_t)$, where $w_i(x, \theta)$ is defined as the weight produced by a single tree with (random) parameter θ_t, as in (4). Thus it suffices to show that, for a single tree,

$$\sum_{i=1}^{n} w_i(x, \theta)\ \|x - X_i\|_1 = o_p(1) \qquad n \to \infty. \tag{8}$$

The rectangular subspace $R_{\ell(x,\theta)} \subseteq [0,1]^p$ of leaf $\ell(x, \theta)$ of tree $T(\theta)$ is defined by the intervals $I(x, m, \theta) \subseteq [0,1]$ for $m = 1, \ldots, p$,

$$R_{\ell(x,\theta)} = \otimes_{m=1}^{p} I(x, m, \theta).$$

Note that $X_i \notin I(x, m, \theta)$ implies $w_i(x, \theta) = 0$ by (4). To show (8), it thus suffices to show that $\max_m |I(x, m, \theta)| = o_p(1)$ for $n \to \infty$, for all $x \in \mathcal{B}$. The proof is thus complete with Lemma 2. ■

Lemma 2 *Under the conditions of Theorem 1, it holds for all* $x \in \mathcal{B}$ *that* $\max_m |I(x, m, \theta)| = o_p(1)$ *for* $n \to \infty$.

Proof As any $x \in \mathcal{B}$ is dropped down a tree, several nodes are passed. Denote by $S(x, m, \theta)$ the number of times that these nodes contain a splitpoint on variable m; this is a function of the random parameter θ and of x. The total number of nodes that x passes through is denoted by

$$S(x, \theta) = \sum_{m=1}^{p} S(x, m, \theta).$$

Using the second part of Assumption 3, the maximal number of observations in any leaf, $\max_\ell k_\theta(\ell)$, is bounded (for every tree θ) from below by $n\gamma^{S_{\min}(\theta)}$, where

$$S_{\min}(\theta) = \min_{x \in \mathcal{B}} S(x, \theta).$$

Using the first part of Assumption 2, the maximal number of observations in any leaf, $\max_\ell k_\theta(\ell)$, is on the other hand bounded from above by an $o(n)$-term. Putting together, one can conclude that $n\gamma^{S_{\min}(\theta)} = o(n)$ for $n \to \infty$ and thus $\gamma^{S_{\min}(\theta)} = o(1)$ for $n \to \infty$. Hence there exists a sequence s_n with $s_n \to \infty$ for $n \to \infty$, such that $S_{\min}(\theta) \geq s_n$ for all n.

As the probability of splitting on variable $m \in \{1, \ldots, p\}$ is bounded from below by a positive constant, by the first part of Assumption 3, there exists a sequence g_n with $g_n \to \infty$ for $n \to \infty$ such that, for every $x \in \mathcal{B}$,

$$P\{\min_m S(x, m, \theta) > g_n\} \to 1 \qquad n \to \infty. \tag{9}$$

Using Assumption 3, the proportion of observations whose m-th component is contained in $I(x, m, \theta)$ is bounded from above by

$$n^{-1}\#\{i \in \{1, \ldots, n\} : X_{i,m} \in I(x, m, \theta)\} \;\leq\; (1 - \gamma)^{S(x,m,\theta)}.$$

Using (9), it follows that

$$\max_m \; n^{-1}\#\{i \in \{1, \ldots, n\} : X_{i,m} \in I(x, m, \theta)\} \;=\; o_p(1). \tag{10}$$

Let $F_n^{(m)}$ be the empirical distribution of $X_{i,m}$, $i = 1, \ldots, n$,

$$F_n^{(m)}(t) \;=\; n^{-1}\#\{i \in \{1, \ldots, n\} : X_{i,m} \leq t\}.$$

As the predictor variables are assumed to be uniform over $[0, 1]$, it holds by a Kolmogorov-Smirnov type argument that

$$\sup_{t \in [0,1]} |F_n^{(m)}(t) - t| \to_p 0 \quad n \to \infty,$$

and (10) implies thus $\max_m |I(x, m, \theta)| = o_p(1)$ for $n \to \infty$, which completes the proof. ■

Note that the discussion of consistency here has several shortcomings, which need to be addressed in follow-up work. For one, no distinction is made between noise variables

and variables that contain signal, as in Breiman (2004). This treatment would require more involved assumptions about the probability that a certain variable is selected for a splitpoint, yet might explain the robustness of quantile regression forests against inclusion of many noise variables, something that has been observed empirically for random forests and, according to some numerical experience, holds as well for quantile regression forests. Second, convergence rates are not discussed. Third, it is noteworthy that Theorem 1 holds regardless of the number k of grown trees. Empirically, however, a single random tree is performing very much worse than a large ensemble of trees. The stabilizing effect of many trees is thus neglected in the current analysis, but would certainly be of relevance when discussing convergence rates.

5. Numerical Examples

Quantile regression forests (QRF) is applied to various popular data sets from the Machine Learning literature and results are compared to four other quantile regression methods: linear quantile regression with interactions (QQR) and without interactions (LQR), and quantile regression trees with with piecewise constant (TRC), piecewise multiple linear (TRM), and piecewise second-degree polynomial form (TRP).

For quantile regression forests (QRF), bagged versions of the training data are used for each of the $k = 1000$ trees. One could use the out-of-bag predictions to determine the optimal number *mtry* of variables to consider for splitpoint selection at each node. However, to demonstrate the stability of QRF with respect to this parameter, the default value is used throughout all simulations (where *mtry* is equal to one-third of all variables). Node-sizes are restricted to have more than 10 observations in each node. It is noteworthy that different values of this latter parameter do not seem to change the results very much; nevertheless, it is pointed out in Lin and Jeon (2002) that growing trees until each node is pure (as originally suggested by Breiman) might lead to overfitting.

Linear quantile regression is very similar to standard linear regression and is extensively covered in Koenker (2005). To make linear quantile regression (LQR) more competitive, interaction terms between variables were added for QQR. Starting from the linear model, interaction terms were added by forward selection until the 5-fold cross-validation error attained a minimum.

Next, tree-based methods are considered. Quantile regression trees with piecewise polynomial form were introduced in Chaudhuri and Loh (2002). Software for quantile regression trees comes in the form of the very useful software package GUIDE, available from `www.stat.wisc.edu/~loh/guide.html`, which makes also piecewise constant and piecewise linear quantile regression trees (TRC) available. The default settings are used for both piecewise linear and piecewise second-degree polynomial approximations.

Data Sets The data sets are taken from the packages *mlbench* and *alr3* of the statistical software package R (R Development Core Team, 2005), and include the well-known *Boston Housing* ($p = 13$ variables, $n = 506$ observations), *Ozone* ($p = 12$, $n = 366$, after having removed all missing value observations) and *Abalone* data set ($p = 8$, limited to $n = 500$ randomly chosen observations) from the UCI machine learning repository. In the package *alr3* (Weisberg, 2005), the data set *BigMac* contains the minutes of labor necessary to purchase a Big Mac in $n = 69$ cities worldwide, along with $p = 9$ other variables like tax

rates or primaries teacher net income; these variables are used as predictor variables. Last, the data set *Fuel* lists average gas-mileage for all $n = 51$ American states in the year 2001 (the ratio of total gallons of gasoline sold and the approximate number of miles driven), along with $p = 5$ variables such as gasoline state tax rate and per capita income; again these are used as predictor variables.

Evaluation To measure the quality of the conditional quantile approximations, loss function (3) is used in conjunction with 5-fold cross-validation. The employed loss function measures the weighted absolute deviations between observations and quantiles, instead of the more common squared error loss. The minimum of the loss would be achieved by the true conditional quantile function, as discussed previously. The empirical loss over the test data is computed for all splits of the data sets at quantiles $\alpha \in \{.005, .025, .05, .5, .95, .975, .995\}$. Additionally to the average loss for each method, one might be interested to see whether the difference in performance between quantile regression forests and the other methods is significant or not. To this end bootstrapping is used, comparing each method against quantile regression forests (QRF). The resulting 95% bootstrap confidence intervals for the difference in average loss is shown by vertical bars; if they do not cross the horizontal line (which marks the average loss of QRF), the difference in average loss is statistically significant. Results are shown in Figure 1.

There is not a single data set on which any competing method performs significantly better than quantile regression forests (QRF). However, QRF is quite often significantly better than competing methods. If the difference is not significant, QRF has most often the smaller average loss.

Aggregated trees thus seem to outperform single trees. This is despite the fact that quantile regression trees have to be grown separately for each quantile α, whereas the same set of trees can be used for all quantiles with QRF. The performance of QRF could be even marginally better when growing different set of trees for each value of α. However, this performance enhancement would come at an additional computational price. Moreover, monotonicity of the quantile estimates would not be guaranteed any longer. As it is, the α-quantile of QRF is always at least as large as the β-quantile if $\alpha \geq \beta$. This monotonicity constraint is not always fulfilled for the other considered methods.

Linear quantile regression with interaction terms works surprisingly well in comparison, especially for moderate quantiles (that is α is not too close to either 0 or 1). However, for more extremal quantiles, quantile regression forests delivers most often a better better approximation. This is even more pronounced if additional noise variables are added. To this end, every original predictor variables is permuted randomly and added to the list of predictor variables. The results are shown in Figure 2. The performance of quantile regression forests seems to be robust with respect to inclusion of noise variables.

Prediction Intervals A possible application of quantile regression forests is the construction of prediction intervals, as discussed previously. For each new data point X, a prediction interval of the form (2) gives a range that will cover the new observation of the response variable Y with high probability.

In Figure 3, some graphical results are shown for the Boston Housing data. Figure 4 shows comparable plots for the remaining data sets. There are two main observations: Firstly, as expected, about 95% of all observations are inside their 95% prediction intervals.

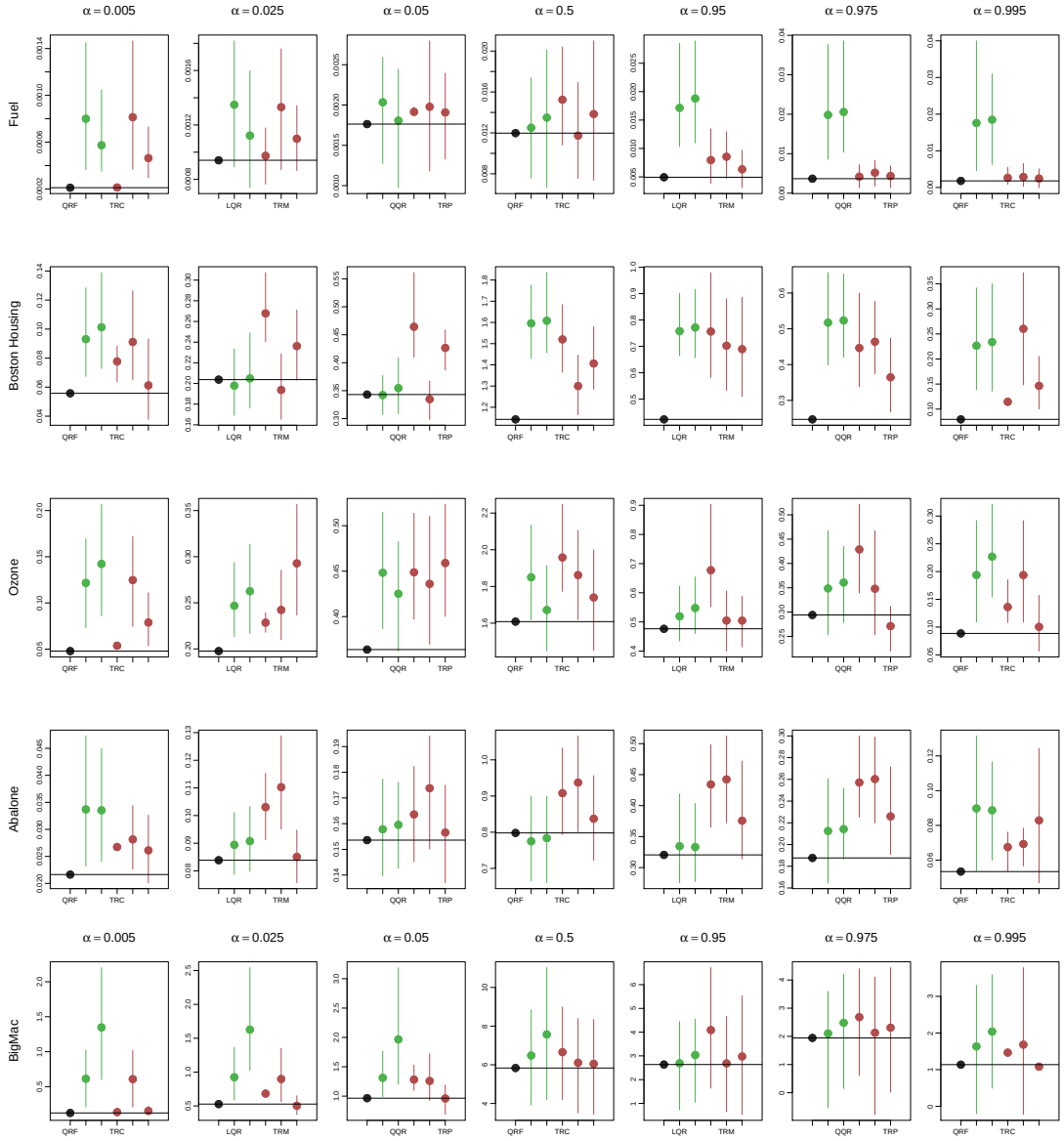


Figure 1: Average loss for various data sets (from top to bottom) and quantiles (from left to right). The average loss of quantile regression forests is shown in each plot as the leftmost dot and is indicated as well by a horizontal line for better comparison. The average losses for competing methods are shown for the linear methods in the middle and the three tree-based methods on the right of each plot. The vertical bars indicate the bootstrap confidence intervals for the difference in average loss for each method against quantile regression forests. Note that no parameters have been fine-tuned for quantile regression forests (the default settings are used throughout).

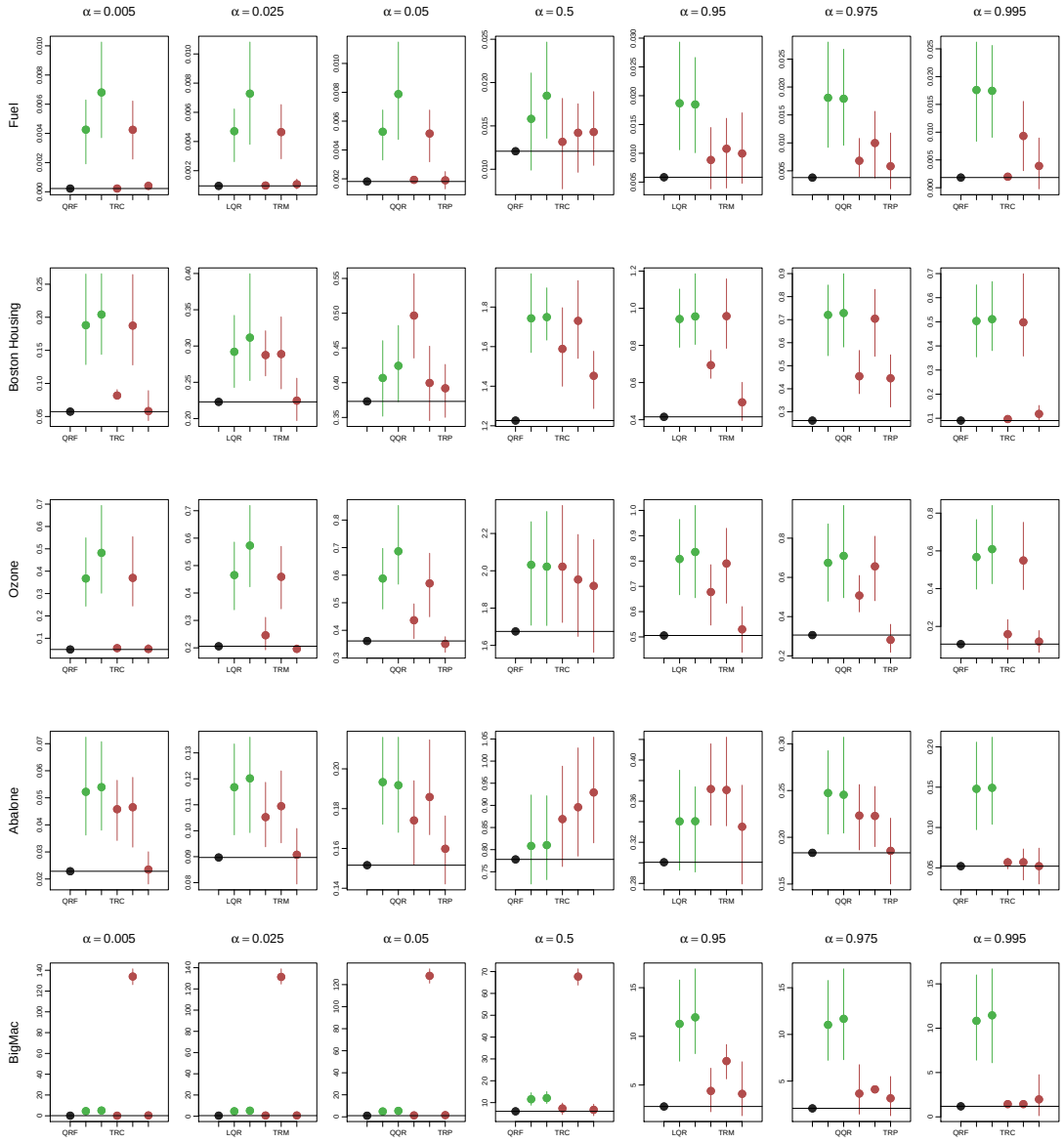


Figure 2: Same plots as in Figure 1. However, to test the performance of the methods under additional noise, each predictor variable is permuted randomly and added to the list of predictor variables.

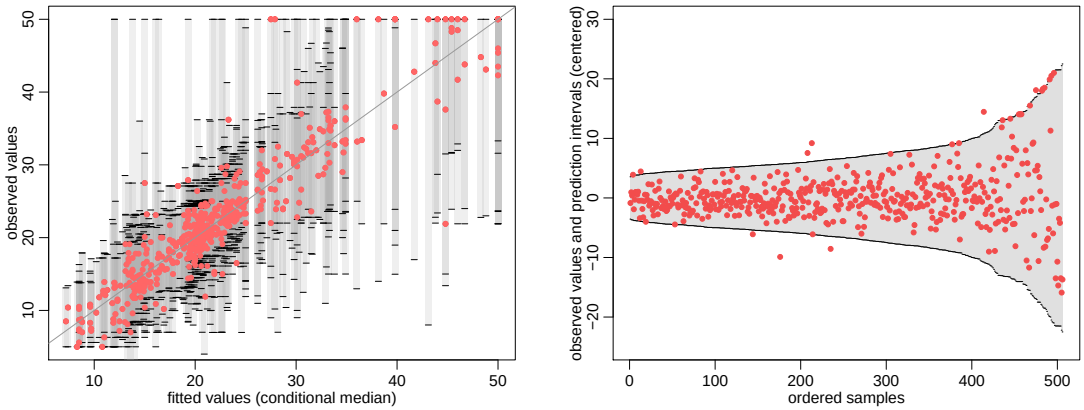


Figure 3: For each data point $i = 1, \ldots, n$ in the Boston Housing data set (with $n = 506$), conditional quantiles are estimated with QRF on a test set which does not include the i-th observation (5-fold cross-validation). Left panel: the observed values are plotted against the predicted median values. Prediction intervals are shown for each $i = 1, \ldots, n$ as transparent grey bars, with vertical black lines at the bottom and top. It can be seen that prediction intervals vary in length, some being much shorter than others. Right panel: For better visualisation, observations $i = 1, \ldots, n$ are ordered according to the length of the corresponding prediction intervals. Moreover, the mean of the upper and lower end of the prediction interval is subtracted from all observations and prediction intervals. All but 10 observations actually lie in their respective 95% prediction intervals.

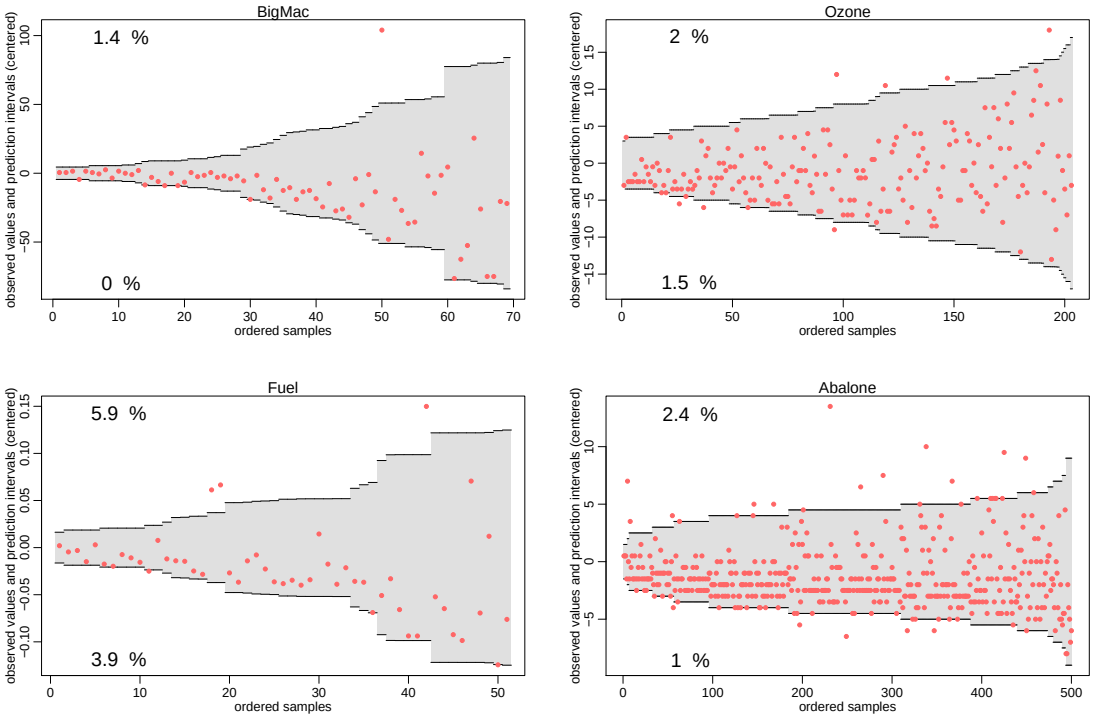


Figure 4: Same plots as in the right panel of Figure 3 for the remaining data sets. Additionally, the percentage of observations that lie above the upper end of their respective prediction intervals (below the lower end) are indicated in the upper left corner (lower left corner). As 95% prediction intervals are shown, on average 2.5% of all observations should be above (and below) their prediction intervals. For the Big Mac, Fuel, and Ozone data sets, it is particularly apparent that the lengths of the prediction intervals vary strongly (some values can thus be predicted more accurately than others).

Secondly, the lengths of prediction intervals vary greatly. Some observations can thus be predicted much more accurately than others.

With quantile regression forests, it is possible to give a range in which each observation is going to be (with high probability). The wider this range for a new instance, the less accurate any prediction is going to be. Vice versa, one knows that a prediction is reliable if the prediction interval is very short.

6. Conclusions

Quantile regression forests infer the full conditional distribution of a response variable. This information can be used to build prediction intervals and detect outliers in the data.

Prediction intervals cover new observations with high probability. The length of the prediction intervals reflect thus the variation of new observations around their predicted values. The accuracy with which new observations can be predicted varies typically quite strongly for instances in the same data set. Quantile regression forests can quantify this accuracy. The estimated conditional distribution is thus a useful addition to the commonly inferred conditional mean of a response variable.

It was shown that quantile regression forests are, under some reasonable assumptions, consistent for conditional quantile estimation. The performance of the algorithm is very competitive in comparison with linear and tree-based methods, as shown for some common Machine Learning benchmark problems.

References

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley and Sons, 1994.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

L. Breiman. Consistency for a simple model of random forests. Technical Report 670, Department of Statistics, University of California, Berkeley, 2004.

L. Breiman and J. H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80:580–598, 1985.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

P. Chaudhuri and W. Loh. Nonparametric estimation of conditional quantiles using quantile regression trees. *Bernoulli*, 8:561–576, 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.

X. He, P. Ng, and S. Portnoy. Bivariate quantile smoothing splines. *Journal of the Royal Statistical Society* B, 3:537–550, 1998.

V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85 – 126, 2004.

P. Huber. Robust regression: asymptotics, conjectures, and monte carlo. *Annals of Statistics*, 1:799–821, 1973.

R. J. Hyndman and Y. Fan. Sample quantiles in statistical packages. *American Statistician*, 50:361–365, 1996.

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

R. Koenker, P. Ng, and S. Portnoy. Quantile smoothing splines. *Biometrika*, 81:673–680, 1994.

Q. V. Le, T. Sears, and A. Smola. Nonparametric quantile regression. Technical report, NICTA, 2005.

Andy Liaw and Matthew Wiener. Classification and regression by randomForest. *R News*, 2:18–22, 2002.

Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. Technical Report 1055, Department of Statistics, University of Wisconsin, 2002.

M. Markou and S. Singh. Novelty detection: A review. *Signal Processing*, 83:2481–2497, 2003.

S Portnoy and R. Koenker. The gaussian hare and the laplacian tortoise: Computability of squared-error versus absolute-error estimates. *Statistical Science*, 12:279–300, 1997.

R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651–1686, 1998.

I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6:211–232, 2005.

S. Weisberg. *Applied Linear Regression.* Wiley, 2005.

Journal of Machine Learning Research 7 (2006) 1001–1024 Submitted 08/05; Revised 3/06; Published 6/06

Sparse Boosting

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar für Statistik
ETH Zürich
Zürich, CH-8092, Switzerland

Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
University of California
Berkeley, CA 94720-3860, USA

Editors: Yoram Singer and Larry Wasserman

Abstract

We propose Sparse Boosting (the SparseL_2Boost algorithm), a variant on boosting with the squared error loss. SparseL_2Boost yields sparser solutions than the previously proposed L_2Boosting by minimizing some penalized L_2-loss functions, the *FPE* model selection criteria, through small-step gradient descent. Although boosting may give already relatively sparse solutions, for example corresponding to the soft-thresholding estimator in orthogonal linear models, there is sometimes a desire for more sparseness to increase prediction accuracy and ability for better variable selection: such goals can be achieved with SparseL_2Boost.

We prove an equivalence of SparseL_2Boost to Breiman's nonnegative garrote estimator for orthogonal linear models and demonstrate the generic nature of SparseL_2Boost for nonparametric interaction modeling. For an automatic selection of the tuning parameter in SparseL_2Boost we propose to employ the gMDL model selection criterion which can also be used for early stopping of L_2Boosting. Consequently, we can select between SparseL_2Boost and L_2Boosting by comparing their gMDL scores.

Keywords: lasso, minimum description length (MDL), model selection, nonnegative garrote, regression

1. Introduction

Since its inception in a practical form in Freund and Schapire (1996), boosting has obtained and maintained its outstanding performance in numerous empirical studies both in the machine learning and statistics literatures. The gradient descent view of boosting as articulated in Breiman (1998, 1999), Friedman et al. (2000) and Rätsch et al. (2001) provides a springboard for the understanding of boosting to leap forward and at the same time serves as the base for new variants of boosting to be generated. In particular, the L_2Boosting (Friedman, 2001) takes the simple form of refitting a base learner to residuals of the previous iteration. It coincides with Tukey's (1977) twicing at its second iteration and reproduces matching pursuit of Mallat and Zhang (1993) when applied to a dictionary or collection of fixed basis functions. A somewhat different approach has been suggested by Rätsch et al. (2002). Bühlmann and Yu (2003) investigated L_2Boosting for linear base procedures (weak learners) and showed that in such cases, the variance or complexity of the boosted procedure is bounded and increases at an increment which is exponentially diminishing as iterations run – this

special case calculation implies that the resistance to the over-fitting behavior of boosting could be due to the fact that the complexity of boosting increases at an extremely slow pace.

Recently Efron et al. (2004) made an intriguing connection for linear models between L_2Boosting and Lasso (Tibshirani, 1996) which is an ℓ^1-penalized least squares method. They consider a modification of L_2Boosting, called forward stagewise least squares (FSLR) and they show that for some special cases, FSLR with infinitesimally small step-sizes produces a set of solutions which coincides with the set of Lasso solutions when varying the regularization parameter in Lasso. Furthermore, Efron et al. (2004) proposed the least angle regression (LARS) algorithm whose variants give a clever computational short-cut for FSLR and Lasso.

For high-dimensional linear regression (or classification) problems with many ineffective predictor variables, the Lasso estimate can be very poor in terms of prediction accuracy and as a variable selection method, see Meinshausen (2005). There is a need for more sparse solutions than produced by the Lasso. Our new SparseL_2Boost algorithm achieves a higher degree of sparsity while still being computationally feasible, in contrast to all subset selection in linear regression whose computational complexity would generally be exponential in the number of predictor variables. For the special case of orthogonal linear models, we prove here an equivalence of SparseL_2Boost to Breiman's (1995) nonnegative garrote estimator. This demonstrates the increased sparsity of SparseL_2Boost over L_2Boosting which is equivalent to soft-thresholding (due to Efron et al. (2004) and Theorem 2 in this article).

Unlike Lasso or the nonnegative garrote estimator, which are restricted to a (generalized) linear model or basis expansion using a fixed dictionary, SparseL_2Boost enjoys much more generic applicability while still being computationally feasible in high-dimensional problems and yielding more sparse solutions than boosting or ℓ^1-regularized versions thereof (see Rätsch et al., 2002; Lugosi and Vayatis, 2004). In particular, we demonstrate its use in the context of nonparametric second-order interaction modeling with a base procedure (weak learner) using thin plate splines, improving upon Friedman's (1991) MARS.

Since our SparseL_2Boost is based on the final prediction error criterion, it opens up the possibility of bypassing the computationally intensive cross-validation by stopping early based on the model selection score. The gMDL model selection criterion (Hansen and Yu, 2001) uses a data-driven penalty to the L_2-loss and as a consequence bridges between the two well-known AIC and BIC criteria. We use it in the SparseL_2Boost algorithm and for early stopping of L_2Boosting. Furthermore, we can select between SparseL_2Boost and L_2Boosting by comparing their gMDL scores.

2. Boosting with the Squared Error Loss

We assume that the data are realizations from

$$(X_1, Y_1), \ldots, (X_n, Y_n),$$

where $X_i \in \mathbb{R}^p$ denotes a p-dimensional predictor variable and $Y_i \in \mathbb{R}$ a univariate response. In the sequel, we denote by $x^{(j)}$ the jth component of a vector $x \in \mathbb{R}^p$. We usually assume that the pairs (X_i, Y_i) are i.i.d. or from a stationary process. The goal is to estimate the regression function $F(x) = \mathbb{E}[Y|X = x]$ which is well known to be the (population) minimizer of the expected squared error loss $\mathbb{E}[(Y - F(X))^2]$.

The boosting methodology in general builds on a user-determined base procedure or weak learner and uses it repeatedly on modified data which are typically outputs from the previous it-

erations. The final boosted procedure takes the form of linear combinations of the base procedures. For L_2Boosting, based on the squared error loss, one simply fits the base procedure to the original data to start with, then uses the residuals from the previous iteration as the new response vector and refits the base procedure, and so on. As we will see in section 2.2, L_2Boosting is a "constrained" minimization of the empirical squared error risk $n^{-1}\sum_{i=1}^{n}(Y_i - F(X_i))^2$ (with respect to $F(\cdot)$) which yields an estimator $\hat{F}(\cdot)$. The regularization of the empirical risk minimization comes in implicitly by the choice of a base procedure and by algorithmical constraints such as early stopping or penalty barriers.

2.1 Base Procedures Which Do Variable Selection

To be more precise, a base procedure is in our setting a function estimator based on the data $\{(X_i, U_i);\ i = 1,\ldots,n\}$, where $U_1,\ldots,U_n$ denote some (pseudo-) response variables which are not necessarily the original $Y_1,\ldots,Y_n$. We denote the base procedure function estimator by

$$\hat{g}(\cdot) = \hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot), \tag{1}$$

where $\mathbf{X} = (X_1,\ldots,X_n)$ and $\mathbf{U} = (U_1,\ldots,U_n)$.

Many base procedures involve some variable selection. That is, only some of the components of the p-dimensional predictor variables X_i are actually contributing in (1). In fact, almost all of the successful boosting algorithms in practice involve base procedures which do variable selection: examples include decision trees (see Freund and Schapire, 1996; Breiman, 1998; Friedman et al., 2000; Friedman, 2001), componentwise smoothing splines which involve selection of the best single predictor variable (see Bühlmann and Yu, 2003), or componentwise linear least squares in linear models with selection of the best single predictor variable (see Mallat and Zhang, 1993; Bühlmann, 2006).

It will be useful to represent the base procedure estimator (at the observed predictors X_i) as a hat-operator, mapping the (pseudo-) response to the fitted values:

$$\mathcal{H} : \mathbf{U} \mapsto (\hat{g}_{(\mathbf{X},\mathbf{U})}(X_1),\ldots,\hat{g}_{(\mathbf{X},\mathbf{U})}(X_n)),\ \mathbf{U} = (U_1,\ldots,U_n).$$

If the base procedure selects from a set of predictor variables, we denote the selected predictor variable index by $\hat{S} \subset \{1,\ldots,p\}$, where $\hat{S}$ has been estimated from a specified set Γ of subsets of variables. To emphasize this, we write for the hat operator of a base procedure

$$\mathcal{H}_{\hat{S}} : \mathbf{U} \mapsto (\hat{g}_{(\mathbf{X}^{(\hat{S})},\mathbf{U})}(X_1),\ldots,\hat{g}_{(\mathbf{X}^{(\hat{S})},\mathbf{U})}(X_n)),\ \mathbf{U} = (U_1,\ldots,U_n), \tag{2}$$

where the base procedure $\hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot) = \hat{g}_{(\mathbf{X}^{(\hat{S})},\mathbf{U})}(\cdot)$ depends only on the components $\mathbf{X}^{(\hat{S})}$ from $\mathbf{X}$. The examples below illustrate this formalism.

Componentwise linear least squares in linear model (see Mallat and Zhang, 1993; Bühlmann, 2006)
We select only single variables at a time from $\Gamma = \{1,2,\ldots,p\}$. The selector $\hat{S}$ chooses the predictor variable which reduces the residual sum of squares most when using least squares fitting:

$$\hat{S} = \text{argmin}_{1\le j\le p}\sum_{i=1}^{n}(U_i - \hat{\gamma}_j X_i^{(j)})^2,\ \hat{\gamma}_j = \frac{\sum_{i=1}^n U_i X_i^{(j)}}{\sum_{i=1}^n (X_i^{(j)})^2}\ (j = 1,\ldots,p).$$

The base procedure is then

$$\hat{g}_{(\mathbf{X},\mathbf{U})}(x) = \hat{\gamma}_{\hat{S}} x^{(\hat{S})},$$

and its hat operator is given by the matrix

$$\mathcal{H}_{\hat{S}} = \mathbf{X}^{(\hat{S})}(\mathbf{X}^{(\hat{S})})^T,\ \mathbf{X}^{(j)} = (X_1^{(j)},\ldots,X_n^{(j)})^T.$$

L_2Boosting with this base procedure yields a linear model with model selection and parameter estimates which are shrunken towards zero. More details are given in sections 2.2 and 2.4.

Componentwise smoothing spline (see Bühlmann and Yu, 2003)
Similarly to a componentwise linear least squares fit, we select only one single variable at a time from $\Gamma = \{1,2,\ldots,p\}$. The selector $\hat{S}$ chooses the predictor variable which reduces residual sum of squares most when using a smoothing spline fit. That is, for a given smoothing spline operator with fixed degrees of freedom `d.f.` (which is the trace of the corresponding hat matrix)

$$\hat{S} = \text{argmin}_{1\le j\le p}\sum_{i=1}^{n}(U_i - \hat{g}_j(X_i^{(j)}))^2,$$

$\hat{g}_j(\cdot)$ is the fit from the smoothing spline to $\mathbf{U}$ versus $\mathbf{X}^{(j)}$ with `d.f.`

Note that we use the same degrees of freedom `d.f.` for all components j's. The hat-matrix corresponding to $\hat{g}_j(\cdot)$ is denoted by $\mathcal{H}_j$ which is symmetric; the exact from is not of particular interest here but is well known, see Green and Silverman (1994). The base procedure is

$$\hat{g}_{(\mathbf{X},\mathbf{U})}(x) = \hat{g}_{\hat{S}}(x^{(\hat{S})}),$$

and its hat operator is then given by a matrix $\mathcal{H}_{\hat{S}}$. Boosting with this base procedure yields an additive model fit based on selected variables (see Bühlmann and Yu, 2003).

Pairwise thin plate splines
Generalizing the componentwise smoothing spline, we select pairs of variables from $\Gamma = \{(j,k);\ 1 \le j < k \le p\}$. The selector $\hat{S}$ chooses the two predictor variables which reduce residual sum of squares most when using thin plate splines with two arguments:

$$\hat{S} = \text{argmin}_{1\le j<k\le p}\sum_{i=1}^{n}(U_i - \hat{g}_{j,k}(X_i^{(j)},X_i^{(k)}))^2,$$

$\hat{g}_{j,k}(\cdot,\cdot)$ is an estimated thin plate spline based on $\mathbf{U}$ and $\mathbf{X}^{(j)},\mathbf{X}^{(k)}$ with `d.f.`,

where the degrees of freedom `d.f.` is the same for all components $j < k$. The hat-matrix corresponding to $\hat{g}_{j,k}$ is denoted by $\mathcal{H}_{j,k}$ which is symmetric; again the exact from is not of particular interest but can be found in Green and Silverman (1994). The base procedure is

$$\hat{g}_{(\mathbf{X},\mathbf{U})}(x) = \hat{g}_{\hat{S}}(x^{(\hat{S})}),$$

where $x^{(\hat{S})}$ denotes the 2-dimensional vector corresponding to the selected pair in $\hat{S}$, and the hat operator is then given by a matrix $\mathcal{H}_{\hat{S}}$. Boosting with this base procedure yields a nonparametric fit with second order interactions based on selected pairs of variables; an illustration is given in section 3.4.

In all the examples above, the selector is given by

$$\hat{S} = \text{argmin}_{S\in\Gamma} \sum_{i=1}^{n} (U_i - (\mathcal{H}_S \mathbf{U})_i)^2 \tag{3}$$

Also (small) regression trees can be cast into this framework. For example for stumps, $\Gamma = \{(j, c_{j,k});\ j = 1,\ldots,p,\ k = 1,\ldots,n-1\}$, where $c_{j,1} < \ldots < c_{j,n-1}$ are the mid-points between (non-tied) observed values $X_i^{(j)}$ $(i = 1,\ldots,n)$. That is, Γ denotes here the set of selected single predictor variables and corresponding split-points. The parameter values for the two terminal nodes in the stump are then given by ordinary least squares which implies a linear hat matrix $\mathcal{H}_{(j,c_{j,k})}$. Note however, that for mid-size or large regression trees, the optimization over the set Γ is usually not done exhaustively.

2.2 L_2Boosting

Before introducing our new SparseL_2Boost algorithm, we describe first its less sparse counterpart L_2Boosting, a boosting procedure based on the squared error loss which amounts to repeated fitting of residuals with the base procedure $\hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot)$. Its derivation from a more general functional gradient descent algorithm using the squared error loss has been described by many authors, see Friedman (2001).

L_2Boosting

Step 1 (initialization). $\hat{F}_0(\cdot) \equiv 0$ and set $m = 0$.

Step 2. Increase m by 1.
Compute residuals $U_i = Y_i - \hat{F}_{m-1}(X_i)$ $(i = 1,\ldots,n)$ and fit the base procedure to the current residuals. The fit is denoted by $\hat{f}_m(\cdot) = \hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot)$.
Update

$$\hat{F}_m(\cdot) = \hat{F}_{m-1}(\cdot) + \nu \hat{f}_m(\cdot),$$

where $0 < \nu \leq 1$ is a pre-specified step-size parameter.

Step 3 (iteration). Repeat Steps 2 and 3 until some stopping value for the number of iterations is reached.

With $m = 2$ and $\nu = 1$, L_2Boosting has already been proposed by Tukey (1977) under the name "twicing". The number of iterations is the main tuning parameter for L_2Boosting. Empirical evidence suggests that the choice for the step-size ν is much less crucial as long as ν is small; we usually use $\nu = 0.1$. The number of boosting iterations may be estimated by cross-validation. As an alternative, we will develop in section 2.5 an approach which allows to use some model selection criteria to bypass cross-validation.

2.3 SparseL_2Boost

As described above, L_2Boosting proceeds in a greedy way: if in Step2 the base procedure is fitted by least squares and when using $\nu = 1$, L_2Boosting pursues the best reduction of residual sum of squares in every iteration.

Alternatively, we may want to proceed such that the out-of-sample prediction error would be most reduced, that is we would like to fit a function $\hat{g}_{\mathbf{X},\mathbf{U}}$ (from the class of weak learner estimates) such that the out-of-sample prediction error becomes minimal. This is not exactly achievable since the out-sample prediction error is unknown. However, we can estimate it via a model selection criterion. To do so, we need a measure of complexity of boosting. Using the notation as in (2), the L_2Boosting operator in iteration m is easily shown to be (see Bühlmann and Yu, 2003)

$$\mathcal{B}_m = I - (I - \nu \mathcal{H}_{\hat{S}_m}) \cdot \cdots \cdot (I - \nu \mathcal{H}_{\hat{S}_1}), \tag{4}$$

where $\hat{S}_m$ denotes the selector in iteration m. Moreover, if all the $\mathcal{H}_S$ are linear (that is the hat matrix), as in all the examples given in section 2.1, L_2Boosting has an approximately linear representation, where only the data-driven selector $\hat{S}$ brings in some additional nonlinearity. Thus, in many situations (for example the examples in the previous section 2.1 and decision tree base procedures), the boosting operator has a corresponding matrix-form when using in (4) the hat-matrices for $\mathcal{H}_S$. The degrees of freedom for boosting are then defined as

$$\text{trace}(\mathcal{B}_m) = \text{trace}(I - (I - \nu \mathcal{H}_{\hat{S}_m}) \cdots (I - \nu \mathcal{H}_{\hat{S}_1})).$$

This is a standard definition for degrees of freedom (see Green and Silverman, 1994) and it has been used in the context of boosting in Bühlmann (2006). An estimate for the prediction error of L_2Boosting in iteration m can then be given in terms of the final prediction error criterion FPE_γ (Akaike, 1970):

$$\sum_{i=1}^{n} (Y_i - \hat{F}_m(X_i))^2 + \gamma \cdot \text{trace}(\mathcal{B}_m). \tag{5}$$

2.3.1 THE SPARSEL_2BOOST ALGORITHM

For SparseL_2Boost, the penalized residual sum of squares in (5) becomes the criterion to move from iteration $m-1$ to iteration m. More precisely, for $\mathcal{B}$ a (boosting) operator, mapping the response vector $\mathbf{Y}$ to the fitted variables, and a criterion $C(RSS, k)$, we use the following objective function to boost:

$$T(\mathbf{Y}, \mathcal{B}) = C\left(\sum_{i=1}^{n} (Y_i - (\mathcal{B}\mathbf{Y})_i)^2, \text{trace}(\mathcal{B})\right). \tag{6}$$

For example, the criterion could be FPE_γ for some $\gamma > 0$ which corresponds to

$$C_\gamma(RSS, k) = RSS + \gamma \cdot k. \tag{7}$$

An alternative which does not require the specification of a parameter γ as in (7) is advocated in section 2.5.

The algorithm is then as follows.

SparseL_2Boost

Step 1 (initialization). $\hat{F}_0(\cdot) \equiv 0$ and set $m = 0$.

Step 2. Increase m by 1.
Search for the best selector

$$\begin{aligned}
&\tilde{S}_m = \text{argmin}_{S \in \Gamma} T(\mathbf{Y}, \text{trace}(\mathcal{B}_m(S))), \\
&\mathcal{B}_m(S) = I - (I - \mathcal{H}_S)(I - \nu \mathcal{H}_{\tilde{S}_{m-1}}) \cdots (I - \nu \mathcal{H}_{\tilde{S}_1}), \\
&(\text{for } m = 1\text{: } \mathcal{B}_1(S) = \mathcal{H}_S).
\end{aligned}$$

Fit the residuals $U_i = Y_i - \hat{F}_{m-1}(X_i)$ with the base procedure using the selected $\tilde{S}_m$ which yields a function estimate

$$\hat{f}_m(\cdot) = \hat{g}_{\tilde{S}_m;(\mathbf{X},\mathbf{U})}(\cdot),$$

where $\hat{g}_{S;(\mathbf{X},\mathbf{U})}(\cdot)$ corresponds to the hat operator $\mathcal{H}_S$ from the base procedure.

Step 3 (update). Update,

$$\hat{F}_m(\cdot) = \hat{F}_{m-1}(\cdot) + \nu \hat{f}_m(\cdot).$$

Step 4 (iteration). Repeat Steps 2 and 3 for a large number of iterations M.

Step 5 (stopping). Estimate the stopping iteration by

$$\hat{m} = \text{argmin}_{1 \le m \le M} T(\mathbf{Y}, \text{trace}(\mathcal{B}_m)), \quad \mathcal{B}_m = I - (I - \nu \mathcal{H}_{\tilde{S}_m}) \cdots (I - \nu \mathcal{H}_{\tilde{S}_1}).$$

The final estimate is $\hat{F}_{\hat{m}}(\cdot)$.

The only difference to L_2Boosting is that the selection in Step 2 yields a different $\tilde{S}_m$ than in (3). While $\hat{S}_m$ in (3) minimizes the residual sum of squares, the selected $\tilde{S}_m$ in SparseL_2Boost minimizes a model selection criterion over all possible selectors. Since the selector $\tilde{S}_m$ depends not only on the current residuals $\mathbf{U}$ but also explicitly on all previous boosting iterations through $\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_{m-1}$ via the trace of $\mathcal{B}_m(S)$, the estimate $\hat{f}_m(\cdot)$ in SparseL_2Boost is not a function of the current residuals $\mathbf{U}$ only. This implies that we cannot represent SparseL_2Boost as a linear combination of base procedures, each of them acting on residuals only.

2.4 Connections to the Nonnegative Garrote Estimator

SparseL_2Boost based on C_γ as in (7) enjoys a surprising equivalence to the nonnegative garrote estimator (Breiman, 1995) in an orthogonal linear model. This special case allows explicit expressions to reveal clearly that SparseL_2Boost (aka nonnegative-garrote) is sparser than L_2Boosting (aka soft-thresholding).

Consider a linear model with n orthonormal predictor variables,

$$Y_i = \sum_{j=1}^n \beta_j x_i^{(j)} + \varepsilon_i,\ i = 1,\ldots,n,$$
$$\sum_{i=1}^n x_i^{(j)} x_i^{(k)} = \delta_{jk}, \tag{8}$$

where δ_{jk} denotes the Kronecker symbol, and $\varepsilon_1,\ldots,\varepsilon_n$ are i.i.d. random variables with $\mathbb{E}[\varepsilon_i] = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma_\varepsilon^2 < \infty$. We assume here the predictor variables as fixed and non-random. Using the standard regression notation, we can re-write model (8) as

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon,\ \ \mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = I, \tag{9}$$

with the $n \times n$ design matrix $\mathbf{X} = (x_i^{(j)})_{i,j=1,\ldots,n}$, the parameter vector $\beta = (\beta_1,\ldots,\beta_n)^T$, the response vector $\mathbf{Y} = (Y_1,\ldots,Y_n)^T$ and the error vector $\varepsilon = (\varepsilon_1,\ldots,\varepsilon_n)^T$. The predictors could also be basis functions $g_j(t_i)$ at observed values t_i with the property that they build an orthonormal system.

The nonnegative garrote estimator has been proposed by Breiman (1995) for a linear regression model to improve over subset selection. It shrinks each ordinary least squares (OLS) estimated coefficient by a nonnegative amount whose sum is subject to an upper bound constraint (the garrote). For a given response vector $\mathbf{Y}$ and a design matrix $\mathbf{X}$ (see (9)), the nonnegative garrote estimator takes the form

$$\hat{\beta}_{Nngar,j} = c_j \hat{\beta}_{OLS,j}$$

such that

$$\sum_{i=1}^n (Y_i - (\mathbf{X}\hat{\beta}_{Nngar})_i)^2 \text{ is minimized, subject to } c_j \geq 0,\ \sum_{j=1}^p c_j \leq s, \tag{10}$$

for some $s > 0$. In the orthonormal case from (8), since the ordinary least squares estimator is simply $\hat{\beta}_{OLS,j} = (\mathbf{X}^T\mathbf{Y})_j = Z_j$, the nonnegative garrote minimization problem becomes finding c_j's such that

$$\sum_{j=1}^n (Z_j - c_j Z_j)^2 \text{ is minimized, subject to } c_j \geq 0,\ \sum_{j=1}^n c_j \leq s.$$

Introducing a Lagrange multiplier $\tau > 0$ for the sum constraint gives the dual optimization problem: minimizing

$$\sum_{j=1}^n (Z_j - c_j Z_j)^2 + \tau \sum_{j=1}^n c_j,\quad c_j \geq 0\ (j = 1,...,n). \tag{11}$$

This minimization problem has an explicit solution (Breiman, 1995):

$$c_j = (1 - \lambda/|Z_j|^2)^+, \ \lambda = \tau/2,$$

where $u^+ = \max(0,u)$. Hence $\hat{\beta}_{Nngar,j} = (1-\lambda/|Z_j|^2)^+ Z_j$ or equivalently,

$$\hat{\beta}_{Nngar,j} = \begin{cases} Z_j - \lambda/|Z_j|, & \text{if } \operatorname{sign}(Z_j) Z_j^2 \geq \lambda, \\ 0, & \text{if } Z_j^2 < \lambda, \\ Z_j + \lambda/|Z_j|, & \text{if } \operatorname{sign}(Z_i) Z_j^2 \leq -\lambda. \end{cases}, \quad \text{where } Z_j = (\mathbf{X}^T \mathbf{Y})_j. \tag{12}$$

We show in Figure 1 the nonnegative garrote threshold function in comparison to hard- and soft-thresholding, the former corresponding to subset variable selection and the latter to the Lasso (Tibshirani, 1996). Hard-thresholding either yields the value zero or the ordinary least squares estimator; the nonnegative garrote and soft-thresholding either yield the value zero or a shrunken ordinary least squares estimate, where the shrinkage towards zero is stronger for the soft-threshold than for the nonnegative garrote estimator. Therefore, for the same amount of "complexity" or "degrees of freedom" (which is in case of hard-thresholding the number of ordinary least squares estimated variables), hard-thresholding (corresponding to subset selection) will typically select the fewest number of variables (non-zero coefficient estimates) while the nonnegative garrote will include more variables and the soft-thresholding will be the least sparse in terms of the number of selected variables; the reason is that for the non-zero coefficient estimates, the shrinkage effect, which is slight in the nonnegative garotte and stronger for soft-thresholding, causes fewer degrees of freedom for every

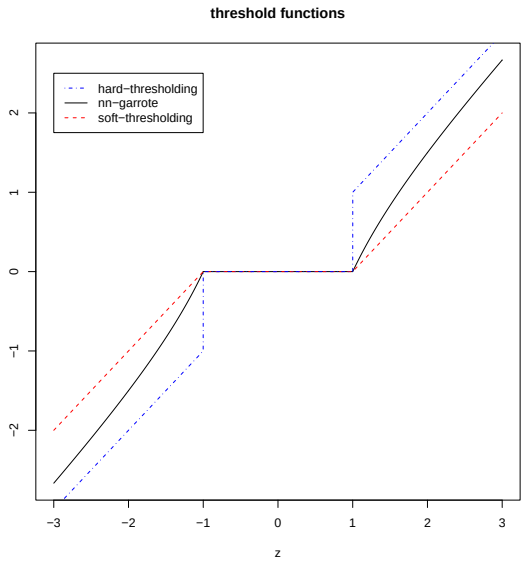


Figure 1: Threshold functions for subset selection or hard-thresholding (dashed-dotted line), nonnegative garrote (solid line) and lasso or soft-thresholding (dashed line).

selected variable. This observation can also be compared with some numerical results in section 3.

The following result shows the equivalence of the nonnegative garrote estimator and SparseL_2Boost with componentwise linear least squares (using $\hat{m}$ iterations) yielding coefficient estimates $\hat{\beta}^{(\hat{m})}_{SparseBoost,j}$.

Theorem 1 *Consider the model in (8) and any sequence $(\gamma_n)_{n\in\mathbb{N}}$. For Sparse$L_2$Boost with componentwise linear least squares, based on C_{γ_n} as in (7) and using a step-size ν, as described in section 2.3, we have*

$$\hat{\beta}^{(\hat{m})}_{SparseBoost,j} = \hat{\beta}_{Nngar,j} \text{ in (12) with parameter } \lambda_n = \frac{1}{2}\gamma_n(1+e_j(\nu)),$$
$$\max_{1\le i\le n}|e_j(\nu)| \le \nu/(1-\nu) \to 0 \ (\nu \to 0).$$

A proof is given in section 5. Note that the sequence $(\gamma_n)_{n\in\mathbb{N}}$ can be arbitrary and does not need to depend on n (and likewise for the corresponding λ_n). For the orthogonal case, Theorem 1 yields the interesting interpretation of SparseL_2Boost as the nonnegative garrote estimator.

We also describe here for the orthogonal case the equivalence of L_2Boosting with componentwise linear least squares (yielding coefficient estimates $\hat{\beta}^{(m)}_{Boost,j}$) to soft-thresholding. A closely related result has been given in Efron et al. (2004) for the forward stagewise linear regression method which is similar to L_2Boosting. However, our result is for (non-modified) L_2Boosting and brings out more explicitly the role of the step-size.

The soft-threshold estimator for the unknown parameter vector β, is

$$\hat{\beta}_{soft,j} = \begin{cases} Z_j - \lambda, & \text{if } Z_j \ge \lambda, \\ 0, & \text{if } |Z_j| < \lambda, \\ Z_j + \lambda, & \text{if } Z_j \le -\lambda. \end{cases} \qquad \text{where } Z_j = (\mathbf{X}^T\mathbf{Y})_j. \tag{13}$$

Theorem 2 *Consider the model in (8) and a threshold λ_n in (13) for any sequence $(\lambda_n)_{n\in\mathbb{N}}$. For L_2Boosting with componentwise linear least squares and using a step-size ν, as described in section 2.2, there exists a boosting iteration m, typically depending on λ_n, ν and the data, such that*

$$\hat{\beta}^{(m)}_{Boost,j} = \hat{\beta}_{soft,j} \text{ in (13) with threshold of the form } \lambda_n(1+e_j(\nu)), \text{ where}$$
$$\max_{1\le j\le n}|e_j(\nu)| \le \nu/(1-\nu) \to 0 \ (\nu \to 0).$$

A proof is given in section 5. We emphasize that the sequence $(\lambda_n)_{n\in\mathbb{N}}$ can be arbitrary: in particular, λ_n does not need to depend on sample size n.

2.5 The gMDL choice for the criterion function

The FPE criterion function $C(\cdot,\cdot)$ in (7) requires in practice the choice of a parameter γ. In principle, we could tune this parameter using some cross-validation scheme. Alternatively, one could use a parameter value corresponding to well-known model selection criteria such as AIC ($\gamma = 2$) or BIC ($\gamma = \log n$). However, in general, the answer to whether to use AIC or BIC depends on the true underlying model being finite or not (see Speed and Yu, 1993, and the references therein). In practice, it is difficult to know which situation one is in and thus hard to choose between AIC and BIC. We employ here instead a relatively new minimum description length criterion, gMDL (see Hansen and Yu, 2001), developed for linear models. For each model class, roughly speaking, gMDL is derived as a mixture code length based on a linear model with an inverse Gamma prior

(with a shape hyperparameter) for the variance and conditioning on the variance, the linear model parameter β follows an independent multivariate normal prior with the given variance multiplied by a scale hyperparameter. The two hyperparameters are then optimized based on the MDL principle and their coding costs are included in the code length. Because of the adaptive choices of the hyperparameters, the resulted gMDL criterion has a data-dependent penalty for each dimension, instead of the fixed penalty 2 or $\log n$ for AIC or BIC, respectively. In other words, gMDL bridges the AIC and BIC criteria by having a data-dependent penalty $\log(F)$ as given below in (14). The F in the gMDL penalty is related to the signal to noise ratio (SNR), as shown in Hansen and Yu (1999). Moreover, the gMDL criterion has an explicit analytical expression which depends only on the residual sum of squares and the model dimension or complexity. It is worth noting that we will not need to tune the criterion function as it will be explicitly given as a function of the data only. The gMDL criterion function takes the form

$$C_{gMDL}(RSS,k) = \log(S) + \frac{k}{n}\log(F),$$
$$S = \frac{RSS}{n-k},\ F = \frac{\sum_{i=1}^{n} Y_i^2 - RSS}{kS}. \tag{14}$$

Here, *RSS* denotes again the residual sum of squares as in formula (6) (first argument of the function $C(\cdot,\cdot)$).

In the SparseL_2Boost algorithm in section 2.3.1, if we take

$$T(\mathbf{Y},\mathcal{B}) = C_{gMDL}(RSS, \text{trace}(\mathcal{B})),$$

then we arrive at the **gMDL-SparseL_2Boost** algorithm. Often though, we simply refer to it as SparseL_2Boost.

The gMDL criterion in (14) can also be used to give a new stopping rule for L_2Boosting. That is, we propose

$$\hat{m} = \text{argmin}_{1\le m\le M} C_{gMDL}(RSS_m, \text{trace}(\mathcal{B}_m)), \tag{15}$$

where M is a large number, RSS_m the residual sum of squares after m boosting iterations and $\mathcal{B}_m$ is the boosting operator described in (4). If the minimizer is not unique, we use the minimal m which minimizes the criterion. Boosting can now be run without tuning any parameter (we typically do not tune over the step-size ν but rather take a value such as $\nu = 0.1$), and we call such an automatically stopped boosting method **gMDL-L_2Boosting**. In the sequel, it is simply referred to as L_2Boosting.

There will be no overall superiority of either SparseL_2Boost or L_2Boosting as shown in Section 3.1. But it is straightforward to do a data-driven selection: we choose the fitted model which has the smaller gMDL-score between gMDL-SparseL_2Boost and the gMDL stopped L_2Boosting. We term this method **gMDL-sel-L_2Boost** which does not rely on cross-validation and thus could bring much computational savings.

3. Numerical Results

In this section, we investigate and compare SparseL_2Boost with L_2Boosting (both with their data-driven gMDL-criterion), and evaluate gMDL-sel-L_2Boost. The step-size in both boosting methods is fixed at $\nu = 0.1$. The simulation models are based on two high-dimensional linear models and one nonparametric model. Except for two real data sets, all our comparisons and results are based on 50 independent model simulations.

3.1 High-Dimensional Linear Models

3.1.1 ℓ^0-SPARSE MODELS

Consider the model

$$Y = 1 + 5X_1 + 2X_2 + X_9 + \varepsilon,$$
$$X = (X_1, \ldots, X_{p-1}) \sim \mathcal{N}_{p-1}(0, \Sigma),\ \varepsilon \sim \mathcal{N}(0,1), \tag{16}$$

where ε is independent from X. The sample size is chosen as $n = 50$ and the predictor-dimension is $p \in \{50, 100, 1000\}$. For the covariance structure of the predictor X, we consider two cases:

$$\Sigma = I_{p-1}, \tag{17}$$
$$[\Sigma]_{ij} = 0.8^{|i-j|}. \tag{18}$$

The models are ℓ^0-sparse, since the ℓ^0-norm of the true regression coefficients (the number of effective variables including an intercept) is 4.

The predictive performance is summarized in Table 1. For the ℓ^0-sparse model (16), SparseL_2Boost outperforms L_2Boosting. Furthermore, in comparison to the oracle performance (denoted by an asterisk $*$ in Table 1), the gMDL rule for the stopping iteration $\hat{m}$ works very well for the lower-dimensional cases with $p \in \{50, 100\}$ and it is still reasonably accurate for the very high-dimensional case with $p = 1000$. Finally, both boosting methods are essentially insensitive when increasing the

Σ , dim.	SparseL_2Boost	L_2Boosting	SparseL_2Boost*	L_2Boosting*
(17), $p = 50$	0.16 (0.018)	0.46 (0.041)	0.16 (0.018)	0.46 (0.036)
(17), $p = 100$	0.14 (0.015)	0.52 (0.043)	0.14 (0.015)	0.48 (0.045)
(17), $p = 1000$	0.77 (0.070)	1.39 (0.102)	0.55 (0.064)	1.27 (0.105)
(18), $p = 50$	0.21 (0.024)	0.31 (0.027)	0.21 (0.024)	0.30 (0.026)
(18), $p = 100$	0.22 (0.024)	0.39 (0.028)	0.22 (0.024)	0.39 (0.028)
(18), $p = 1000$	0.45 (0.035)	0.97 (0.052)	0.38 (0.030)	0.72 (0.049)

Table 1: Mean squared error (MSE), $\mathbb{E}[(\hat{f}(X) - f(X))^2]$ ($f(x) = \mathbb{E}[Y|X = x]$), in model (16) for gMDL-SparseL_2Boost and gMDL early stopped L_2Boosting using the estimated stopping iteration $\hat{m}$. The performance using the oracle m which minimizes MSE is denoted by an asterisk *. Estimated standard errors are given in parentheses. Sample size is $n = 50$.

number of ineffective variables from 46 ($p = 50$) to 96 ($p = 100$). However, with very many, that is 996 ($p = 1000$), ineffective variables, a significant loss in accuracy shows up in the orthogonal design (17) and there is an indication that the relative differences between SparseL_2Boost and L_2Boosting become smaller. For the positive dependent design in (18), the loss in accuracy in the $p = 1000$ case is not as significant as in the orthogonal design case in (17), and the relative differences between SparseL_2Boost and L_2Boosting actually become larger.

It is also worth pointing out that the resulting mean squared errors (MSEs) in design (17) and (18) are not really comparable even for the same number p of predictors. This is because, even though the noise level is $\mathbb{E}|\varepsilon|^2 = 1$ for both designs, the signal levels $\mathbb{E}|f(X)|^2$ are different, that is

31 for the uncorrelated design in (17) and 49.5 for the correlated design in (18). If we would like to compare the performances among the two designs, we should rather look at the signal-adjusted mean squared error

$$\frac{\mathbb{E}|\hat{f}(X) - f(X)|^2}{\mathbb{E}|f(X)|^2}$$

which is the test-set analogue of $1 - R^2$ in linear models. This signal adjusted error measure can be computed from the results in Table 1 and the signal levels given above. We then obtain for the lower dimensional cases with $p \in \{50, 100\}$ that the prediction accuracies are about the same for the correlated and the uncorrelated design (for SparseL_2Boost and for L_2Boosting). However, for the high-dimensional case with $p = 1000$, the performance (of SparseL_2Boost and of L_2Boosting) is significantly better in the correlated than the uncorrelated design.

Next, we consider the ability of selecting the correct variables: the results are given in Table 2.

Σ , dim.		SparseL_2Boost	L_2Boosting
(17), $p = 50$:	ℓ^0-norm	5.00 (0.125)	13.68 (0.438)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	1.00 (0.125)	9.68 (0.438)
(17), $p = 100$:	ℓ^0-norm	5.78 (0.211)	21.20 (0.811)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	1.78 (0.211)	17.20 (0.811)
(17), $p = 1000$:	ℓ^0-norm	23.70 (0.704)	78.80 (0.628)
	non-selected T	0.02 (0.020)	0.02 (0.020)
	selected F	19.72 (0.706)	74.82 (0.630)
(18), $p = 50$:	ℓ^0-norm	4.98 (0.129)	9.12 (0.356)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	0.98 (0.129)	5.12 (0.356)
(18), $p = 100$:	ℓ^0-norm	5.50 (0.170)	12.44 (0.398)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	1.50 (0.170)	8.44 (0.398)
(18), $p = 1000$:	ℓ^0-norm	13.08 (0.517)	71.68 (1.018)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	9.08 (0.517)	67.68 (1.018)

Table 2: Model (16): expected number of selected variables (ℓ^0-norm), expected number of non-selected true effective variables (non-selected T) which is in the range of $[0,4]$, and expected number of selected non-effective (false) variables (selected F) which is in the range of $[0, p-4]$. Methods: SparseL_2Boost and L_2Boosting using the estimated stopping iteration $\hat{m}$ (Step 5 in the SparseL_2Boost algorithm and (15) respectively). Estimated standard errors are given in parentheses. Sample size is $n = 50$.

In the orthogonal case, we have argued that SparseL_2Boost has a tendency for sparser results than L_2Boosting; see the discussion of different threshold functions in section 2.4. This is confirmed in

all our numerical experiments. In particular, for our ℓ^0-sparse model (16), the detailed results are reported in Table 2. SparseL_2Boost selects much fewer predictors than L_2Boosting. Moreover, for this model, SparseL_2Boost is a good model selector as long as the dimensionality is not very large, that is for $p \in \{50, 100\}$, while L_2Boosting is much worse selecting too many false predictors (that is too many false positives). For the very high-dimensional case with $p = 1000$, the selected models are clearly too large when compared with the true model size, even when using SparseL_2Boost. However, the results are pretty good considering the fact that we are dealing with a much harder problem of getting rid of 996 irrelevant predictors based on only 50 sample points. To summarize, for this synthetic example, SparseL_2Boost works significantly better than L_2Boosting both in terms of MSE, model selection and sparsity, due to the sparsity of the true model.

3.1.2 A NON-SPARSE MODEL WITH RESPECT TO THE ℓ^0-NORM

We provide here an example where L_2Boosting will be better than SparseL_2Boost. Consider the model

$$Y = \sum_{j=1}^{p} \frac{1}{5}\beta_j X_j + \varepsilon,$$
$$X_1, \ldots, X_p \sim \mathcal{N}_p(0, I_p),\ \varepsilon \sim \mathcal{N}(0,1), \tag{19}$$

where $\beta_1, \ldots, \beta_p$ are fixed values from i.i.d. realizations of the double-exponential density $p(x) = \exp(-|x|)/2$. The magnitude of the coefficients $|\beta_j|/5$ is chosen to vary the signal to noise ratio from model (16), making it about 5 times smaller than for (19). Since Lasso (coinciding with L_2Boosting in the orthogonal case) is the maximum a-posteriori (MAP) method when the coefficients are from a double-exponential distribution and the observations from a Gaussian distribution, as in (19), we expect L_2Boosting to be better than SparseL_2Boost for this example (even though we understand that MAP is not the Bayesian estimator under the L^2 loss). The squared error performance is given in Table 3, supporting our expectations. SparseL_2Boost nevertheless still has the virtue of sparsity with only about 1/3 of the number of selected predictors but with an MSE which is larger by a factor 1.7 when compared with L_2Boosting.

	SparseL_2Boost	L_2Boosting	SparseL_2Boost*	L_2Boosting*
MSE	3.64 (0.188)	2.19 (0.083)	3.61 (0.189)	2.08 (0.078)
ℓ^0-norm	11.78 (0.524)	29.16 (0.676)	11.14 (0.434)	35.76 (0.382)

Table 3: Mean squared error (MSE) and expected number of selected variables (ℓ^0-norm) in model (19) with $p = 50$. Estimated standard errors are given in parentheses. All other specifications are described in the caption of Table 1.

3.1.3 DATA-DRIVEN CHOICE BETWEEN SPARSEL_2BOOST AND L_2BOOSTING: GMDL-SEL-L_2BOOST

We illustrate here the gMDL-sel-L_2Boost proposal from section 2.5 that uses the gMDL model selection score to choose in a data-driven way between SparseL_2Boost and L_2Boosting. As an

illustration, we consider again the models in (16)-(17) and (19) with $p = 50$ and $n = 50$. Figure 2 displays the results in the form of boxplots across 50 rounds of simulations.

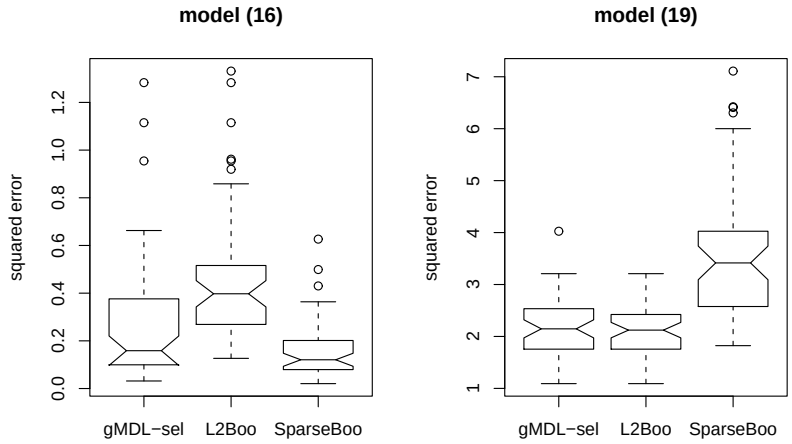


Figure 2: Out-of-sample squared error losses, $\text{ave}_X[(\hat{f}(X) - f(X))^2]$ ($f(x) = \mathbb{E}[Y|X = x]$), from the 50 simulations for the models in (16)-(17) and (19) with $p = 50$. gMDL-sel-L_2Boost (gMDL-sel), L_2Boosting (L2Boo) and SparseL_2Boost (SparseBoo). Sample size is $n = 50$.

The gMDL-sel-L_2Boost method performs between the better and the worse of the two boosting algorithms, but closer to the better performer in each situation (the latter is only known for simulated data sets). For model (19), there is essentially no degraded performance when doing a data-driven selection between the two boosting algorithms (in comparison to the best performer).

3.2 Ozone Data with Interactions Terms

We consider a real data set about ozone concentration in the Los Angeles basin. There are $p = 8$ meteorological predictors and a real-valued response about daily ozone concentration; see Breiman (1996). We constructed second-order interaction and quadratic terms after having centered the original predictors. We then obtain a model with $p = 45$ predictors (including an intercept) and a response. We used 10-fold cross-validation to estimate the out-of-sample squared prediction error and the average number of selected predictor variables. When scaling the predictor variables (and their interactions) to zero mean and variance one, the performances were very similar. Our results are comparable to the analysis of bagging in Breiman (1996) which yielded a cross-validated squared error of 18.8 for bagging trees based on the original eight predictors.

We also run SparseL_2Boost and L_2Boosting on the whole data set and choose the method according to the better gMDL-score, that is gMDL-sel-L_2Boost (see section 2.5). Some results are given in Table 5. Based on SparseL_2Boost, an estimate for the error variance is $n^{-1}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = 15.56$

	SparseL_2Boost	L_2Boosting
10-fold CV squared error	16.52	16.57
10-fold CV ℓ^0-norm	10.20	16.10

Table 4: Boosting with componentwise linear least squares for ozone data with first order-interactions ($n = 330,\ p = 45$). Squared prediction error and average number of selected predictor variables using 10-fold cross-validation.

and the goodness of fit equals $R^2 = \sum_{i=1}^n (\hat{Y}_i - \overline{Y})^2 / \sum_{i=1}^n (Y_i - \overline{Y})^2 = 0.71$, where $\hat{Y}_i = \hat{F}(X_i)$ and $\overline{Y} = n^{-1} \sum_{i=1}^n Y_i$.

	SparseL_2Boost (#)	L_2Boosting
gMDL-score	2.853	2.862
RSS	15.56	15.24
ℓ^0-norm	10	18

Table 5: Boosting with componentwise linear least squares for ozone data with first order-interactions ($n = 330,\ p = 45$). gMDL-score, $n^{-1}\times$ residual sum of squares (RSS) and number of selected terms (ℓ^0-norm). (#) gMDL-sel-L_2Boost selects SparseL_2Boost as the better method.

In summary, while SparseL_2Boost is about as good as L_2Boosting in terms of predictive accuracy, see Table 4, it yields a sparser model fit, see Tables 4 and 5.

3.3 Binary Tumor Classification Using Gene Expressions

We consider a real data set which contains $p = 7129$ gene expressions in 49 breast tumor samples using the Affymetrix technology, see West et al. (2001). After thresholding to a floor of 100 and a ceiling of 16,000 expression units, we applied a base 10 log-transformation and standardized each experiment to zero mean and unit variance. For each sample, a binary response variable $Y \in \{0,1\}$ is available, describing the status of lymph node involvement in breast cancer. The data are available at `http://mgm.duke.edu/genome/dna_micro/work/`.

Although the data has the structure of a binary classification problem, the squared error loss is quite often employed for estimation. We use L_2Boosting and SparseL_2Boost with componentwise linear least squares. We classify the label 1 if $\hat{p}(x) = \hat{\mathbb{P}}[Y+1|X=x] > 1/2$ and zero otherwise. The estimate for $\hat{p}(\cdot)$ is obtained as follows:

$$\begin{aligned} &\hat{p}_m(\cdot) = 1/2 + \hat{F}_m(\cdot), \\ &\hat{F}_m(\cdot) \text{ the } L_2\text{- or Sparse}L_2\text{Boost estimate using } \tilde{Y} = Y - 1/2. \end{aligned} \tag{20}$$

Note that $\hat{F}_m(\cdot)$ is an estimate of $p(\cdot) - 1/2$. Using this procedure amounts to modelling and estimating the deviation from the boundary value 1/2 (we do not use an intercept term anymore in our model). This is usually much better because the L_2- or SparseL_2Boost estimate is shrunken towards

zero. When using L_2- or SparseL_2Boost on $Y \in \{0,1\}$ directly, with an intercept term, we would obtain a shrunken boosting estimate of the intercept introducing a bias rendering $\hat{p}(\cdot)$ to be systematically too small. The latter approach has been used in Bühlmann (2006) yielding worse results for L_2Boosting than what we report here for L_2Boosting using (20).

Since the gMDL criterion is relatively new, its classification counterpart is not yet well developed (see Hansen and Yu, 2002). Instead of the gMLD criterion in (14) and (15), we use the BIC score for the Bernoulli-likelihood in a binary classification:

$$BIC(m) = -2 \cdot \text{log-likelihood} + \log(n) \cdot \text{trace}(\mathcal{B}_m).$$

The AIC criterion would be another option: it yields similar, a bit less sparse results for our tumor classification problem.

We estimate the classification performance by a cross-validation scheme where we randomly divide the 49 samples into balanced training- and test-data of sizes $2n/3$ and $n/3$, respectively, and we repeat this 50 times. We also report on the average of selected predictor variables. The reports are given in Table 6.

	SparseL_2Boost	L_2Boosting
CV misclassification error	21.88%	23.13%
CV ℓ^0-norm	12.90	15.30

Table 6: Boosting with componentwise linear least squares for tumor classification data ($n = 46,\ p = 7129$). Misclassification error and average number of selected predictor variables using cross-validation (with random 2/3 training and 1/3 test sets).

The predictive performance of L_2- and SparseL_2Boosting compares favourably with four other methods, namely 1-nearest neighbors, diagonal linear discriminant analysis, support vector machine with radial basis kernel (from the R-package `e1071` and using its default values), and a forward selection penalized logistic regression model (using some reasonable penalty parameter and number of selected genes). For 1-nearest neighbors, diagonal linear discriminant analysis and support vector machine, we pre-select the 200 genes which have the best Wilcoxon score in a two-sample problem (estimated from the training data set only), which is recommended to improve the classification performance. Forward selection penalized logistic regression is run without pre-selection of genes. The results are given in Table 5 which is taken from Bühlmann (2006).

	FPLR	1-NN	DLDA	SVM
CV misclassification error	35.25%	43.25%	36.12%	36.88%

Table 7: Cross-validated misclassification rates for lymph node breast cancer data. Forward variable selection penalized logistic regression (FPLR), 1-nearest-neighbor rule (1-NN), diagonal linear discriminant analysis (DLDA) and a support vector machine (SVM)

When using SparseL_2Boost and L_2Boosting on the whole data set, we get the following results displayed in Table 8. The 12 variables (genes) which are selected by SparseL_2Boost are a subset

of the 14 selected variables (genes) from L_2Boosting. Analogously as in section 3.2, we give some ANOVA-type numbers of SparseL_2Boosting: the error variability is $n^{-1}\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = 0.052$ and the goodness of fit equals $R^2 = \sum_{i=1}^n (\hat{Y}_i - \overline{Y})^2 / \sum_{i=1}^n (Y_i - \overline{Y})^2 = 0.57$, where $\hat{Y}_i = \hat{F}(X_i)$ and $\overline{Y} = n^{-1}\sum_{i=1}^n Y_i$.

	SparseL_2Boost (#)	L_2Boosting
BIC score	35.09	37.19
RSS	0.052	0.061
ℓ^0-norm	12	14

Table 8: Boosting with componentwise linear least squares for tumor classification ($n = 49,\ p = 7129$). BIC score, $n^{-1}\times$ residual sum of squares (RSS) and number of selected terms (ℓ^0-norm). (#) BIC-sel-L_2Boost selects SparseL_2Boost as the better method.

In summary, the predictive performance of SparseL_2Boost is slightly better than of L_2Boosting, see Table 6, and SparseL_2Boost selects a bit fewer variables (genes) than L_2Boosting, see Tables 7 and 8.

3.4 Nonparametric Function Estimation with Second-Order Interactions

Consider the Friedman #1 model Friedman (1991),

$$\begin{aligned} &Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 + 10X_4 + 5X_5 + \varepsilon, \\ &X \sim \text{Unif.}([0,1]^p),\ \varepsilon \sim \mathcal{N}(0,1), \end{aligned} \tag{21}$$

where ε is independent from X. The sample size is chosen as $n = 50$ and the predictor dimension is $p \in \{10, 20\}$ which is still large relative to n for a nonparametric problem.

SparseL_2Boost and L_2Boosting with a pairwise thin plate spline, which selects the best pair of predictor variables yielding lowest residual sum of squares (when having the same degrees of freedom `d.f.` $= 5$ for every thin plate spline), yields a second-order interaction model; see also section 2.1. We demonstrate in Table 9 the effectiveness of these procedures, also in comparison with the MARS Friedman (1991) fit constrained to second-order interaction terms. SparseL_2Boost is a bit better than L_2Boosting. But the estimation of the boosting iterations by gMDL did not do as well as in section 3.1 since the oracle methods perform significantly better. The reason is that this example has a high signal to noise ratio. From (Hansen and Yu, 1999), the F in the gMDL penalty (see (14)) is related to the signal to noise ratio (SNR). Thus, when SNR is high, the $\log(F)$ is high too, leading to too small models in both SparseL_2Boost and L_2Boosting: that is, this large penalty forces both SparseL_2Boost and L_2Boosting to stop too early in comparison to the oracle stopping iteration which minimizes MSE. However, both boosting methods nevertheless are quite a bit better than MARS.

When increasing the noise level, using $\text{Var}(\varepsilon) = 16$, we obtain the following MSEs for $p = 10$: 11.70 for SparseL_2Boost, 11.65 for SparseL_2Boost* with the oracle stopping rule and 24.11 for MARS. Thus, for lower signal to noise ratios, stopping the boosting iterations with the gMDL criterion works very well, and our SparseL_2Boost algorithm is much better than MARS.

dim.	SparseL_2Boost	L_2Boosting	MARS	SparseL_2Boost*	L_2Boosting*
$p = 10$	3.71 (0.241)	4.10 (0.239)	5.79 (0.538)	2.22 (0.220)	2.69 (0.185)
$p = 20$	4.36 (0.238)	4.81 (0.197)	5.82 (0.527)	2.68 (0.240)	3.56 (0.159)

Table 9: Mean squared error (MSE) in model (21). All other specifications are described in the caption of Table 1, except for MARS which is constrained to second-order interaction terms.

4. Conclusions

We propose SparseL_2Boost, a gradient descent algorithm on a penalized squared error loss which yields sparser solutions than L_2Boosting or ℓ^1-regularized versions thereof. The new method is mainly useful for high-dimensional problems with many ineffective predictor variables (noise variables). Moreover, it is computationally feasible in high dimensions, for example having linear complexity in the number of predictor variables p when using componentwise linear least squares or componentwise smoothing splines (see section 2.1).

SparseL_2Boost is essentially as generic as L_2Boosting and can be used in connection with nonparametric base procedures (weak learners). The idea of sparse boosting could also be transferred to boosting algorithms with other loss functions, leading to sparser variants of AdaBoost and LogitBoost.

There is no general superiority of sparse boosting over boosting, even though we did find in four out of our five examples (two real data and two synthetic data sets) that SparseL_2Boost outperforms L_2Boosting in terms of sparsity and SparseL_2Boost is as good or better than L_2Boosting in terms of predictive performance. In the synthetic data example in section 3.1.2, chosen to be the ideal situation for L_2Boosting, SparseL_2Boost loses 70% in terms of MSE, but uses only 1/3 of the predictors. Hence if one cares about sparsity, SparseL_2Boost seems a better choice than L_2Boosting. In our framework, the boosting approach automatically comes with a reasonable notion for statistical complexity or degrees of freedom, namely the trace of the boosting operator when it can be expressed in hat matrix form. This trace complexity is well defined for many popular base procedures (weak learners) including componentwise linear least squares and decision trees, see also section 2.1. SparseL_2Boost gives rise to a direct, fast computable estimate of the out-of-sample error when combined with the gMDL model selection criterion (and thus, by-passing cross-validation). This out-of-sample error estimate can also be used for choosing the stopping iteration in L_2Boosting and for selecting between sparse and traditional boosting, resulting in the gMDL-sel-L_2Boost algorithm.

Theoretical results in the orthogonal linear regression model as well as simulation and data experiments are provided to demonstrate that the SparseL_2Boost indeed gives sparser model fits than L_2Boosting and that gMDL-sel-L_2Boost automatically chooses between the two to give a rather satisfactory performance in terms of sparsity and prediction.

5. Proofs

We first give the proof of Theorem 2. It then serves as a basis for proving Theorem 1.

Proof of Theorem 2. We represent the componentwise linear least squares base procedure as a hat operator $\mathcal{H}_{\hat{s}}$ with $\mathcal{H}_j = \mathbf{x}^{(j)}(\mathbf{x}^{(j)})^T$, where $\mathbf{x}^{(j)} = (x_1^{(j)}, \ldots, x_n^{(j)})^T$; see also section 2.1. The L_2Boosting operator in iteration m is then given by the matrix

$$\mathcal{B}_m = I - (I - \nu\mathcal{H}_1)^{m_1}(I - \nu\mathcal{H}_2)^{m_2} \cdots (I - \nu\mathcal{H}_n)^{m_n},$$

where m_i equals the number of times that the ith predictor variable has been selected during the m boosting iterations; and hence $m = \sum_{i=1}^n m_i$. The derivation of the formula above is straightforward because of the orthogonality of the predictors $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$ which implies the commutation $\mathcal{H}_j\mathcal{H}_k = \mathcal{H}_k\mathcal{H}_j$. Moreover, $\mathcal{B}_m$ can be diagonalized

$$\mathcal{B}_m = \mathbf{X}D_m\mathbf{X}^T \text{ with } \mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = I,\ D_m = \text{diag}(d_{m,1}, \ldots, d_{m,n}),\ d_{m,i} = 1 - (1-\nu)^{m_i}.$$

Therefore, the residual sum of squares in the mth boosting iteration is:

$$RSS_m = \|\mathbf{Y} - \mathcal{B}_m\mathbf{Y}\|^2 = \|\mathbf{X}^T\mathbf{Y} - \mathbf{X}^T\mathcal{B}_m\mathbf{Y}\|^2 = \|Z - D_mZ\|^2 = \|(I - D_m)Z\|^2,$$

where $Z = \mathbf{X}^T\mathbf{Y}$.

The RSS_m decreases monotonically in m. Moreover, the amount of decrease $RSS_m - RSS_{m+1}$ is decaying monotonously in m, because L_2Boosting proceeds to decrease the RSS as much as possible in every step (by selecting the most reducing predictor $\mathbf{x}^{(j)}$) and due to the structure of $(1 - d_{m,i}) = (1-\nu)^{m_i}$. Thus, every stopping of boosting with an iteration number m corresponds to a tolerance δ^2 such that

$$\begin{aligned} &RSS_k - RSS_{k+1} > \delta^2,\ k = 1, 2, ..., m-1, \\ &RSS_m - RSS_{m+1} \le \delta^2, \end{aligned} \tag{22}$$

that is, the iteration number m corresponds to a numerical tolerance where the difference $RSS_m - RSS_{m+1}$ is smaller than δ^2.

Since L_2Boosting changes only one of the summands in RSS_m in the boosting iteration $m+1$, the criterion in (22) implies that for all $i \in \{1, \ldots, n\}$

$$\begin{aligned} &((1-\nu)^{2(m_i-1)} - (1-\nu)^{2m_i})Z_i^2 > \delta^2, \\ &((1-\nu)^{2m_i} - (1-\nu)^{2(m_i+1)})Z_i^2 \le \delta^2. \end{aligned} \tag{23}$$

If $m_i = 0$, only the second line in the above expression is relevant. The L_2Boosting solution with tolerance δ^2 is thus characterized by (23).

Let us first, for the sake of insight, replace the "$\le$" in (23) by "$\approx$": we will deal later in which sense such an approximate equality holds. If $m_i \ge 1$, we get

$$((1-\nu)^{2m_i} - (1-\nu)^{2(m_i+1)})Z_i^2 = (1-\nu)^{2m_i}(1 - (1-\nu)^2)Z_i^2 \approx \delta^2,$$

and hence

$$(1-\nu)^{m_i} \approx \frac{\delta}{\sqrt{1 - (1-\nu)^2}|Z_i|}. \tag{24}$$

In case where $m_i = 0$, we obviously have that $1-(1-\nu)^{m_i} = 0$. Therefore,

$$\hat{\beta}^{(m)}_{Boost,i} = \hat{Z}_i = d_{m,i} = (1-(1-\nu)^{m_i})Z_i \approx Z_i - \frac{\delta}{\sqrt{1-(1-\nu)^2}|Z_i|}Z_i \quad \text{if } m_1 \geq 1,$$
$$\hat{\beta}^{(m)}_{Boost,i} = 0 \quad \text{if } m_i = 0.$$

Since $m_i = 0$ happens only if $|Z_i| \leq \frac{\delta}{\sqrt{1-(1-\nu)^2}}$, we can write the estimator as

$$\hat{\beta}^{(m)}_{Boost,i} \approx \begin{cases} Z_i - \lambda, & \text{if } Z_i \geq \lambda, \\ 0, & \text{if } |Z_i| < \lambda, \\ Z_i + \lambda, & \text{if } Z_i \leq -\lambda. \end{cases} \tag{25}$$

where $\lambda = \frac{\delta}{\sqrt{1-(1-\nu)^2}}$ (note that m is connected to δ, and hence to λ via the criterion in (22)). This is the soft-threshold estimator with threshold λ, as in (13). By choosing $\delta = \lambda_n\sqrt{1-(1-\nu)^2}$, we get the desired threshold λ_n.

We will now deal with the approximation in (24). By the choice of δ two lines above, we would like that

$$(1-\nu)^{m_i} \approx \lambda_n/|Z_i|.$$

As we will see, this approximation is accurate when choosing ν small. We only have to deal with the case where $|Z_i| > \lambda_n$; if $|Z_i| \leq \lambda_n$, we know that $m_i = 0$ and $\hat{\beta}_i = 0$ exactly, as claimed in the right hand side of (25). Denote by

$$V_i = V(Z_i) = \frac{\lambda_n}{|Z_i|} \in (0,1).$$

(The range $(0,1)$ holds for the case we are considering here). According to the stopping criterion in (23), the derivation as for (24) and the choice of δ, this says that

$$\begin{aligned} (1-\nu)^{m_i} &> V_i, \\ (1-\nu)^{m_i+1} &\leq V_i, \end{aligned} \tag{26}$$

and hence

$$\begin{aligned} \Delta(\nu, V_i) &\stackrel{\text{def}}{=} ((1-\nu)^{m_i} - V_i) \leq ((1-\nu)^{m_i} - (1-\nu)^{m_i+1}) \\ &= \frac{\nu}{1-\nu}(1-\nu)^{m_i+1} \leq \frac{\nu}{1-\nu}V_i, \end{aligned}$$

by using (26). Thus,

$$\begin{aligned} &(1-\nu)^{m_i} = V_i + ((1-\nu)^{m_i} - V_i) = V_i(1+\Delta(\nu,V_i)/V_i) = V_i(1+e_i(\nu)), \\ &|e_i(\nu)| = |\Delta(\nu,V_i)/V_i| \leq \nu/(1-\nu). \end{aligned} \tag{27}$$

Thus, when multiplying with $(-1)Z_i$ and adding Z_i,

$$\begin{aligned} \hat{\beta}^{(m)}_{Boost,i} &= (1-(1-\nu)^{m_i})Z_i = Z_i - Z_iV_i(1+e_i(\nu)) \\ &= \text{soft-threshold estimator with threshold } \lambda_n(1+e_i(\nu)), \end{aligned}$$

where $\max_{1\le i\le n}|e_i(\nu)| \le \nu/(1-\nu)$ as in (27). □

Proof of Theorem 1. The proof is based on similar ideas as for Theorem 2. The SparseL_2Boost in iteration m aims to minimize

$$MSB_m = RSS_m + \gamma_n \text{trace}(\mathcal{B}_m) = \|\mathbf{Y} - \mathbf{X}\hat{\beta}^{(m)}_{ms-boost}\|^2 + \gamma_n \text{trace}(\mathcal{B}_m).$$

When using the orthogonal transformation by multiplying with $\mathbf{X}^T$, the criterion above becomes

$$MSB_m = \|Z - \hat{\beta}^{(m)}_{ms-boost}\|^2 + \gamma_n \text{trace}(\mathcal{B}_m),$$

where $\text{trace}(\mathcal{B}_m) = \sum_{i=1}^n (1-(1-\nu)^{m_i})$. Moreover, we run SparseL_2Boost until the stopping iteration m satisfies the following:

$$\begin{aligned} &MSB_k - MSB_{k+1} > 0,\ k = 1,2,\ldots,m-1, \\ &MSB_m - MSB_{m+1} \le 0. \end{aligned} \tag{28}$$

It is straightforward to see for the orthonormal case, that such an m coincides with the definition for $\hat{m}$ in section 2.3. Since SparseL_2Boost changes only one of the summands in *RSS* and the trace of $\mathcal{B}_m$, the criterion above implies that for all $i = 1,\ldots,n$, using the definition of *MSB*,

$$\begin{aligned} &(1-\nu)^{2(m_i-1)} Z_i^2 (1-(1-\nu)^2) - \gamma_n \nu (1-\nu)^{m_i-1} > 0, \\ &(1-\nu)^{2m_i} Z_i^2 (1-(1-\nu)^2) - \gamma_n \nu (1-\nu)^{m_i} \le 0. \end{aligned} \tag{29}$$

Note that if $|Z_i|^2 \le \gamma_n \nu/(1-(1-\nu)^2)$, then $m_i = 0$. This also implies uniqueness of the iteration m such that (28) holds or of the m_i such that (29) holds.

Similarly to the proof of Theorem 2, we look at this expression first in terms of an approximate equality to zero, that is ≈ 0. We then immediately find that

$$(1-\nu)^{m_i} \approx \frac{\gamma_n \nu}{(1-(1-\nu)^2)|Z_i|^2}.$$

Hence,

$$\begin{aligned} \hat{\beta}^{(m)}_{ms-boost,i} &= (\mathbf{X}^T \mathcal{B}_m \mathbf{Y})_i = (\mathbf{X}^T \mathbf{X} D_m \mathbf{X}^T \mathbf{Y})_i = (D_m Z)_i = (1-(1-\nu)^{m_i}) Z_i \\ &\approx Z_i - \frac{\gamma_n \nu Z_i}{(1-(1-\nu)^2)|Z_i|^2} = Z_i - sign(Z_i) \frac{\gamma_n}{2-\nu} \frac{1}{|Z_i|}. \end{aligned}$$

The right-hand side is the nonnegative garrote estimator as in (12) with threshold $\gamma_n/(2-\nu)$.

Dealing with the approximation "≈" can be done similarly as in the proof of Theorem 2. We define here

$$V_i = V(Z_i) = \frac{\gamma_n \nu}{(1-(1-\nu)^2)|Z_i|^2}.$$

We then define $\Delta(\nu, V_i)$ and $e_i(\nu)$ as in the proof of Theorem 2, and we complete the proof as for Theorem 2. □

Acknowledgments

B. Yu would like to acknowledge gratefully the partial supports from NSF grants FD01-12731 and CCR-0106656 and ARO grants DAAD19-01-1-0643 and W911NF-05-1-0104, and the Miller Research Professorship in Spring 2004 from the Miller Institute at University of California at Berkeley. Both authors thank David Mease, Leo Breiman, two anonymous referees and the action editors for their helpful comments and discussions on the paper.

References

H. Akaike. Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203, 1970.

L. Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37:373–384, 1995.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Arcing classifiers (with discussion). *Ann. Statist.*, 26:801–849, 1998.

L. Breiman. Prediction games & arcing algorithms. *Neural Computation*, 11:1493–1517, 1999.

P. Bühlmann. Boosting for high-dimensional linear models. *To appear in Ann. Statist.*, 34, 2006.

P. Bühlmann and B. Yu. Boosting with the $l_2$loss: regression and classification. *J. Amer. Statist. Assoc.*, 98:324–339, 2003.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *Ann. Statist.*, 32:407–451, 2004.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proc. Thirteenth Intern. Conf.*, pages 148–156. Morgan Kauffman, 1996.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Ann.Statist.*, 29: 1189–1232, 2001.

J. H. Friedman. Multivariate adaptive regression splines (with discussion). *Ann.Statist.*, 19:1–141, 1991.

J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *Ann. Statist.*, 28:337–407, 2000.

P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman and Hall, 1994.

M. Hansen and B. Yu. Model selection and minimum description length principle. *J. Amer. Statist. Assoc.*, 96:746–774, 2001.

M. Hansen and B. Yu. *Minimum Description Length Model Selection Criteria for Generalized Linear Models*. IMS Lecture Notes – Monograph Series, Vol. 40, 2002.

M. Hansen and B. Yu. Bridging aic and bic: an mdl model selection criterion. In *IEEE Information Theory Workshop on Detection, Imaging and Estimation; Santa Fe*, 1999.

G. Lugosi and N. Vayatis. On the bayes-risk consistency of regularized boosting methods (with discussion). *Ann. Statist.*, 32:30–55 (disc. pp. 85–134), 2004.

S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Proc.*, 41:3397–3415, 1993.

N. Meinshausen. Lasso with relaxation. Technical report, 2005.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320., 2001.

G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48:193–221, 2002.

T. Speed and B. Yu. Model selection and prediction: normal regression. *Ann. Inst. Statist. Math.*, 45:35–54, 1993.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc., Ser. B*, 58: 267–288, 1996.

J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. Olson, J. Marks, and J. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proc. Nat. Acad. Sci. (USA)*, 98:11462–11467, 2001.

Journal of Machine Learning Research 7 (2006) 1025 – 1044 Submitted 5/05; Revised 1/06; Published 6/06

One-Class Novelty Detection for Seizure Analysis from Intracranial EEG

Andrew B. Gardner AGARDNER@NEUROENG.ORG
Departments of Bioengineering and Neurology
University of Pennsylvania
301 Hayden Hall
Philadelphia, PA 19104, USA

Abba M. Krieger KRIEGER@WHARTON.UPENN.EDU
Department of Statistics, Wharton School
University of Pennsylvania
3733 Spruce Street
Philadelphia, PA 19104, USA

George Vachtsevanos GJV@ECE.GATECH.EDU
Department of Electrical and Computer Engineering
Georgia Institute of Technology
777 Atlantic Drive
Atlanta, GA 30332, USA

Brian Litt LITTB@MAIL.MED.UPENN.EDU
Departments of Bioengineering and Neurology
Hospital of the University of Pennsylvania
3 West Gates, 3400 Spruce Street
Philadelphia, PA 19104, USA

Editor: Leslie Pack Kaelbing

Abstract

This paper describes an application of one-class support vector machine (SVM) novelty detection for detecting seizures in humans. Our technique maps intracranial electroencephalogram (EEG) time series into corresponding novelty sequences by classifying short-time, energy-based statistics computed from one-second windows of data. We train a classifier on epochs of interictal (normal) EEG. During ictal (seizure) epochs of EEG, seizure activity induces distributional changes in feature space that increase the empirical outlier fraction. A hypothesis test determines when the parameter change differs significantly from its nominal value, signaling a seizure detection event. Outputs are gated in a "one-shot" manner using persistence to reduce the false alarm rate of the system. The detector was validated using leave-one-out cross-validation (LOO-CV) on a sample of 41 interictal and 29 ictal epochs, and achieved 97.1% sensitivity, a mean detection latency of

-7.58 seconds, and an asymptotic false positive rate (FPR) of 1.56 false positives per hour (Fp/hr). These results are better than those obtained from a novelty detection technique based on Mahalanobis distance outlier detection, and comparable to the performance of a supervised learning technique used in experimental implantable devices (Echauz et al., 2001). The novelty detection paradigm overcomes three significant limitations of competing methods: the need to collect seizure data, precisely mark seizure onset and offset times, and perform patient-specific parameter tuning for detector training.

Keywords: seizure detection, novelty detection, one-class SVM, epilepsy, unsupervised learning

1 Introduction

Epilepsy, a neurological disorder in which patients suffer from recurring seizures, affects approximately 1% of the world population. In the United States, 200,000 new cases are reported annually. There are more than 30 distinct classes of seizure. Their manifestations range from subtle, abnormal sensations to unpredictable changes in awareness, to immediate loss of consciousness and convulsions. In spite of available dietary, drug, and surgical treatment options, more than 25% of individuals with epilepsy have seizures that are uncontrollable (Kandel, Schwartz & Jessel, 1991). Daily life for these patients is greatly impaired—education, employment, and even transportation can become difficult endeavors. Many new therapies for medically resistant epilepsy are being investigated. Among the most promising are implantable devices that deliver local therapy, such as direct electrical stimulation or chemical infusions, to affected regions of the brain. These treatments rely on robust algorithms for seizure detection to perform effectively.

Over the past 30 years seizure detection technology has matured. Despite impressive advances, all reported approaches suffer from one or more of the following limitations:

- Accurate detection requires careful, patient-specific tuning
- Seizure detections do not occur "early enough" (i.e., interventions are more likely to be effective if therapy is administered with minimal delay following onset)
- *A priori* localization of the seizure focus is required
- Usefulness for poorly localized epilepsies is limited
- Seizure data (which is expensive to collect) is required for training

Techniques for overcoming some or all of these limitations hold promise for more precise and widely applicable methods to control or eliminate seizures. This paper presents one technique for improving the state of the art in seizure detection by reformulating the task as a time-series novelty detection problem. While seizure detection is traditionally considered a supervised learning problem (e.g., binary classification), an unsupervised approach allows for uniform treatment of seizure detection and prediction, and offers four key advantages for implementation. First, there is no need to perform supervised, patient-specific tuning during training. Second, the assumption that seizures are electrographically homogeneous—often required for classifier training due to very small data sets—is relaxed. Third, there is no need to collect seizure data for training. Such data collection is typically expensive (seizures occur infrequently, and patients must be continually monitored until an event is observed), and often invasive (e.g., craniotomy or burr hole for electrode implantation). Finally, there is no need to precisely mark seizure intervals. This practical issue is often overlooked, but is critical for training and validation: while expert markers usually agree on the presence of a seizure, there is considerable variability in marking its onset and offset.

Other researchers have investigated novelty detection for event detection from time series, for instance by directly extending the Incremental SVM algorithm (Tax & Laskov, 2003), or

modeling novelty region-of-support evolution to detect change-points (Desobry & Davy, 2003). In contrast to these approaches, we robustly detect empirical changes in the novelty parameter itself, and use these change-points to segment the (EEG) time series. For "properly chosen" features, novelties correspond well with the ictal events of interest, and our EEG time series are successfully segmented in a one-class-from-many manner.

2 Background

In this section, we present a brief review of seizure-related terminology, the seizure detection literature, and the one-class SVM.

2.1 Seizure-Related Terminology

Seizure analysis refers collectively to algorithms for seizure detection, seizure prediction, and automatic focus channel identification. These analyses are primarily performed on the EEG. In this study, analyses were carried out on the intracranial EEG (IEEG), which has considerably better spatial resolution, higher signal-to-noise ratio, and greater bandwidth than scalp EEG. When multiple channels are considered, the electrode location that exhibits the earliest evidence of seizure activity is labeled the *focus channel*. It is convenient to describe segments of the EEG signals by their temporal proximity to seizure activity. The *ictal* period refers to the time during which a seizure occurs. The *interictal* period is the time between successive seizures. The *unequivocal electrographic onset (UEO)* is defined as the earliest time that a seizure occurrence is evident to an epileptologist viewing an EEG without prior knowledge that a seizure follows; the *unequivocal clinical onset (UCO)* is the earliest time that a seizure occurrence is apparent by visually observing a patient. *Seizure onset* in this paper is synonymous with UEO. It is worth noting that the UEO almost always precedes the UCO by several seconds, and that many previously published papers defined "seizure onset" as the UCO.

2.2 Seizure Detection

Early attempts to detect seizures began in the 1970s (Viglione, Ordon & Risch, 1970; Liss, 1973) and primarily considered scalp EEG recordings to detect the clinical (and less frequently) electrographic onset of seizures. In 1990, Gotman reported a technique for automated seizure detection that achieved 76% detection accuracy at 1 Fp/hr for 293 seizures recorded from 49 patients (Gotman, 1990). In 1993, it was shown that the short-time mean Teager energy could be used to detect seizures from electrocorticograms (Zaveri, Williams & Sackellares, 1993). Their detector achieved 100% detection accuracy on an 11-seizure database. In 1995, Qu and Gotman presented an early seizure warning system trained on template EEG activity that achieved 100% detection accuracy at a mean detection latency of 9.35 seconds and false alarm rate of 0.2 Fp/hr (Qu & Gotman, 1995). Similar results were also reported using time- and frequency-domain features classified by a k-nearest neighbor classifier (Qu & Gotman, 1997). In 1998, Osorio et al. claimed 100% detection sensitivity with a mean detection latency of 2.1 seconds using a wavelet-based measure called seizure intensity. They analyzed a database of 125 patients, but the same data were used for training and validation (Osorio, Frei & Wilkinson, 1998). The algorithm was more extensively analyzed in 2002 using offline electrocorticogram recordings; again, 100% sensitivity was reported, with detection latencies ranging from 1.8 – 31.1 seconds (Osorio et al., 2002).

Several successful attempts at seizure detection using artificial neural network classifiers have been reported since 1996 (Khorasani & Weng, 1996; Webber et al., 1996; Gabor, 1998; Esteller, 2000). Evaluation of 31 distinct features (Esteller, 2000) showed that fractal dimension, wavelet packet energy, and mean Teager energy were especially promising for seizure detection. In 2001, Esteller reported a detector based on the line length feature that achieved a mean

detection latency of 4.1 seconds at a false alarm rate of 0.051 Fp/hr (Esteller et al., 2001). A total of 111 seizures (many subclinical) were used for validation. NeuroPace, Inc., subsequently reported a similar detector based upon this work that achieved 97% sensitivity at a mean detection latency of 5.01 seconds (Echauz et al., 2001). This detector was evaluated on 1265 hours of IEEG data, but was tuned heuristically in a patient-specific manner. The NeuroPace detector claims represent the state of the art in seizure detection performance. More complete reviews of the seizure detection and prediction literature are available elsewhere (Litt & Echauz, 2002; Gardner, 2004).

2.3 Novelty Detection

Traditional classification architectures rely on empirical risk minimization algorithms to specify "good" models for a classification decision function; as such, they are prone to over- or underfitting. In addition, their performance tends to be highly sensitive to parameter tuning and researcher skill. Statistical learning theory poses a structural risk minimization (SRM) criterion that balances the trade-off between good empirical performance (i.e., classification accuracy on training data) and good generalization ability (i.e., classification accuracy on unseen data). One popular application of SRM is the SVM, first presented in 1992 (Boser, Guyon & Vapnik, 1992). The basic idea behind the SVM is to find a hyperplane in a feature space that "optimally" separates two classes. Many other linear learning machines have been considered for this task, but the SVM yields a unique solution that can be shown to minimize the expected risk of misclassifying unseen examples (Vapnik, 1999). Training algorithms involve the solution of a well-known optimization problem, constrained quadratic programming, that is computationally efficient and yields global solutions. Several excellent tutorials provide historical context and details on the SVM (Burges, 1998; Bennett & Campbell, 2000; Müller et al., 2001).

In 1998, Schölkopf et al. introduced an extension to SVMs to estimate the support of a distribution (Schölkopf et al., 1999). Their motivation was to solve a simplified version of the density estimation problem, e.g., finding a minimum volume quantile estimator that is "simple." The solution they arrived at, the one-class SVM, was introduced for novelty detection.

Definition 1 (Novelty Detection). *Given a set of independent identically distributed (iid) training samples,* $x_1, \cdots, x_n \in X \subseteq \mathbf{R}^N$, *drawn from a probability distribution in feature space,* P, *the goal of novelty detection is to determine the "simplest" subset,* S, *of the feature space such that the probability that an unseen test point,* x', *drawn from* P *lies outside of* S *is bounded by an a priori specified value,* $\upsilon \in (0,1]$.

In the one-class formulation, data are first mapped into a feature space using an appropriate kernel function and then maximally separated from the origin using a hyperplane. The hyperplane parameters are determined by solving a quadratic programming problem, similar to the basic SVM case:

$$\min\left(\frac{1}{2}\|w\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho\right) \tag{1}$$

subject to

$$(w \cdot \Phi(x_i)) \geq \rho - \xi_i \quad i = 1,2,\ldots,l \quad \xi_i \geq 0 \tag{2}$$

where w and ρ are hyperplane parameters, Φ is the map from input space to feature space, ν is the asymptotic fraction of outliers (novelties) allowed, l is the number of training instances, and ξ is a slack variable. For solutions to this problem, w and ρ, the decision function

$$f(x) = \text{sgn}(w \cdot \Phi(x) - \rho) \tag{3}$$

specifies labels for examples, e.g., -1 for novelty instances. Figure 1 shows the geometry of the one-class SVM in feature space.

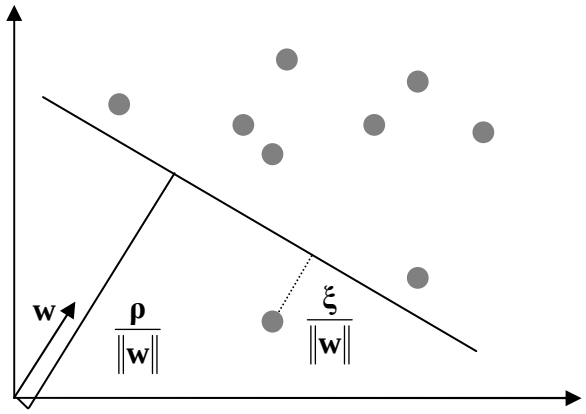


Figure 1: Geometry of the ν-SVM in feature space. Note the hyperplane and associated parameters, ρ and w, and the slack-variable, ξ, penalizing misclassifications.

Basic properties of the one-class SVM were proven in the initial paper (Scholkopf et al., 1999). The most important result is the interpretation of ν as both the asymptotic fraction of data labeled as outliers, and the fraction of support vectors returned by the algorithm. Implementation of the one-class SVM algorithm requires the following specifications: kernel function, kernel parameters, outlier fraction, and separating point in feature space. As with the basic SVM, there is no automatic method for specifying one-class SVM model parameters, but the interpretation of ν eases this task to some degree: the choice of outlier fraction should incorporate prior knowledge about the frequency of novelty occurrences (for example, a typical value for patient seizure frequency). Additionally, smaller values of ν increase the computational efficiency of the algorithm. The choice of origin as the separation point is arbitrary and affects the decision boundary returned by the algorithm. Other work (e.g., Hayton et al., 2001; Manevitz & Yousef, 2001) has addressed separation point selection given partial knowledge of outlier classes.

3 Methodology

In this section, we describe and discuss the experimental methods for detecting seizures under a novelty detection framework. A block diagram of this system is shown in Figure 2.

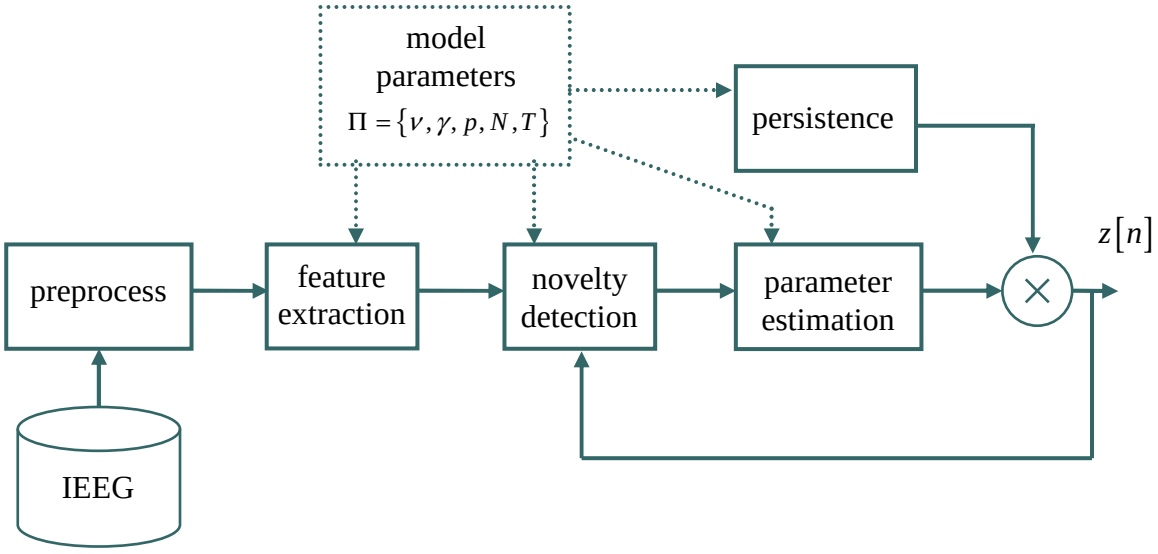


Figure 2: The seizure analysis architecture. IEEG time series data is block-processed in stages to produce the final output sequence, $z[n]$, indicating the presence/absence of ictal activity.

3.1 Human Data Preparation

The data analyzed were selected from intracranial EEG recordings of epilepsy patients implanted as part of standard evaluation for epilepsy surgery. Patients diagnosed with mesial temporal lobe epilepsy were observed in a hospital for 3 to 14 days. Between 20 and 36 electrodes were surgically placed either on the brain (grids or strips of electrodes), or in the brain substance (depth electrodes), and simultaneous IEEG and video were recorded. The IEEG data were amplified, bandpass-filtered (cutoffs at 0.1 Hz and 100 Hz), and digitized at 200 samples/second, 12 bits-per-sample resolution. Five consecutive patients with seizures arising from the temporal lobe(s) were selected for review, and the corresponding data were expertly and independently marked by two certified epileptologists to indicate UEO and UCO times. Collectively, these five patient records contain over 200 hours of data. Further details on this database are available elsewhere (D'Alessandro, 2001; Gardner, 2004).

Ictal epochs were selected from the focus channel for each temporal lobe seizure that a patient exhibited. Two patients exhibited some seizures with extra-temporal focal regions: those events were excluded from further analysis. Ictal epochs were extracted in a consistent manner such that the UEO occurred at a 10-minute offset within the epoch, allowing for analysis of both pre-ictal and post-ictal regimes. Interictal epochs from each patient were randomly selected. All epochs were expertly reviewed to ensure the absence of recording artifacts. The final data set consisted of 29 ictal- and 41 interictal epochs, each of 15-minute duration.

3.2 Feature Extraction

Many features have been proposed for seizure analysis (Esteller, 2000; D'Alessandro, 2001; Esteller et al., 2001). We selected a feature vector, q, composed of three energy-based statistics that have proven especially effective for seizure detection: mean curve length, CL; mean energy, E; and mean Teager energy, TE,

$$CL[n] = \log\left(\frac{1}{N} \sum_{m=n-N+2}^{n} \left| x[m] - x[m-1] \right| \right) \tag{4}$$

$$E[n] = \log\left(\frac{1}{N} \sum_{m=n-N+1}^{n} x[m]^2 \right) \tag{5}$$

$$TE[n] = \log\left(\frac{1}{N} \sum_{m=n-N+3}^{n} \left(x[m-1]^2 - x[m]\, x[m-2] \right) \right) \tag{6}$$

where $x[m]$ is an EEG time series, and N is the window length. We applied logarithmic scaling for feature normalization. Features were extracted using a block processing approach. In block processing, the data are windowed, a feature vector is computed, and the window is advanced in time. The selection of window length is an important issue (Esteller, 2000). Values typically range between 0.25 and 5 seconds; we used 1-second windows with 0.5-second overlap.

3.3 One-Class SVM

Feature extraction was performed on interictal epochs to generate feature vectors for training. A one-class SVM classifier was implemented using LIBSVM, a freely-available library of SVM tools available from http://www.csie.ntu.edu.tw/~cjlin/libsvm. A Gaussian radial basis function ($\gamma = 1.0$) was selected as the kernel function, and $\nu = 0.1$ was chosen consistent with the estimated fraction of ictal data. The resulting classifier model was stored for subsequent use in testing.

3.4 Parameter Estimation

For a stationary process, the one-class SVM novelty parameter, ν, asymptotically equals the outlier fraction. We exploit this property by training on features which strongly discriminate interictal from ictal EEG: features are stationary during interictal periods, but change markedly during periods of seizure activity, causing significant changes in the empirical outlier fraction.

We modeled classifier outputs, $y \in \{+1, -1\}$, as (iid) Bernoulli random variables where $P(novelty) = P(y = -1) = \nu$. We assumed that $\nu = \nu_0$ for interictal EEG, and $\nu = \nu_1 > \nu_0$ for ictal EEG. At each output sample, we computed the maximum likelihood estimate of the outlier fraction, $\hat{\nu}$, as

$$\hat{\nu} = \frac{n_{neg}}{n} = \frac{1}{2}\left(1 - \frac{1}{n}\sum_{i=1}^{n} y[i]\right) \tag{7}$$

where n_{neg} is the number of negative output occurrences in the n most recent samples of y. Note that the sequence length, n, affects the adaptation rate of the system. We then used this estimate to compute a seizure event indicator variable,

$$z[k] = \mathrm{sgn}\left(\hat{\nu} - C\right) \tag{8}$$

where $z = +1$ if a seizure is indicated or $z = -1$ otherwise, and $C \in [0,1]$ is a threshold parameter. Thresholding is equivalent to a standard hypothesis test of $H_0 : \nu = \nu_0$ vs. $H_1 : \nu = \nu_1$ where the null hypothesis is rejected if $\hat{\nu} > C$. For nominal values of $n = 20$ and $\nu_0 = 0.1$, we retained the null hypothesis (that is, we declared a frame to be interictal) if we observed fewer than five novelty outputs ($C = 0.8$). Under the iid assumption, this rule has a 4.33% chance of falsely rejecting the null hypothesis (i.e., producing a false positive). The chance of committing a Type II error (i.e., producing a false negative) depends on the unknown value ν_1. We calculated this error rate for several plausible values of ν_1 in Section 4.2.

3.5 Persistence (Detector Refractory Period)

During early experiments we observed that the detector tended to generate novelty events (i.e., "fire") in bursts, with increasing frequency near seizure onset. This behavior may indicate the presence of preictal states, periods of EEG activity that are likely to transition from interictal to ictal state. The bursty behavior can be problematic for performance assessment as multiple detections of a single seizure, or multiple false positive declarations may occur during a short interval of time. To address this problem, a refractory parameter, T_R, was introduced to the detection system. The refractory parameter specifies an interval during which the detector, if triggered, maintains its state and ignores subsequent triggers. In this sense it behaves like a "one-shot" device familiar from digital circuits. The use of this refractory rule is termed persistence.

Persistence offers an improvement to the basic system beyond false positive rate improvement: it allows for the characterization of the detector over a range of detection time horizons. As persistence decreases, one expects the false positive rate to increase and the detection latency to approach zero seconds. Conversely, as persistence increases, one expects the false positive rate to decrease, asymptotically approaching a value determined jointly by the novelty parameters of the system (some fraction of the data will always be novel) and the actual novelty rate due to epileptiform activity. Figure 3 illustrates the use of persistence. We heuristically set the detector persistence to $T_R = 180$ seconds for our experiments.

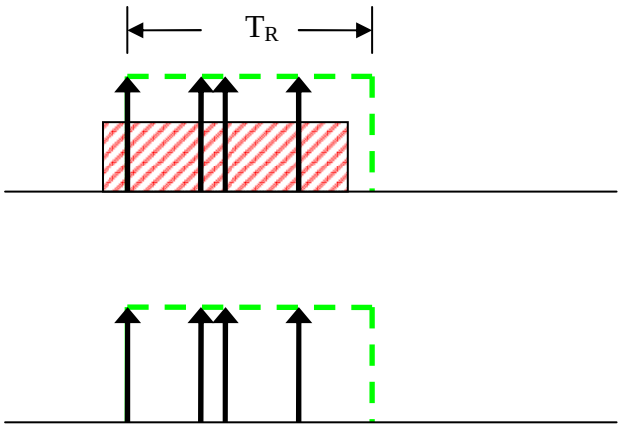


Figure 3: Examples of persistence for improving detector false alarm performance. (*Top*) Ictal epoch showing seizure activity (*red, diagonal hatching*). (*Bottom*) Interictal epoch. When persistence is applied, detections (*arrows*) are treated as a single event (*dashed green line*).

3.6 Performance Metrics

The detector was evaluated using LOO-CV and identical model parameters for each patient. Training was only performed using interictal epochs, however, testing was performed on each ictal segment, in addition to the withheld interictal epoch. This scheme yields $C\left(N_{BL},1\right)$ interictal- and $C\left(N_{BL},1\right)\times N_{SZ}$ ictal statistics per patient, where N_{BL}, N_{SZ} are the patient-specific number of interictal and ictal epochs, respectively. From these statistics we estimate three key performance metrics: sensitivity, false positive rate, and mean detection latency.

The detector's sensitivity (9) and false positive rate (10) measure its classification accuracy:

$$S = \frac{TP}{TP + FN} \tag{9}$$

$$FPR = \frac{FP}{T} \tag{10}$$

where *TP*, *FN*, and FP are the number of block true positives, block false negatives, and block false positives; and T is the duration (in hours) of the data analyzed. A block true positive occurs when the detector output, after applying persistence, correctly identifies an interval containing a seizure onset (c.f., Figure 4). Block false negatives and false positives occur when the detector incorrectly labels interictal and ictal intervals, respectively.

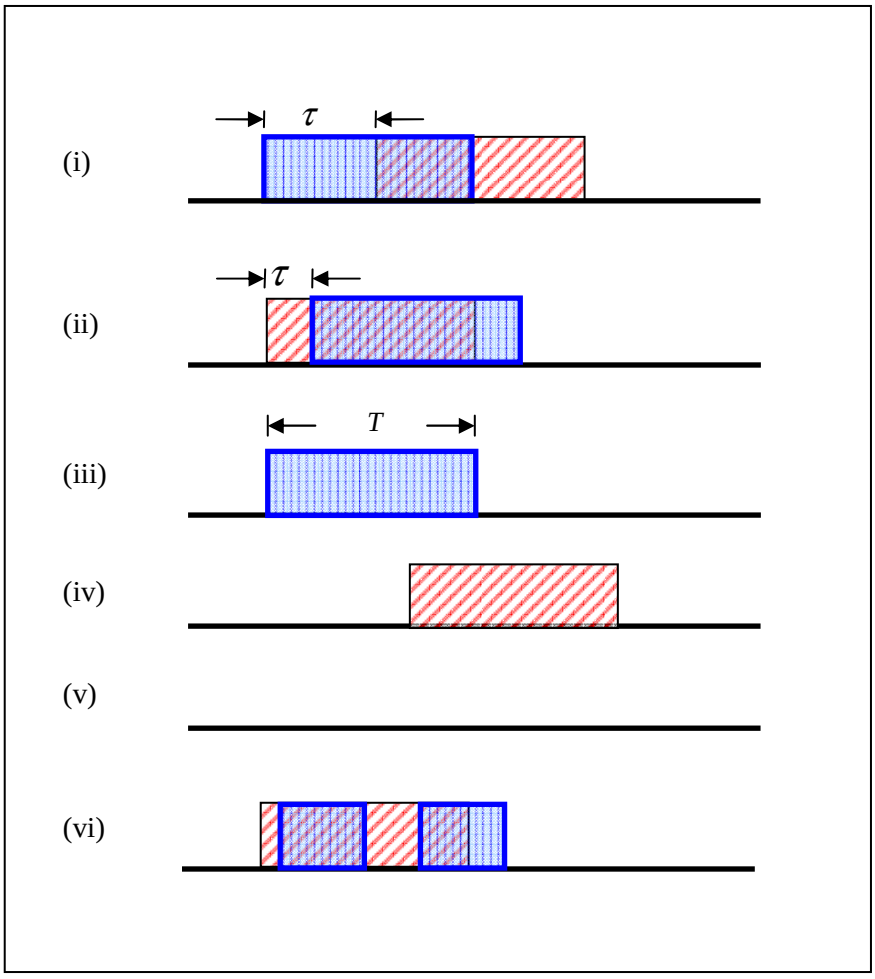


Figure 4: Temporal relationships considered in detector evaluation: intervals representing detected novelty (*blue, vertical hatching*) and ictal activity (*red, diagonal hatching*). *(i), (ii)* Two examples of block true positives (e.g., the novelty output interval overlaps ictal activity). The detection latency, τ, is also shown. An early-detection results in negative latency. *(iii)* A false positive detector error. *(iv)* A false negative detector error. *(v)* A true negative. *(vi)* An example of a degenerate case (multiple detection) producing both a true positive and a false positive event.

Mean detection latency (11) measures detector responsiveness:

$$\mu_\tau = \frac{1}{N}\sum_{i=1}^{N}\tau_i \tag{11}$$

where τ_i is the detection latency of each detected seizure. A negative latency indicates seizure event detection prior to the expert-labeled onset time.

3.7 Benchmark Novelty Detection

To provide a reference for the relative performance of our algorithm, and the general application of unsupervised learning to the seizure detection problem, we implemented a simple benchmark novelty detection algorithm.

During training, feature vectors, q, were extracted from IEEG time series (c.f. 3.2) and used to estimate the covariance matrix, Σ, and mean, μ, of the training data. We subsequently computed the Mahalanobis distances

$$D_M(q) = \sqrt{(q-\mu)^T \Sigma^{-1} (q-\mu)} \tag{12}$$

between each sample in the training data set and the centroid of the training set. An outlier threshold, K, was selected as the ν quantile of the Mahalanobis distances. As with the one-class SVM, we set $\nu = 0.1$.

During testing, feature vectors were thresholded to produce a frame-wise novelty sequence, y,

$$y[n] = \begin{cases} +1, & D_M(q_n) < K \\ -1, & D_M(q_n) \geq K \end{cases} \tag{13}$$

as a replacement for the SVM classifier output. This sequence was processed in the same manner as before (c.f. 3.4) to generate detections.

4 Results

In this section we present the results of both seizure detection approaches. Details on the effects of varying the one-class SVM model parameters—ν, γ, p, N, and T— and results from a genetic algorithm optimization are given in Gardner (2004).

4.1 Performance

Detection statistics from our LOO-CV analysis are presented in Table 1. Columns show patient id, the number of epochs analyzed for interictal (N_{BL}) and ictal data (N_{SZ}), the fraction of false positive detections on interictal (FP_{BL}) and ictal (FP_{SZ}) trials, the fraction of seizure epochs producing false positives (M_{FP}), the fraction of false negative detections (FN), and the mean detection latency ($\bar{\tau}$). The bottom row of the table shows aggregate statistics weighted by the number of seizures or number of baselines.

Patient	N_{BL}	N_{SZ}	*One-class Novelty*					*Benchmark Novelty*				
			FP_{BL}	FP_{SZ}	M_{FP}	FN	$\overline{\tau}$	FP_{BL}	FP_{SZ}	M_{FP}	FN	$\overline{\tau}$
1	6	5	0.00	0.10	0.20	0	2.07	0.17	0.13	0.20	0	4.60
2	9	7	0.22	0.43	0.57	0	-13.6	0.33	1.03[c]	0.71	0	-9.51
3	10	6	0.40[a]	0.13	0.17	0	6.57	0.40[a]	0.17	0.17	0	7.08
4	10	6	1.00	0.48	0.17	0.12	-6.57	1.00	0.48	1.00	0.17	-2.08
5	6	5	0.00	0.03	0.20	0	-27.0[b]	0.33	0.17	0.20	0	-24.9[b]
	41	**29**	**0.39**	**0.28**	**0.28**	**0.029**	**-7.58**	**0.49**	**0.47**	**0.45**	**0.041**	**-4.76**

[a]All false positives occurred on a single ictal epoch.
[b]Several seizure onsets were originally mislabeled by as much as 110 seconds. Results in this table are calculated from the corrected markings.
[c]Note that multiple false positive events per epoch can produce fractional values greater than one.

Table 1: Summary of detection statistics. Bottom row of table summarizes aggregate statistics.

We estimate the FPR over interictal EEG from the data in Table 1 by dividing FP_{BL} by the epoch duration (0.25 hours), yielding 1.56 Fp/hr for the one-class technique, and 1.96 Fp/hr for the benchmark technique. Since ictal events are rare, and the aggregate false positive rate on ictal segments is lower than the corresponding rate on interictal segments, we take the interictal FPR as an asymptotic measure of the overall FPR.

We reviewed the results for those patients (2, 4, and 5) with negative mean detection latencies. For each of these patients we found that the distribution of detection latencies was skewed, and a fraction (less than one-third) of the models detected seizures early. The median detection latencies for these patients, which might give a more balanced view of performance, ranged between 1.5 and 9.8 seconds for both models; the one-class delays were always less than the benchmarked values.

The SVM seizure detector achieved 97.1% sensitivity and a mean detection latency of -7.58 seconds at an estimated 1.56 Fp/hr. Representative IEEG time series, novelty sequences, and estimated outlier fractions for interictal- and ictal epochs are shown in Figures 5 and 6. As expected, the outlier fraction remained near its (small) nominal value except during periods of seizure activity. Onsets were detected quickly, and the entire seizure event—not just the onset—was correctly identified as novel. The near-zero false negative rate (FNR) of the detector was surprising because the data used for training originated from unknown states of consciousness (e.g., sleep or wake). Typically, seizure detection performance is drastically affected by patient state-of-consciousness; evaluation on larger data sets with concomitant sleep staging information will provide a better estimate of the true FNR.

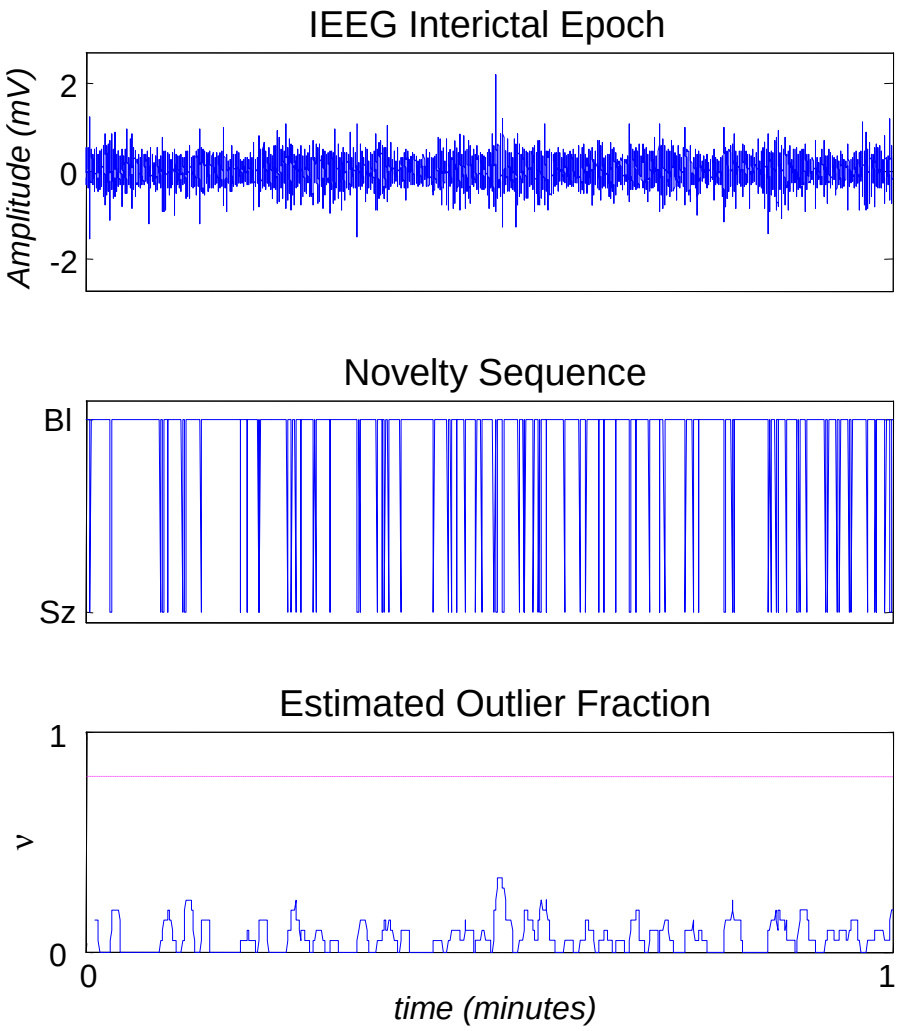


Figure 5: A typical interictal epoch. (*Top*) IEEG signal, (*Middle*) frame-wise output of the novelty detector, z, (*Bottom*) estimated outlier fraction (dashed line is 0.8). The mean of ν in this figure is 0.063.

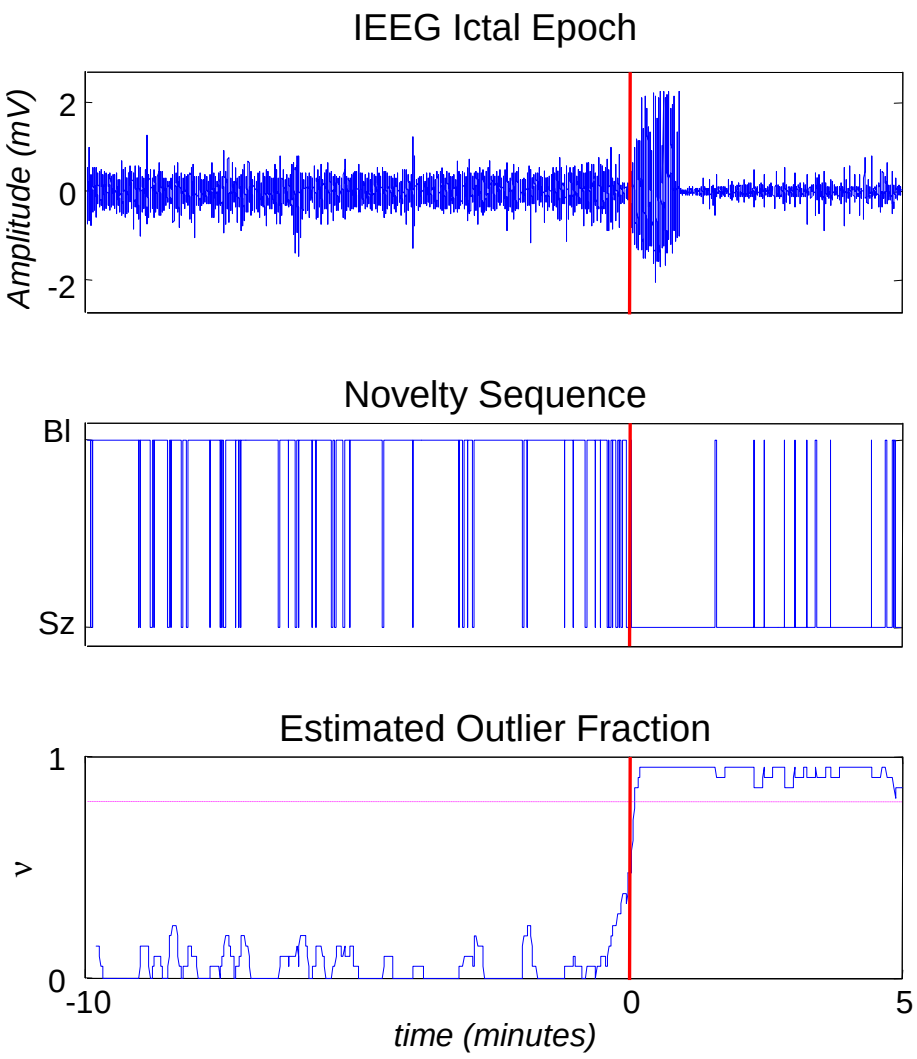


Figure 6: A typical ictal epoch. (*Top*) IEEG signal. The earliest electrographic change is visible as the beginning of the pinched region prior to the high-amplitude seizure onset. The UEO occurs at time zero, (*Middle*) frame-wise output of the novelty detector, z , (*Bottom*) estimated outlier fraction and 0.8 threshold. The detector has a latency of about 3 seconds in this example.

The SVM detector's mean detection latency outperformed all previously reported seizure detection algorithms. It should be noted, however, that this result is attributable to the large fraction of seizures (27%) that were detected early. This finding suggests the presence of two subclasses of seizures: those that are merely detectable, and those that may be predictable. These classes of seizures appear to be patient-dependent.

A direct comparison to other published detection algorithms is generally not meaningful due to the disparity of data sets that each research group operates on. However, NeuroPace (Echauz et al., 2001) previously evaluated their supervised algorithm on the same data set. While they did not perform cross-validation, and optimized in-sample for each patient, their reported results—a

mean detection latency of 5.01 seconds at 97% sensitivity and 0.013 Fp/hr—support the use of our approach.

The benchmark seizure detector achieved 95.8% sensitivity and a mean detection latency of -4.76 seconds at 1.96 Fp/hr. Both techniques are surprisingly effective at seizure detection, but the one-class SVM method performed consistently better, especially with respect to false positive rate. To gain insight into the relative performance difference between the two approaches, we examined the feature space for a single patient. Figure 7 shows the marginal distributions of features for both interictal and ictal data. It is clear from this figure that the feature distributions are highly skewed and possibly bimodal. An obvious explanation for the discrepancy in performance is that the normality assumption of the benchmark detector is severely violated, and the non-parametric estimation of the one-class SVM is better for modeling the data. The fact that the one-class SVM performs better, albeit on a limited number of patients, suggests that it tends to exclude vectors in feature space that appear more commonly when seizures occur as compared to the benchmark approach. Additionally, we examined the regions-of-support for this patient produced by each algorithm (Figure 8).

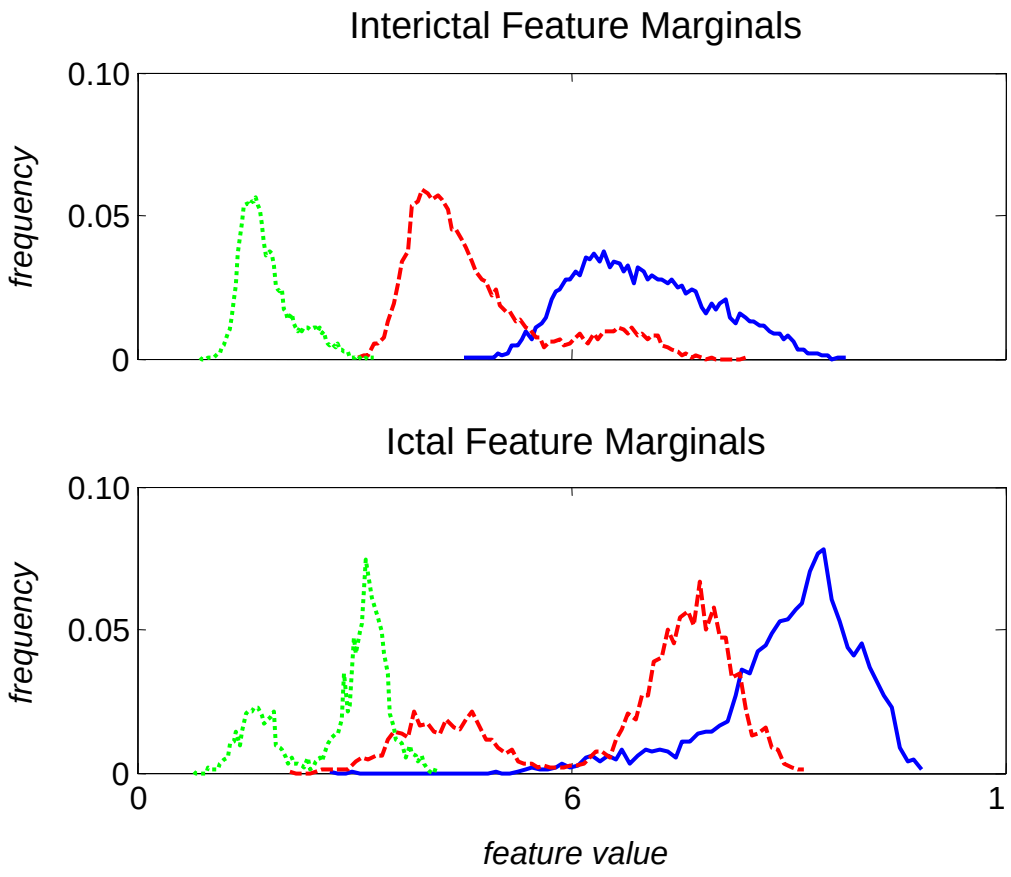


Figure 7: Representative marginals of the feature vector—E (*solid blue*), TE (*dashed red*), CL (*dotted green*)—for patient 5 corresponding to interictal (*top*) and ictal (*bottom*) frames.

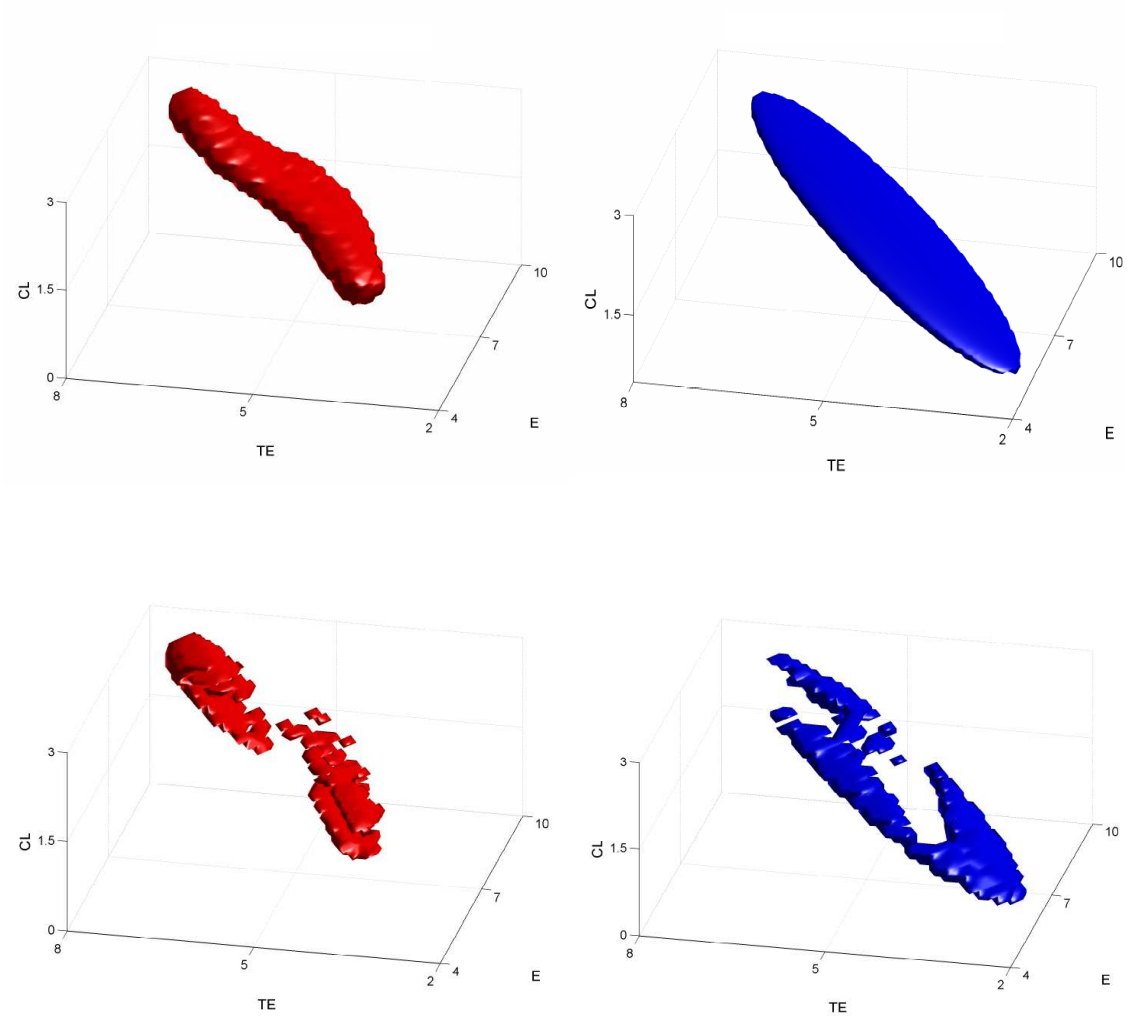


Figure 8: Representative isosurfaces in interictal feature space produced by each method. (*Top-left*) S_1, the $\nu = 0.1$ enclosing surface for the one-class SVM; (*Top-right*) S_2, the $\nu = 0.1$ enclosing surface for the benchmark method; (*Bottom-left*) $S_1 \setminus S_2$, the volume unique to the one-class SVM; (*Bottom-right*) $S_2 \setminus S_1$, the volume unique to the benchmark method.

Both approaches, SVM and Mahalanobis, find regions, S_1 and S_2, in feature space that include 90% of the observations from interictal data. It is interesting to note that, although the overlap of the regions, $S_1 \cap S_2$, must contain at least 80% of the training samples, 25.2% of the volume of S_1 and 37.5% of the volume of S_2 are non-intersecting. The minimum-volume property of the one-class SVM is also evident—S_1, is 84.4% of the volume of the benchmark technique—and may be a contributing factor to its increased performance over the Mahalanobis method.

4.2 Detector Output Analysis

We analyzed a sample of 850 interictal detector outputs and confirmed that the empirical outlier fraction equaled its nominal value, 0.10. We also investigated the performance of the detector output under the hypothesis $H_1 : \nu = \nu_1 > \nu_0$, assuming iid outputs. Illustratively, we considered $\nu_1 = 0.3$ and $\nu_1 = 0.5$ for the probabilities of a novelty occurrence during ictal epochs. Results in Table 1 show that the probability of falsely retaining the null hypothesis is small, and is of course smaller for $\nu_1 = 0.5$ than for $\nu_1 = 0.3$. This explains the superior FNR performance of the detector that we observed.

We performed logistic regressions between the outputs at times t, $t-1$, and $t-2$ to test our assumption that detector outputs are Bernoulli. We observed significant ($P < 0.001$) serial dependence. Empirically, the conditional probability of a novel detector output given a previous novelty output increases dramatically from 0.1 to 0.3. This analysis suggests that the detector output sequence obeys a Markov process where the probability at each point in time of a novelty is $P(z_t = -1) = \nu_0 = 0.1$, but the conditional probabilities for novelty outputs are $P(z_t = -1 \mid z_{t-1} = -1) = 0.3$ and $P(z_t = -1 \mid z_{t-1} = +1) = 0.078$.

We wrote a program to compute the probability of observing k novel outputs in N trials under the Markov process described above, and repeated our performance analysis. The results (Table 2) clearly show that the performance of the detector is worse under the serial dependence model.

	Binomial Output			Markovian Output		
	H_0	H_1	H_1	H_0	H_1	H_1
		$p_1 = 0.3$	$p_1 = 0.5$		$p_1 = 0.3$ $p_1^* = 0.5$	$p_1 = 0.5$ $p_1^* = 0.75$
Normal	0.9567	0.2374	0.0059	0.9175	0.3095	0.0736
Seizure	0.0433	0.7626	0.9941	0.0825	0.6905	0.8264

Table 2: An analysis of the hypothesis test for the detector output for both the binomial, and Markov cases for the rule where we declare an event if 5 or more out of 20 outputs are novelties. The estimated probability that an ictal frame is declared novel is p_1, and its corresponding conditional probability, $P(z_k = -1 \mid z_{k-1} = -1)$, is p_1^*.

5 Conclusions

Traditional approaches to seizure detection rely on binary classification. They require seizure data for training, which is difficult and invasive to collect, and do not address the class imbalance problem between interictal and ictal EEG, as less than 1% of EEG data from epileptic patients is seizure-related. These approaches assume that seizures develop in a consistent manner and seek to identify features and architectures that discriminate seizure EEG from "other" EEG. In contrast, we have presented a technique for seizure detection based on novelty detection that operates by modeling the dominant data class, interictal EEG, and declaring outliers to this class as seizure events. The success of our method relies on detecting change points in the empirical outlier fraction with respect to a feature space that strongly discriminates interictal from ictal EEG. If the feature space is well-chosen, the implication is that novelties are seizures.

In addition to achieving state-of-the-art performance, our technique overcomes three severe limitations of competing algorithms: (1) it does not need to be trained on seizures, (2) it does not require patient-specific tuning, and (3) it does not require knowledge of patient state-of-consciousness. While the false positive performance of the detector is not as good as other reported algorithms, this may be attributable to the presence of subclinical seizures, or other non-ictal anomalies in the data (e.g., normal periodic rhythms, artifacts, etc.). Furthermore, the acceptance by the research community of "hyperdetection strategies"—high false-positive rates and high-sensitivity detection—diminishes the emphasis placed on FPR metrics. For example, in early prototype reactive stimulation devices to treat seizures, the very brief and subthreshold stimulation involved in therapy appears to be well tolerated without any significant side-effects. In this setting, the need to prevent seizures (avoid false negative events), and the apparent relative harmlessness of false positive stimulations, encourage making the detector hypersensitive. As a final note, the entire algorithm is computationally efficient because of the use of the SVM and small novelty threshold.

Ongoing work includes methodological refinements for reducing FPR, and online implementations for validation on very large continuous, multichannel data sets.

Acknowledgements

This work was partly supported by funding from the Esther and Joseph Klingenstein Foundation, Whitaker Foundation, Epilepsy Foundation, American Epilepsy Society, Dana Foundation, Epilepsy Project, and the National Institutes of Health grant #RO1-NS041811-01.

References

K. P. Bennett and C. Campbell, "Support Vector Machines: Hype or Hallelujah?," *SIGKDD Explorations*, 2:1-13, 2000.

B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers." In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, Pittsburgh, PA, USA, 1992.

C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, 2:1-47, 1998.

M. D'Alessandro, "The Utility of Intracranial EEG Feature and Channel Synergy for Evaluating the Spatial and Temporal Behavior of Seizure Precursors." Ph.D. Dissertation, *Georgia Institute of Technology, Dept. of Electrical and Computer Engineering*. Atlanta, 2001.

F. Desobry and M. Davy, "Support Vector-Based Online Detection of Abrupt Changes." In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03)*, part IV, pp. 872-875, 2003.

J. Echauz, R. Esteller, T. Tcheng, B. Pless, B. Gibb, E. Kishawi, and B. Litt, "Long-Term Validation of Detection Algorithms Suitable for an Implantable Device," *Epilepsia*, supplement 7, 42:35-36, Dec. 2001.

R. Esteller, "Detection of Seizure Onset in Epileptic Patients from Intracranial EEG Signals," Ph.D. Dissertation, *Georgia Institute of Technology, Dept. of Electrical and Computer Engineering*. Atlanta, 2000.

R. Esteller, J. Echauz, T. Tcheng, B. Litt, and B. Pless, "Line Length: An Efficient Feature for Seizure Onset Detection." In *Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2:1707-1710, 2001.

A. J. Gabor, "Seizure Detection Using a Self-organizing Neural Network: Validation and Comparison with other Detection Strategies," *Electroencephalography and Clinical Neurology*, 107(1):27 – 32, 1998.

A. Gardner, "A Novelty Detection Approach to Seizure Analysis from Intracranial EEG," Ph.D. Dissertation, *Georgia Institute of Technology, Dept. of Electrical and Computer Engineering*. Atlanta, 2004.

J. Gotman, "Automatic Seizure Detection: Improvements and Evaluation," *Electroencephalography and Clinical Neurophysiology*, 76:317-24, 1990.

P. Hayton, L. Tarrasenko, B. Schölkopf, and P. Anuzis, "Support Vector Novelty Detection Applied to Jet Engine Vibration Spectra." In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pp. 946-952. MIT Press, 2001.

E. R. Kandel, J. H. Schwartz, and T. M. Jessel, *Principles of Neural Science*. Prentice-Hall, New Jersey, 1991.

K. Khorasani and W. Weng, "An Adaptive Structure Neural Networks with Application to EEG Automatic Seizure Detection, " *Neural Networks*, 9(7):1223 – 1240, 1996.

S. Liss, "Apparatus for Monitoring and Counteracting Excess Brain Electrical Energy to Prevent Epileptic Seizures and the Like." US patent #3850161, 1973.

B. Litt and J. Echauz, "Prediction of Epileptic Seizures," *The Lancet Neurology*, 1:22-30, 2002.

L. Manevitz and M. Yousef, "One-Class SVMs for Document Classification," *Journal of Machine Learning Research*, 2:139-154, 2001.

K.-R. M. Müller; G. Ratsch, K. Tsuda, K.; B. Schölkopf, "An Introduction to Kernel-based Learning Algorithms," *IEEE Transactions on Neural Networks*, 2:181-201, 2001.

I. Osorio, M. G. Frei, and S. B. Wilkinson, "Real-time Automated Detection and Quantitative Analysis of Seizures and Short-term Prediction of Clinical Onset," *Epilepsia*, 39:615-27, 1998.

I. Osorio, M. G. Frei, J. Giftakis, T. Peters, J. Ingram, M. Turnbull, M. Herzog, M. T. Rise, S. Schaffner, R. A. Wennberg, T. S. Walczak, M. W. Risinger, and C. Ajmone-Marsan, "Performance Reassessment of a Real-time Seizure-detection Algorithm on Long ECoG Series," *Epilepsia*, 43:1522-1535, 2002.

H. Qu and J. Gotman, "A Seizure Warning System for Long-term Epilepsy Monitoring," *Neurology*, 45:2250-2254, 1995.

H. Qu and J. Gotman, "A Patient-specific Algorithm for the Detection of Seizure Onset in Long-term EEG Monitoring: Possible Use as a Warning Device," *IEEE Transactions on Biomedical Engineering*, 44:115-22, 1997.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the Support of a High-dimensional Distribution," Microsoft Research, Redmond, WA, USA MSR-TR-99-87, 1999.

D. Tax and P. Laskov, "Online SVM Learning: From Classification to Data Description and Back," *Proceedings of the 13th IEEE Workshop on Neural Network for Signal Processing*, pp. 499-508, 2003.

V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1999.

S. S. Viglione, V. A. Ordon, and F. Risch, "A Methodology for Detecting Ongoing Changes in the EEG Prior to Clinical Seizures." In *21st Western Institute on Epilepsy*, 1970.

W. R. Webber, R. P. Lesser, R. T. Richardson, and K. Wilson, "An Approach to Seizure Detection Using an Artificial Neural Network (ANN)," *Electroencephalography and Clinical Neurophysiology*, 98(4):250-2722, 1996.

H. P. Zaveri, W. J. Williams, and J. C. Sackellares, "Energy Based Detection of Seizures." In *15th Annual International Conference on Engineering and Medicine in Biology*, pp. 363-364, 1993.

Journal of Machine Learning Research 7 (2006) 1045–1078 Submitted 2/06; Published 6/06

A Graphical Representation of Equivalence Classes of AMP Chain Graphs

Alberto Roverato ROVERATO@UNIMORE.IT
Department of Social, Cognitive and Quantitative Sciences
University of Modena and Reggio Emilia
Viale Allegri 9
I-42100 Reggio Emilia, Italy

Milan Studený STUDENY@UTIA.CAS.CZ
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 4
18208 Prague 8 Libeň, Czech Republic

Editor: David Madigan

Abstract

This paper deals with chain graph models under alternative AMP interpretation. A new representative of an AMP Markov equivalence class, called the *largest deflagged graph*, is proposed. The representative is based on revealed internal structure of the AMP Markov equivalence class. More specifically, the AMP Markov equivalence class decomposes into finer *strong equivalence* classes and there exists a distinguished strong equivalence class among those forming the AMP Markov equivalence class. The largest deflagged graph is the largest chain graph in that distinguished strong equivalence class. A composed graphical procedure to get the largest deflagged graph on the basis of any AMP Markov equivalent chain graph is presented. In general, the largest deflagged graph differs from the AMP essential graph, which is another representative of the AMP Markov equivalence class.

Keywords: chain graph, AMP Markov equivalence, strong equivalence, largest deflagged graph, component merging procedure, deflagging procedure, essential graph

1. Introduction

This paper studies chain graph models under the alternative interpretation introduced by Andersson, Madigan and Perlman (2001). In general, a *chain graph model* is a statistical model in which a chain graph is used to represent the conditional independence structure defining the statistical model. The vertices of the graph represent random variables and the conditional independence structure is determined through the respective *Markov property*. The class of chain graphs was introduced and the original interpretation was given by Lauritzen and Wermuth (1984); see also Lauritzen and Wermuth (1989). The mathematical theory of chain graphs was developed by Frydenberg (1990), who formally defined the Markov property corresponding to the original interpretation. Following the standard literature in this field, we will refer to this property as the *LWF* Markov property.

More recently, another interpretation of chain graphs was introduced by Andersson, Madigan and Perlman (1996); see also Andersson et al. (2001). This interpretation leads to an alternative Markov property and we will refer to it as the *AMP* Markov property.

Two different chain graphs may be equivalent with respect to a considered Markov property, by which is meant that they define the same statistical model. The distinction between different interpretations of chain graphs is reflected by the different concepts of equivalence, namely the *LWF Markov equivalence* and the *AMP Markov equivalence*.

From a statistical perspective, the point of interest is a statistical model. However, if we represent a statistical model using an arbitrary graph in the respective *Markov equivalence class*, then the non-unique nature of graphical description may result in difficulties. One type of difficulty concerns problems one can meet in structural learning of graphical models; see Section 2.3 of Chickering (2002) for a review in the case of acyclic directed graphs. A solution to these problems may be provided by a suitable choice of a unique *representative* of each Markov equivalence class, that is, of a particular element in that equivalence class. The choice of a suitable representative is also important from the perspective of causal inference in chain graphs (Lauritzen, 2001, Section 11.2).

The problem of the representative choice has a natural solution in the LWF case. Frydenberg (1990) showed that every LWF equivalence class contains the *largest chain graph*, which is the graph with the largest amount of undirected edges within the LWF equivalence class. Furthermore, every arrow in the largest chain graph is an arrow with the same direction in every chain graph from the class. The largest chain graph is uniquely determined and can serve as a natural representative of the LWF equivalence class. Moreover, there exist at least two procedures that transform every chain graph into the largest chain graph of the respective LWF equivalence class Roverato (2005); Volf and Studený (1999).

However, the situation is different in the AMP case. It is not clear what is a natural representative of an AMP equivalence class and, in particular, the notion of the "largest AMP chain graph" makes no sense. Andersson et al. (2001) proposed to represent an AMP equivalence class by a so-called *AMP essential graph*. Every arrow in the AMP essential graph is either an arrow with the same direction or an undirected edge in every chain graph from the equivalence class. Their terminology was inspired by the case of acyclic directed graph models, in which case the corresponding equivalence class has a suitable representative called the *essential graph* (Andersson et al., 1997). Indeed, if an AMP equivalence class contains an equivalence class of acyclic directed graphs, then the AMP essential graph coincides with the respective essential graph; see Proposition 4.2 in Andersson and Perlman (2006). However, as of now, there is no algorithm to construct the AMP essential graph, as in the LWF case. Furthermore, in the case that the AMP equivalence class contains a completely undirected graph, it may happen that the AMP essential graph has some arrows, and this unpleasant phenomenon was already reported in Section 7 of Andersson et al. (2001).

The aim of this paper is to provide an alternative solution to the problem of unique graphical representation of AMP equivalence classes. The point is that AMP equivalence classes have a more complicated structure than LWF equivalence classes. We succeed in revealing this structure and provide a representing graph as well as an algorithm for its construction. Our solution, called the *largest deflagged graph*, is different from the AMP essential graph proposed in Andersson et al. (2001). Nevertheless, if an AMP equivalence class contains an acyclic directed graph then our representative reduces to the essential graph of the corresponding equivalence class of acyclic directed graphs. Moreover, it provides a better solution if the AMP equivalence class contains an undirected

graph because then, as one would expect, that undirected graph coincides with our largest deflagged graph.

These are the main contributions of the paper:

1. We introduce a new concept, namely, the concept of *strong equivalence* of chain graphs. An important observation is that every AMP equivalence class of chain graphs decomposes into strong equivalence classes.

2. Every strong equivalence class has a unique representative given by the largest chain graph of the class. We introduce a procedure, called the *component merging procedure*, which starts from any chain graph G and, by replacing arrows with undirected edges, finds the largest graph in the class of chain graphs strongly equivalent to G.

3. There exists a unique distinguished strong equivalence class among those forming an AMP equivalence class. Its elements have the largest amount of immoralities and the least possible amount of flags. We call the graphs in this class the maximally deflagged graphs, briefly the *deflagged graphs*. We introduce a procedure, called the *deflagging procedure*, which starts from any chain graph G and, by replacing undirected edges with arrows, produces a deflagged graph $\widehat{G}$ AMP equivalent to G.

4. We propose to characterize every AMP equivalence class by means of the respective *largest deflagged graph*. This representative can be constructed by applying the component merging procedure to the chain graph $\widehat{G}$ obtained by the deflagging procedure.

The next section recalls basic graphical concepts. Then, in Section 3, the main results in the LWF case are recalled in order to let the reader see some analogy. In Section 4, we give an overview of our new results and illustrate them by an example. The results on strong equivalence of chain graphs are formulated in Section 5. They appear to be analogous to the results valid in the LWF case. In Section 6, we present a deflagging procedure to get a deflagged graph in a given AMP equivalence class. Section 7 contains some concluding remarks. Proofs of the main results are moved to the Appendix.

2. Basic Concepts

In this paper we consider graphs that admit both directed edges, called arrows, and undirected edges, called lines. Formally, given a non-empty finite set N, an *arrow* over N is an ordered pair (a,b) of distinct elements of N and a *line* over N is a subset $\{a,b\}$ of N of cardinality two, that is, an unordered pair of distinct elements of N. A *hybrid graph* is a triplet $H \equiv (N, \mathcal{A}, \mathcal{L})$ where N is a finite non-empty set of *nodes*, $\mathcal{A}$ a set of arrows over N and $\mathcal{L}$ a set of lines over N such that no multiple edges are allowed, which means that if $(a,b) \in \mathcal{A}$ then $(b,a) \notin \mathcal{A}$ and $\{a,b\} \notin \mathcal{L}$. To express that N is the set of nodes of H we also say that H is a hybrid graph *over* N.

Given a hybrid graph H, we will write $a \longrightarrow b$ in H or $b \longleftarrow a$ in H to denote $(a,b) \in \mathcal{A}$. Analogously, we will write $a \multimap\!\!\!\!- b$ in H or $b \multimap\!\!\!\!- a$ in H if $\{a,b\} \in \mathcal{L}$. This notation is in accordance with usual pictures. An ordered pair $[a,b]$ of distinct nodes in H will be called an *edge* in H if $a \longrightarrow b$ in H, $a \multimap\!\!\!\!- b$ in H or $a \longleftarrow b$ in H. Observe that $[a,b]$ is an edge iff $[b,a]$ is an edge: this means that edges can be viewed as unordered pairs of distinct nodes. If $[a,b]$ is an edge in H we also say that a and b are *adjacent* in H.

A set of nodes $C \subseteq N$ is *connected* in H if, for every $a, b \in C$, there exists an *undirected path* connecting them, that is, a sequence of distinct nodes $a = c_1, \ldots, c_n = b$, $n \geq 1$ such that $c_i \multimap\!\!-\!\!\multimap c_{i+1}$ in H for $i = 1, \ldots, n-1$. A (connectivity) *component* in H is a maximal connected set in H with respect to set inclusion. Evidently, components in a hybrid graph are pairwise disjoint.

Given a set of nodes $A \subseteq N$ in a hybrid graph H, the set of *parents* of nodes in A, denoted by $\mathrm{pa}_H(A)$, is the set

$$\mathrm{pa}_H(A) \equiv \{b \in N;\ b \longrightarrow a \ \text{ in } H \text{ for some } \ a \in A\}.$$

A *descending path* in H from a node a to a node b is a sequence of distinct nodes $a = c_1, \ldots, c_n = b$, $n \geq 1$ such that either $c_i \longrightarrow c_{i+1}$ in H or $c_i \multimap\!\!-\!\!\multimap c_{i+1}$ in H for $i = 1, \ldots, n-1$. If there exists a path of this kind in H then we say that a is an *ancestor* of b in H. The set of ancestors in H of nodes in a set $A \subseteq N$ will be denoted by $\mathrm{an}_H(A)$; observe that one has $A \subseteq \mathrm{an}_H(A)$.

An *undirected graph* is a hybrid graph without arrows, that is, $\mathcal{A} = \emptyset$. A set $K \subseteq N$ is *complete* in an undirected graph H if, $\forall\, a, b \in K$, $a \neq b$, one has $a \multimap\!\!-\!\!\multimap b$ in H. Given a hybrid graph H over N, the respective *underlying graph* is an undirected graph H^u over N such that $a \multimap\!\!-\!\!\multimap b$ in H^u iff $[a, b]$ is an edge in H.

A *directed graph* is a hybrid graph without lines, that is, $\mathcal{L} = \emptyset$. A directed graph H is *acyclic* if there is no directed cycle in H, that is, there is no sequence of nodes $d_0, \ldots, d_{n-1}, d_n = d_0$, $n \geq 3$ such that $d_0, \ldots, d_{n-1}$ are distinct and, $\forall\, i = 0, \ldots, n-1$, $d_i \longrightarrow d_{i+1}$ in H.

The concept of a *chain graph* (CG) can be introduced in two equivalent ways. Note that we are going to use the abbreviation *CG* in the rest of the paper. The first definition is that a CG is a hybrid graph H whose components can be ordered to form a chain, that is, a sequence $C_1, \ldots, C_m$, $m \geq 1$ such that

- if $a \multimap\!\!-\!\!\multimap b$ in H then $a, b \in C_i$ for some i,
- if $a \longrightarrow b$ in H then $a \in C_i, b \in C_j$ with $i < j$.

Note that this is the reason for which some authors call the components in a CG *chain components*. A consequence of this definition is that every CG has a *terminal component*, that is, a component T such that there is no arrow $a \longrightarrow b$ in H with $a \in T$. The other definition is that a CG is a hybrid graph H without *semi-directed cycles*. A semi-directed cycle of the length n is a sequence of nodes $d_0, \ldots, d_{n-1}, d_n = d_0$ with $n \geq 3$ such that $d_0, \ldots, d_{n-1}$ are distinct, $d_0 \longrightarrow d_1$ in H and, $\forall\, i = 1, \ldots, n-1$, either $d_i \longrightarrow d_{i+1}$ in H or $d_i \multimap\!\!-\!\!\multimap d_{i+1}$ in H. See Lemma 2.1 in Studený (1997) for the proof of equivalence of both definitions of a CG. It is easy to see that every undirected graph and every acyclic directed graph is a CG.

Given a hybrid graph H over N and $\emptyset \neq A \subseteq N$, the *induced subgraph* of H for A, denoted by H_A, is the graph $H_A \equiv (A, \mathcal{A} \cap (A \times A), \mathcal{L} \cap \mathcal{P}(A))$, where $\mathcal{P}(A) \equiv \{B; B \subseteq A\}$. We will deal with a few special induced subgraphs. A *complex* in a hybrid graph H is an induced subgraph of H of the form $a \longrightarrow c_1 \multimap\!\!-\!\!\multimap \ldots \multimap\!\!-\!\!\multimap c_s \longleftarrow b$, $s \geq 1$, which means that no other edge between distinct nodes $\{a, b, c_1, \ldots, c_s\}$ is present in H. An example of a complex is shown in the left-hand picture of Figure 1. A special case of a complex is an *immorality*, which is a configuration $a \longrightarrow c \longleftarrow b$ in H where a, b, c are distinct nodes and $[a, b]$ is not an edge in H. An example of an immorality is shown in the middle picture of Figure 1. Two CGs G and H over N will be called *complex equivalent* iff they have the same underlying graph and complexes. Given a CG G, the class of CGs that are complex equivalent to G will be denoted by $\mathcal{G}$.

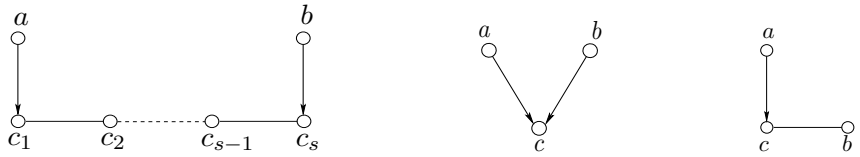


Figure 1: A complex, an immorality and a flag.

A *flag* in a hybrid graph H is another induced subgraph of H for three nodes, namely $a \longrightarrow c \multimap b$ where a,b,c are distinct nodes and $[a,b]$ is not an edge in H. An example of a flag is shown in the right-hand picture of Figure 1. A *triplex* in a hybrid graph H is a pair $\langle\{a,b\},c\rangle$ such that either $a \longrightarrow c \longleftarrow b$ is an immorality in H, $a \longrightarrow c \multimap b$ is a flag in H or $a \multimap c \longleftarrow b$ is a flag in H. All three different versions of a triplex are shown in Figure 2. Two CGs G and H over N will be called *triplex equivalent* iff they have the same underlying graph and triplexes. Note that coincidence of triplexes is understood as follows. If, for instance, $a \longrightarrow c \longleftarrow b$ is an immorality in G then it need not be an immorality in H but it has to be one of three versions of the triplex $\langle\{a,b\},c\rangle$. Given a CG H, the class of CGs that are triplex equivalent to H will be denoted by $\mathbb{H}$.

3. Representation of LWF Equivalence Classes

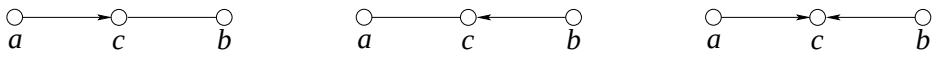
In this section we recall known results concerning the LWF case. The aim is to help the reader to realize the analogy between these former results and our new results on strong equivalence of CGs presented in Section 5. Moreover, an overview of the results in the LWF case will indicate what is the main difference from the AMP case, which is reported in Section 4.

3.1 Largest Chain Graph in a LWF Equivalence Class

In this paper, we omit the formal definition of LWF Markov property and LWF Markov equivalence; this can be found in Frydenberg (1990). Instead, we recall Frydenberg's graphical characterization of LWF equivalence of CGs (see Proposition 5.6 in Frydenberg, 1990). He showed that two CGs over the same set of nodes are LWF Markov equivalent iff they are complex equivalent.

The second crucial point is that every LWF equivalence class is endowed with a natural partial ordering. Supposing that $H = (N, \mathcal{A}_H, \mathcal{L}_H)$ and $G = (N, \mathcal{A}_G, \mathcal{L}_G)$ are two LWF equivalent CGs, we say that H is *larger* than G if $\mathcal{A}_H \subseteq \mathcal{A}_G$, that is

$$a \longrightarrow b \text{ in } H \text{ implies } a \longrightarrow b \text{ in } G, \tag{1}$$



(i) (ii) (iii)

Figure 2: Three different versions of the triplex $\langle\{a,b\},c\rangle$.

for every pair a and b of distinct nodes in H. Observe that the fact that G and H have the same underlying graph necessitates that $\mathcal{L}_G \subseteq \mathcal{L}_H$, that is

$$a \multimap b \text{ in } G \text{ implies } a \multimap b \text{ in } H, \tag{2}$$

which means H has 'more' lines than G. One can easily show that the relation defined by (1) is a partial ordering on every LWF equivalence class; we will write $H \geq G$ if (1) is fulfilled.

Third, Frydenberg also showed (Proposition 5.7 of Frydenberg, 1990) that every LWF equivalence class $\mathcal{G}$ has the largest element with respect to this ordering, that is, $G_\infty \in \mathcal{G}$ such that for every G in $\mathcal{G}$ one has $G_\infty \geq G$. Thus, this graph G_∞, named the *largest chain graph* of $\mathcal{G}$, can serve as a natural representative of $\mathcal{G}$.

3.2 Feasible Merging of Components

The last important point is that there are procedures which allow one to get the largest CG $G_\infty \in \mathcal{G}$ on the basis of any CG $G \in \mathcal{G}$ from the LWF equivalence class. At least three procedures of this kind have been presented in the literature; however, two of them are methodologically equivalent.

One of them could be a procedure based on Theorem 3.9 of Volf and Studený (1999). The basic idea is that some arrows in a CG $G \in \mathcal{G}$ are indicated as 'protected' arrows. Then all arrows in G which are not 'protected' are replaced with lines and the largest chain graph G_∞ of $\mathcal{G}$ is obtained.

Another procedure, called the *pool-component rule*, was presented in Section 5 of Studený (1997). The basic idea is that there is an elementary operation of merging components in a CG whose result is an LWF equivalent CG. By consecutive application of this operation, the respective largest chain graph can be obtained. However, the formal description of that elementary operation given in Studený (1997) is still awkward.

The third procedure is described in Roverato (2005). Its basic idea is essentially the same; an elementary step of that procedure consists of merging components of an 'insubstantial' meta-arrow, that is, of the bunch of arrows between two certain components. It is shown in Section 4 of Roverato (2005) that, by consecutive application of that elementary step, the respective largest CG is obtained. One can show that the elementary operations presented in Studený (1997) and Roverato (2005) are equivalent (see Studený et al., 2006), but the formal description of the operation presented in Roverato (2005) is much more elegant from the mathematical point of view. We decided to take it as the basis of the following definitions.

Definition 1 *(meta-arrow)*
Let G be a CG. A pair of components (U,L) in G such that there exists an arrow $a \longrightarrow b$ in G with $a \in U$ and $b \in L$ determines a meta-arrow *in G. More specifically, the meta-arrow is the collection of all arrows $a \longrightarrow b$ with $a \in U$ and $b \in L$. The component U will be called the* upper component *and the component L the* lower component *(of the meta-arrow). We will occasionally use the notation $U \rightrightarrows L$.*

Note that the above notion is a minor modification of the concept of a meta-arrow from Roverato (2005). The essential difference is that in Definition 1 we require that at least one arrow exists from a member of U to a member of L, while in Roverato (2005) a possibly empty collection of arrows from U to L was allowed. Thus, the concept of a meta-arrow used in this paper coincides with the concept of a non-empty meta-arrow from Roverato (2005). Since empty meta-arrows play no role

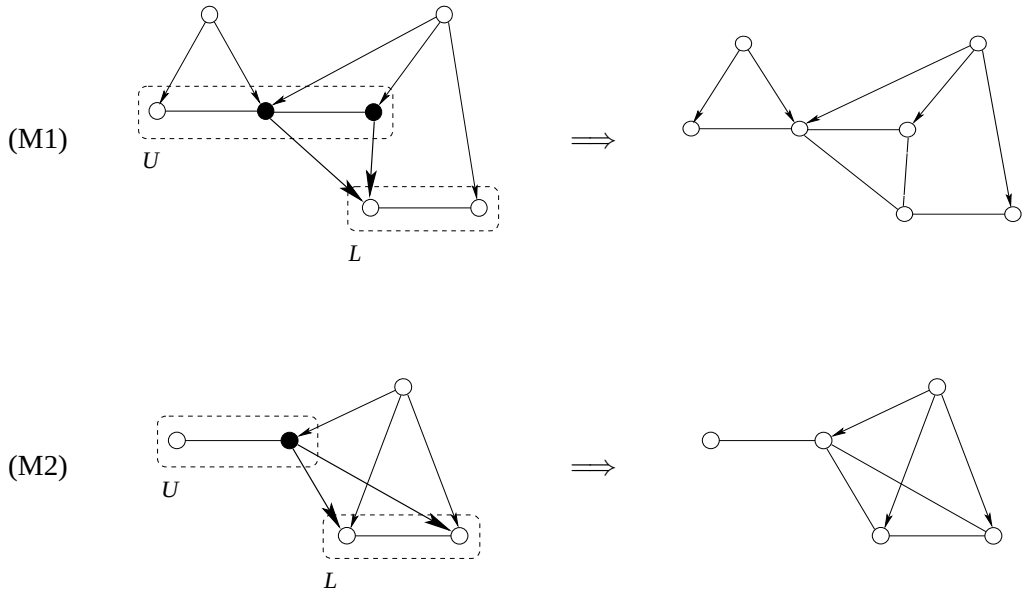


Figure 3: Two examples of feasible merging. The vertices belonging to the set K from (i) are filled in and arrows of the meta-arrow $U \rightrightarrows L$ are bold.

in a CG, we have decided to simplify our terminology. Our additional assumption also implies that the considered components U and L are different.

Definition 2 *(merging of components)*
By merging of components *in a CG G we understand the following operation applicable to G. Given a pair of components (U,L) which defines a meta-arrow, we replace all arrows of the meta-arrow $U \rightrightarrows L$ with lines and say that the resulting hybrid graph G' is obtained by merging of components U and L; more specifically, by* merging *of the upper component U and the lower component L.*

Note that the above terminology was inspired by terminology from Studený (2004). In general, the result of the operation of merging components in a CG need not be a CG. However, there are sufficient conditions for this; one of them is as follows.

Definition 3 *(feasible merging)*
Let (U,L) be a pair of components in a CG G that defines a meta-arrow in G. We say that merging of components U and L is feasible *(in G) if the following two conditions hold:*

(i) $K \equiv \mathrm{pa}_G(L) \cap U$ *is a complete set in G,*

(ii) $\forall\, b \in K \quad \mathrm{pa}_G(L) \setminus U \subseteq \mathrm{pa}_G(b)$.

Note that the assumption that (U,L) defines a meta-arrow implies that the set K in (i) is a non-empty set. Two examples of feasible merging are shown in Figure 3.

It is shown in Section 4 of Roverato (2005) that a hybrid graph G' obtained from a CG G by feasible merging of its components is a CG complex equivalent to G; actually, it is shown there that the requirements (i) and (ii) together establish a necessary and sufficient condition for this. In fact, that is the reason we decided to name this operation with CGs "feasible merging of components'' because the condition ensures that one remains in the same LWF equivalence class of CG after the merging operation. Moreover, it is also proven in Roverato (2005) that, by repeated application of this operation to a CG $G \in \mathcal{G}$, the respective largest CG $G_\infty \in \mathcal{G}$ is obtained.

4. Representation of AMP Equivalence Classes

In this section we reveal the internal structure of AMP Markov equivalence classes. First, we recall the graphical characterization of AMP equivalence. Then we introduce a special kind of equivalence of CGs, called *strong equivalence*, such that every AMP equivalence class decomposes into strong equivalence classes. Basic results on strong equivalence are postponed to Section 5. The next step is to introduce a special *flag ordering* between strong equivalence classes within a fixed AMP equivalence class. We show that the smallest element with respect to that ordering exists and, finally, we propose to represent the whole AMP equivalence class by a natural representative of that distinguished strong equivalence class, called *largest deflagged graph*.

4.1 Graphical Characterization of AMP Equivalence

The formal definitions of AMP Markov property and AMP Markov equivalence are omitted; they can be found in Andersson et al. (2001). Here we recall graphical characterization of AMP equivalence given by Andersson et al. (2001, Theorem 5). They showed that two CGs over the same set of nodes are AMP Markov equivalent iff they are triplex equivalent. An example of an AMP equivalence class is given in Figure 2. A further, less trivial, example containing ten CGs is given in Figure 4.

Given a CG H, let us consider the set $\mathbb{H}$ of all CGs triplex equivalent to H. If we consider the partial ordering of CGs in $\mathbb{H}$ defined by (1) then it may be the case that the largest CG in $\mathbb{H}$ does not exist. This is illustrated in Figure 2, where none of the three graphs is larger than the others, but also in Figure 4.

This is the main difference between the case of LWF equivalence and the case of AMP equivalence. In the LWF case, the key role is played by the ordering of CGs defined by (1). The result on the existence and uniqueness of the largest CG with respect to this ordering in each LWF equivalence class reported in Section 3 makes this object a natural representative of the LWF equivalence class. In the representation of an AMP equivalence class, the ordering defined by (1) also plays an important role, even though its use in this case is more subtle than in the LWF case. What is important is that every AMP equivalence class decomposes into some finer equivalence classes.

4.2 Definition of Strong Equivalence

Our decomposition of a given AMP equivalence class is based on the distinction between *triplex edges*, namely the arrows and lines that belong to a triplex, and non-triplex edges. More specifically, if two triplex equivalent CGs have identical triplex edges, then we say that they are strongly equivalent.

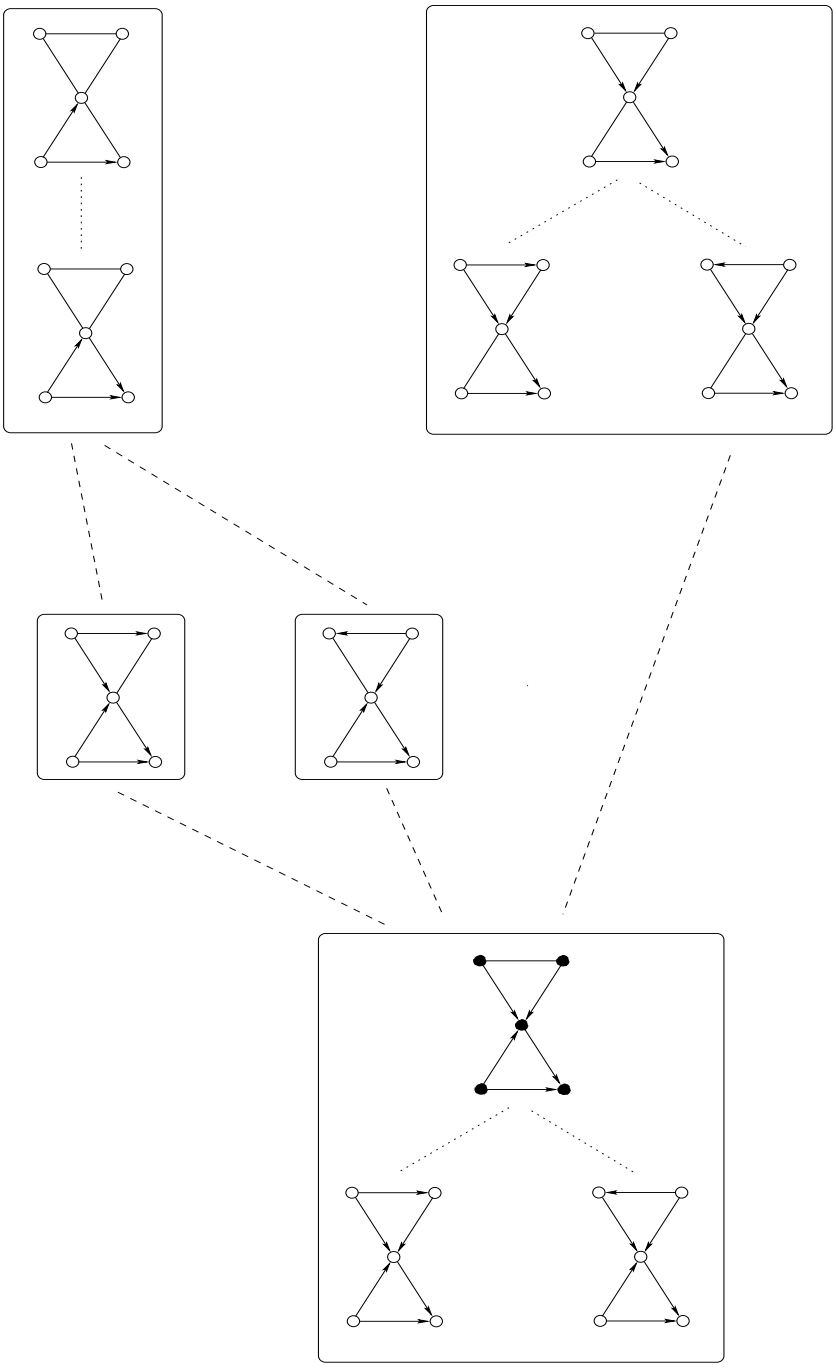


Figure 4: An example of an AMP equivalence class. The boxes represent strong equivalence classes. They are ordered by the flag ordering. There exists a unique largest graph within every strong equivalence class. The largest deflagged graph has vertices filled in.

Definition 4 *(strong equivalence of chain graphs)*
Let G, H be CGs over N. We say that they are strongly equivalent *iff*

[a] *G and H have the same underlying graph,*

[b] *an immorality $a \longrightarrow c \longleftarrow b$ occurs in G iff it occurs in H,*

[c] *a flag $a \longrightarrow c \multimap b$ occurs in G iff it occurs in H.*

It is easy to see that strongly equivalent CGs have the same complexes. In particular, they are both complex equivalent and triplex equivalent. On the other hand, two CGs which are both LWF and AMP Markov equivalent need not be strongly equivalent as shown, for example, by the graphs (i) and (ii) in Figure 2.

Given a CG H, the class of CGs that are strongly equivalent to H will be denoted by $\mathcal{H}$. In Figure 4, strong equivalence classes are represented by boxes. Note that, since all the graphs in $\mathcal{H}$ have the same triplex edges, it makes sense to say that $a \longrightarrow c$ is a triplex arrow in $\mathcal{H}$ if $a \longrightarrow c$ is a triplex arrow in every CG from $\mathcal{H}$, and similarly for triplex lines. We are going to show in Section 5 that, similarly to the LWF case, every strong equivalence class $\mathcal{H}$ has a unique largest element. We also present a special component merging procedure to get the largest element on basis of any graph in $\mathcal{H}$ there.

Strong equivalence is an equivalence relation that induces a partition of any AMP equivalence class $\mathbb{H}$ of CGs. We will denote the set of all strong equivalence classes included in $\mathbb{H}$ by $\overline{\mathbb{H}} \equiv \{\mathcal{H}; \mathcal{H} \subseteq \mathbb{H}\}$.

4.3 Flag Ordering

Interestingly, the relation (1) restricted to triplex edges defines a partial ordering *between* strong equivalence classes from $\overline{\mathbb{H}}$.

Definition 5 *(flag larger)*
Let $\mathbb{H}$ be an AMP equivalence class and $\mathcal{H}, \mathcal{G} \in \overline{\mathbb{H}}$. We say that $\mathcal{H}$ is flag larger *than $\mathcal{G}$ and write $\mathcal{H} \succeq \mathcal{G}$ if the following condition holds:*

$$\textit{whenever } a \longrightarrow b \textit{ is a triplex arrow in } \mathcal{H} \textit{ then } a \longrightarrow b \textit{ in } \mathcal{G}. \tag{3}$$

Observe that (3) and the fact $\mathcal{G}, \mathcal{H} \in \overline{\mathbb{H}}$ imply that

$$\text{whenever } a \multimap b \text{ is a triplex line in } \mathcal{G} \text{ then } a \multimap b \text{ in } \mathcal{H}. \tag{4}$$

Hence, $\mathcal{H} \succeq \mathcal{G} \succeq \mathcal{H}$ for $\mathcal{H}, \mathcal{G} \in \overline{\mathbb{H}}$ implies that $\mathcal{H}$ and $\mathcal{G}$ have the same triplex edges, that is, $\mathcal{G} = \mathcal{H}$. This allows one to see that the relation $\succeq$ is indeed an ordering on $\overline{\mathbb{H}}$. Another point is that (4) means that $\mathcal{H}$ has 'more' triplex lines than $\mathcal{G}$. In particular, if $\mathcal{H} \succeq \mathcal{G}$ then every flag in $\mathcal{G}$ is a flag of the same type in $\mathcal{H}$. For this reason, we will refer to the ordering defined by (3) as to the *flag ordering* of strong equivalence classes. In Figure 4 we illustrated this ordering by dashed lines. Note that there exists the smallest element with respect to flag ordering. It is a natural distinguished strong equivalence class within $\overline{\mathbb{H}}$ and, now, we prove its existence.

Proposition 6 *Given an AMP equivalence class $\mathbb{H}$, there exists a unique strong equivalence class $\mathcal{H}^{\downarrow} \in \overline{\mathbb{H}}$ such that $\mathcal{H} \succeq \mathcal{H}^{\downarrow}$ for all $\mathcal{H} \in \overline{\mathbb{H}}$.*

Proof As $\overline{\mathbb{H}}$ is finite and $\succeq$ is an ordering on $\overline{\mathbb{H}}$ it suffices to show that, for every $\mathcal{G},\mathcal{H} \in \overline{\mathbb{H}}$, there exists $\mathcal{F} \in \overline{\mathbb{H}}$ with $\mathcal{G} \succeq \mathcal{F}$ and $\mathcal{H} \succeq \mathcal{F}$. Choose $G \in \mathcal{G}$ and $H \in \mathcal{H}$ and construct a hybrid graph F with the same underlying graph as G (and H) in this way: $a \longrightarrow b$ in F iff either $a \longrightarrow b$ in G or [$a \longrightarrow\!\!\!\!- \; b$ in G and $a \longrightarrow b$ in H]. Lemma 4 in Andersson et al. (2001) says that F is a CG which is triplex equivalent to G (and H). Let $\mathcal{F}$ denote the strong equivalence class of CGs containing F. Thus, $\mathcal{F} \in \overline{\mathbb{H}}$ and the fact $G \geq F$ implies $\mathcal{G} \succeq \mathcal{F}$. The conclusion $\mathcal{H} \succeq \mathcal{F}$ can be verified directly: if $a \longrightarrow b$ is a triplex arrow in H (= in $\mathcal{H}$) then the fact that H and G are triplex equivalent implies that either $a \longrightarrow b$ in G or $a \longrightarrow\!\!\!\!- \; b$ in G which both gives $a \longrightarrow b$ in F (= in $\mathcal{F}$). ■

4.4 Deflagged Graphs and Essential Flags

Given an AMP equivalence class $\mathbb{H}$, the symbol $\mathcal{H}^{\downarrow}$ will be used to denote the least strong equivalence class in $\overline{\mathbb{H}}$ with respect to $\succeq$. The graphs in $\mathcal{H}^{\downarrow}$ will be called *maximally deflagged graphs* or, briefly, *deflagged graphs*.

In the example in Figure 4, both triplexes in the deflagged graphs are immoralities. However, in general, not all triplex edges in $\mathcal{H}^{\downarrow}$ have to be arrows. Some flags appear to be essential for the specification of the set $\mathbb{H}$ and, therefore, their lines are shared by all graphs from $\mathbb{H}$. An example is given in Figure 5 where a single graph, which has two flags, forms the whole AMP equivalence class.

Figure 5: An example of a pair of essential flags.

Definition 7 *(essential flag)*
Let $\mathbb{H}$ be an AMP equivalence class. If $a \longrightarrow b \longrightarrow\!\!\!\!- \; d$ is a flag in H for every $H \in \mathbb{H}$ then we say that it is an essential flag *in $\mathbb{H}$.*

Actually, deflagged graphs can equivalently be introduced as follows.

Proposition 8 *Given an AMP equivalence class $\mathbb{H}$, one has $G \in \mathcal{H}^{\downarrow}$ iff $G \in \mathbb{H}$ and every flag in G is an essential flag in $\mathbb{H}$.*

Proof To verify the necessity of the condition, consider a flag $a \longrightarrow b \longrightarrow\!\!\!\!- \; c$ in G and $H \in \mathcal{H} \in \overline{\mathbb{H}}$. Then the assumption $\mathcal{H} \succeq \mathcal{H}^{\downarrow} \ni G$ implies by (4) that the triplex line $b \longrightarrow\!\!\!\!- \; c$ in G is also in H. As $\langle\{a,c\},b\rangle$ is a triplex both in G and H it allows one to derive $a \longrightarrow b$ in H. Thus, $a \longrightarrow b \longrightarrow\!\!\!\!- \; c$ is a flag in every $H \in \mathbb{H}$.

For sufficiency, assume that $G \in \mathbb{H}$ only has essential flags. Let $\mathcal{G} \in \overline{\mathbb{H}}$ be the strong equivalence class containing G. We are to show that $\mathcal{H} \succeq \mathcal{G}$ for every $\mathcal{H} \in \overline{\mathbb{H}}$. Consider a triplex arrow $a \longrightarrow b$ in $\mathcal{H}$. It has to be a part of a triplex $\langle\{a,c\},b\rangle$. Since it has to be a triplex in G the only option which excludes $a \longrightarrow b$ in G is that $a \longrightarrow\!\!\!\!- \; b \longleftarrow c$ in G. However, then it is an essential flag in $\mathbb{H}$ and $a \longrightarrow\!\!\!\!- \; b \longleftarrow c$ in $\mathcal{H}$. This contradicts the assumption and one necessarily has $a \longrightarrow b$ in $\mathcal{G}$. ■

4.5 Largest Deflagged Graph

Let us summarize. AMP equivalence classes can effectively be handled by first considering their natural partition into strong equivalence classes (partially ordered by $\succeq$), and then by dealing with the CGs in every strong equivalence class (partially ordered by $\geq$). In this way, it is possible to identify unambiguously a graph in $\mathbb{H}$ by first considering the flag-smallest strong equivalence class and then by taking the largest graph within that class.

Definition 9 *(largest deflagged graph)*
The graph $H^{\downarrow}$ is the largest deflagged graph *of an AMP equivalence class $\mathbb{H}$ if*

(i) $H^{\downarrow} \in \mathcal{H}^{\downarrow}$,

(ii) $H^{\downarrow} \geq H$ *for all* $H \in \mathcal{H}^{\downarrow}$.

In Figure 4, the ordering of CGs within strong equivalence classes is illustrated by means of dotted lines. The largest deflagged graph is emphasized by means of vertices filled in.

Recall that the existence of the strong equivalence class $\mathcal{H}^{\downarrow}$ was proven in Proposition 6 whereas the existence and uniqueness of the largest CG in $\mathcal{H}^{\downarrow}$ is shown in Section 5. Furthermore, in Section 6, we provide a deflagging procedure which, starting from any CG G in an AMP equivalence class $\mathbb{H}$, returns a CG $\widehat{G}$ in $\mathcal{H}^{\downarrow}$. Then a component merging procedure from Section 5 can be applied to $\widehat{G}$ to get the largest deflagged graph $H^{\downarrow}$.

5. Strong Equivalence

This section is devoted to basic results on strong equivalence of CGs. These results are analogous to the results on LWF Markov equivalence recalled in Section 3. More specifically, we prove the existence of the largest CG within each strong equivalence class, introduce the respective elementary operation ascribing a larger strongly equivalent CG to a CG, and show that the largest CG in a strong equivalence class is attainable by this operation.

5.1 Largest Chain Graph in a Strong Equivalence Class

In this subsection we show the existence of the largest CG within a strong equivalence class. The first step for this is a direct construction of the supremum of two CGs with a shared underlying graph with respect to the ordering $H \geq G$ defined by (1). Note that the construction was already mentioned without further details in Frydenberg (1990). The construction utilizes the following auxiliary concept.

Definition 10 *(cyclic arrow)*
Given a hybrid graph H, we say that an arrow $a \longrightarrow b$ in H is a cyclic arrow *in H if $b \in \mathrm{an}_H(a)$. An equivalent formulation is that there exists a semi-directed cycle in H containing $a \longrightarrow b$.*

Lemma 11 *Let us consider the class E of all CGs over N with a prescribed underlying graph E, ordered by the relation $\geq$ defined by (1). Then every pair of graphs G and H from E has the supremum $G \vee H$ in $(E, \geq)$. It can be obtained directly in two steps.*

1. *Define a hybrid graph $G \cup H$ over N as follows*

$$a \longrightarrow b \text{ in } G \cup H \text{ iff both } a \longrightarrow b \text{ in } G \text{ and } a \longrightarrow b \text{ in } H\,,$$

and a —— b in $G \cup H$ for remaining edges in E.

2. *Replace all cyclic arrows in $G \cup H$ with lines and obtain $G \vee H$.*

Proof It is easy to see that $(E, \geq)$ is a partially ordered set. We need to show that $G \vee H \in E$, $G \vee H \geq G$, $G \vee H \geq H$ and, whenever there is $F \in E$ with $F \geq G, H$ then $F \geq G \vee H$.

The fact that $G \vee H$ is a CG was proven as Consequence 2.5 in Volf and Studený (1999). Hence, it is clear that $G \vee H \in E$ and that $G \vee H$ is larger than both G and H.

To show that $F \geq G \vee H$ for $F \in E$ with $F \geq G, H$, consider an arrow $a \longrightarrow b$ in F in order to verify $a \longrightarrow b$ in $G \vee H$. Since $a \longrightarrow b$ in $G \cup H$, it suffices to show $b \notin \mathrm{an}_{G \cup H}(a)$. Suppose for contradiction that there exists a descending path $\rho : b = c_1, \ldots, c_n = a$, $n \geq 2$ in $G \cup H$. There is no $1 \leq i \leq n-1$ with $c_i \longleftarrow c_{i+1}$ in F, as otherwise $c_i \longleftarrow c_{i+1}$ in $G \cup H$. Thus, ρ is a descending path in F which contradicts the assumption that F is a CG. ■

The preceding construction can be utilized to prove that every strong equivalence class of CGs is a join semi-lattice with respect to $\geq$.

Proposition 12 *Let G and H be strongly equivalent CGs over N. Then their supremum $G \vee H$ is strongly equivalent to them as well.*

Because the proof is technical, it is moved to the Appendix. Proposition 12 has the following consequence.

Corollary 13 *Given a strong equivalence class $\mathcal{G}$ of CGs over N, there exists $G^{\dagger} \in \mathcal{G}$ which is the largest CG in $\mathcal{G}$.*

Proof Since $\mathcal{G}$ is a finite set, one can apply Proposition 12 repeatedly to get the supremum of all graphs in $\mathcal{G}$. Of course, it is the largest CG in $\mathcal{G}$. ■

5.2 Legal Merging of Components

In this subsection we introduce an elementary operation that produces a strongly equivalent CG when applied to a CG. Here is the definition.

Definition 14 *(legal merging of components)*
Let (U, L) be a pair of components in a CG G that defines a meta-arrow. We say that merging of components U and L is legal *(in G) if the following three conditions hold:*

[i] $K \equiv \mathrm{pa}_G(L) \cap U$ *is a complete set in G,*

[ii] $\forall b \in K \quad \mathrm{pa}_G(L) \setminus U = \mathrm{pa}_G(b)$,

[iii] *for every $d \in L$ one has* $\mathrm{pa}_G(L) = \mathrm{pa}_G(d)$.

Evidently, the conditions [i]-[iii] imply the conditions (i)-(ii) from Definition 3. In brief, every legal merging (of components in a CG) is feasible. In Figure 3, (M1) is an example of feasible merging that is not legal whereas (M2) is an example of legal merging. If G is a CG without flags then the condition [iii] is always fulfilled and [ii] takes a simpler form:

[$\widetilde{\text{ii}}$] $\text{pa}_G(L) \setminus U = \text{pa}_G(U)$.

Thus, the operation from Definition 14 generalizes the operation of legal merging of components (of a CG without flags) from Studený (2004). The requirement [i]+[$\widetilde{\text{ii}}$] also coincides with the condition from Roverato (2005) demanding that the arrowhead of the meta-arrow $U \rightrightarrows L$ is strongly insubstantial.

Proposition 15 *Let G be a CG over N, and (U,L) be a pair of its components which defines a meta-arrow. Then the conditions from Definition 14 are satisfied iff the graph G' obtained by merging of components U and L is a CG strongly equivalent to G; of course, it is (strictly) larger than G.*

The proof is moved to the Appendix. Note that one has to replace the whole collection of arrows between components with lines; otherwise the obtained graph would not be a CG. This is the reason why legal merging is indeed an elementary operation yielding a larger and strongly equivalent CG.

5.3 Component Merging Procedure

An important fact is that the largest CG in a strong equivalence class $\mathcal{G}$ can be obtained from any CG in $\mathcal{G}$ by consecutive application of the operation of legal merging of components. Actually, we show the following, formally stronger, result.

Proposition 16 *Let G and H be strongly equivalent CGs over N such that $H \geq G$. Then there exists a finite sequence $G \equiv F_1, \ldots, F_m \equiv H$, $m \geq 1$ of CGs over N such that, for every $i = 1, \ldots, m-1$, the graph F_{i+1} is obtained from F_i by legal merging of components.*

The proof is technical and it is moved to the Appendix. Proposition 16 has the following consequence.

Corollary 17 *Given a strong equivalence class $\mathcal{G}$ of CGs over N and $G \in \mathcal{G}$, the largest CG $G^{\dagger}$ in $\mathcal{G}$ is attainable from G by a series of legal mergings.*

Proof We simply put $H = G^{\dagger}$ in Proposition 16. ■

6. Deflagging Procedure

In this section we describe a procedure to construct a deflagged graph $\widehat{G}$ starting from any CG G in the respective AMP equivalence class $\mathbb{H}$. We proceed as follows. First, we introduce a *labeling algorithm* that assigns some labels to endings of lines in G. Second, we introduce a *directing algorithm* which, on the basis of those labels, replaces certain lines in G with arrows. In this way, we get a CG which is both triplex equivalent to G and flag-smaller than G. Finally, we provide a deflagging procedure which consists of repeated application of these two algorithms. We show that the result is a deflagged graph.

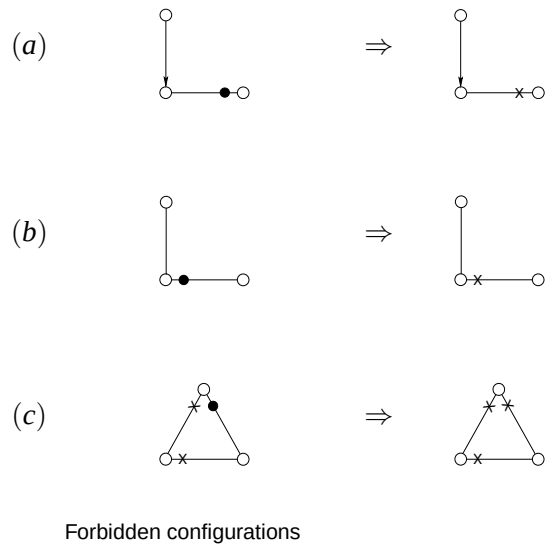


Figure 6: Three blocking rules from the labeling algorithm.

6.1 Labeling Algorithm

Let $G = (N, \mathcal{A}, \mathcal{L})$ be a CG. A *labeled graph* $G^\ell = (N, \mathcal{A}, \mathcal{L}^\ell)$ is a graph obtained by ascribing a pair of labels to every line $\{a,b\} \in \mathcal{L}$. The labels on a line a —— b correspond to endings of the line: one of them is associated with a and the other with b. We use two different kinds of labels: a blocking label denoted by a cross, 'x', and a label denoted by a dot, '•', to be read as 'free'. Thus, if the blocking label is associated with a on a —— b then we will say that the line is *blocked at* a and write $a \,\text{—}\!\!\times\!\!\text{—}\, b$ in G^ℓ. On the other hand, the notation $a \,\text{—}\!\!\bullet\!\!\text{—}\, b$ in G^ℓ will mean that the line is *free at* a. The intuition behind the terminology is as follows. A blocked ending at a node a will mean that the line cannot be replaced with an arrow directed to a, for otherwise we would get a graph outside $\mathbb{H}$. A free ending at a will mean that no such conclusion has been derived so far.

Consequently, a labeled CG has three types of lines: two symmetric forms $\times\!\!-\!\!\times$ and $\bullet\!\!-\!\!\bullet$, and an asymmetric form $\times\!\!-\!\!\bullet$. Let us emphasize that we only consider labeled graphs in which all lines have both endings labeled. However, in our notation, labels need not be explicitly indicated. For instance, the notation a ——$\times$ b in G^ℓ will mean that either $a \times\!\!-\!\!\times b$ in G^ℓ or $a \bullet\!\!-\!\!\times b$ in G^ℓ.

The *labeling algorithm*, whose pseudo-code is given in Algorithm 1, produces a special labeled version G^ℓ of a given CG G. Initially, all lines are replaced with labeled lines with free endings. Then, three *blocking rules*, illustrated in Figure 6, are repeatedly applied until they are not applicable. Each blocking rule modifies just one ending of one line: a free ending is blocked. In this way, we get a labeled CG in which no *forbidden configuration* (see Figure 6) is present. The labelling algorithm is the first step of the overall deflagging procedure and in the following step some lines of G are replaced by arrows; thus, the reader can possibly understand that the three forbidden configurations actually correspond to three unwanted operations: (a) corresponds to cancellation of a triplex, (b) to creation of a triplex and (c) to creation of a semi-directed cycle (of the length 3).

Algorithm 1 Pseudo-code for the `LabelingAlgorithm`(G).

1: input a CG $G = (N, \mathcal{A}, \mathcal{L})$
2: put $i = 0$
3: initialize $G_i^\ell = (N, \mathcal{A}, \mathcal{L}^\ell)$ by replacing every line a —— b in G by $a \circ\!\!-\!\!\circ b$ in G_i^ℓ
4: **while** at least one forbidden configuration is present in G_i^ℓ **do**
5: $\quad i = i + 1$
6: $\quad G_i^\ell =$ modify G_{i-1}^ℓ by applying one of the following rules (see also Figure 6):
(a) if $a \longrightarrow b \circ\!\!-\!\!\circ c$ in G_{i-1}^ℓ and a and c are not adjacent then $b \circ\!\!-\!\!* c$ in G_i^ℓ
(b) if a —— $b \circ\!\!-\!\!\circ c$ in G_{i-1}^ℓ and a and c are not adjacent then $b *\!\!-\!\!\circ c$ in G_i^ℓ
(c) if $a \circ\!\!-\!\!* b \circ\!\!-\!\!* c \circ\!\!-\!\!\circ a$ in G_{i-1}^ℓ then $c *\!\!-\!\!\circ a$ in G_i^ℓ
7: **end while**
8: return $G^\ell = G_i^\ell$

The point is that the result of the labelling algorithm is invariant with respect to the order in which the blocking rules are applied.

Proposition 18 *For any CG G, the labeled graph G^ℓ=*`LabelingAlgorithm`(G) *is unique. This means that the output of the labeling algorithm does not depend on the ordering in which the three blocking rules are applied.*

The proof can be found in the Appendix. In the rest of the paper, G^ℓ will always denote the labeled version of G resulting from the application of Algorithm 1. An example of application of the labeling algorithm is given in Figure 7.

Note that Algorithm 1 is specified so that just one single label is changed in one iteration. This is useful in the proofs of the results of this section, but may be inefficient in practice. A more efficient implementation of the procedure can be achieved by applying the rules (a) and (b) first in a multi-step, and then only applying the rule (c) iteratively. This follows from Proposition 18 and the fact that the application of the rules (a) and (b) does not depend on the result of previous iterations of Algorithm 1.

6.2 Directing Algorithm

The *directing algorithm*, described in Algorithm 2, is the second building block of the deflagging procedure. It replaces some (labeled) lines with arrows in order to possibly reduce the number of flags in the original CG. More precisely, every line of the form $a *\!\!-\!\!\circ b$ is replaced with the arrow $a \longrightarrow b$ and then the labels on other lines are removed.

Algorithm 2 Pseudo-code for the `DirectingAlgorithm`(G^ℓ).

1: input a labeled CG $G^\ell = (N, \mathcal{A}, \mathcal{L}^\ell)$
2: $G_*^\ell =$ modify G^ℓ by applying the following rule:

$$a *\!\!-\!\!\circ b \text{ in } G^\ell \quad \Rightarrow \quad a \longrightarrow b \text{ in } G_*^\ell$$

3: $G' =$ unlabeled version of G_*^ℓ
4: return G'

We show that if the directing algorithm is applied to the result of the labeling algorithm then an AMP equivalent graph is obtained.

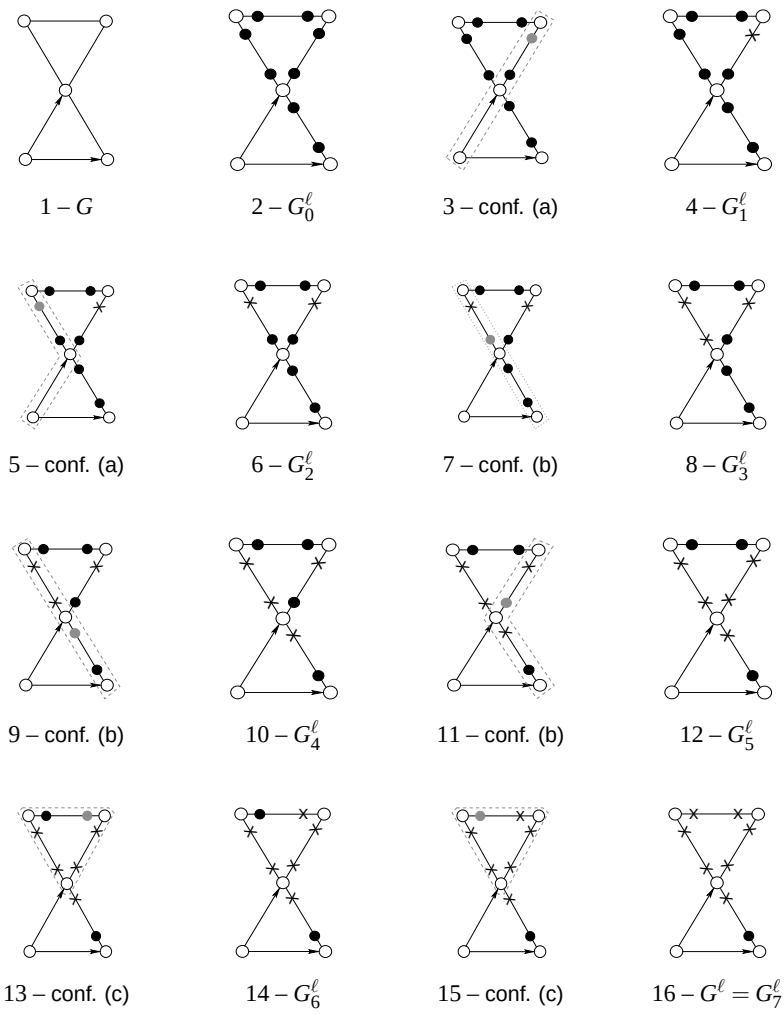


Figure 7: An example of the application of the labeling algorithm to be read following the numbering. Initially, all lines of G are replaced with labeled lines with free endings. Then, in every pair of successive pictures, a forbidden configuration is highlighted and the corresponding rule is applied.

Theorem 19 *Let G be a CG, G^ℓ denote the labeled graph obtained from G by Algorithm 1, and G' the graph resulting from G^ℓ by Algorithm 2. Then G' is a CG which is triplex equivalent to G.*

The proof is relatively long and we have placed it in the Appendix. Clearly, one has $G \geq G'$ and, hence, $\mathcal{G} \succeq \mathcal{G}'$ for the respective equivalence classes. Moreover, one has $G \neq G'$ unless no line is replaced with an arrow in the directing phase. An example of the application of the directing algorithm will be shown in the next section.

6.3 Overall Procedure

The application of the above algorithms to a CG G produces a graph G' in the same AMP equivalence class such that $G \geq G'$. However, G' still need not be a maximally deflagged graph and one can then apply the same procedure to G'. In Algorithm 3, we provide the pseudo-code of the overall deflagging procedure which consists in repeated application of both algorithms until no line is replaced with an arrow during the directing phase. Its result will be denoted by $\widehat{G}$.

Algorithm 3 Pseudo-code for the `DeflaggingProcedure`(G).

input $G = (N, \mathcal{A}, \mathcal{L})$
$j = 0$
initialize $\widehat{G}_j = G$
repeat
  $j = j + 1$
  $\widehat{G}^\ell_{j-1} =$ `LabelingAlgorithm`$(\widehat{G}_{j-1})$
  $\widehat{G}_j =$ `DirectingAlgorithm`$(\widehat{G}^\ell_{j-1})$
until $\widehat{G}_j$ is equal to $\widehat{G}_{j-1}$
return $\widehat{G} = \widehat{G}_j$

Since G has a finite number of lines, the procedure will return a result in finitely many steps. An example of the application of the deflagging algorithm is given in Figure 8. Note that, in this example, $\widehat{G}$ is already the largest deflagged graph from Figure 4; however, this is not true in general.

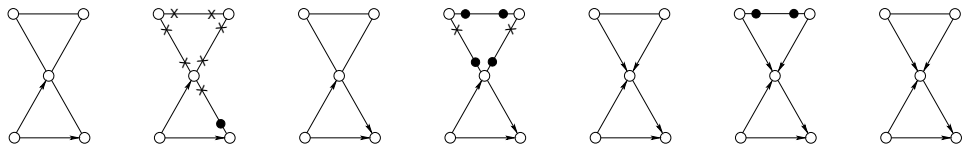


Figure 8: An example of the application of the deflagging procedure, where G is the top left graph in Figure 4. Note that the first application of the labeling algorithm, to obtain G^ℓ_0 from G^ℓ, is detailed in Figure 7.

We are to show that $\widehat{G}$ is a deflagged graph, that is, $\widehat{G}$ in $\mathcal{H}^\downarrow$ where $\mathcal{H}^\downarrow$ is the class of deflagged graphs in the respective AMP equivalence class. It follows from Algorithm 3 that $\widehat{G}$ is such that the

directing algorithm does not direct any line if applied to the labeled version of $\widehat{G}$. This means, every line in $\widehat{G}^{\ell}$ is either of the type $\bullet\!\!-\!\!\bullet$ or of the type $*\!\!-\!\!*$.

Proposition 20 *Let G be a CG such that there is no line of asymmetric form $*\!\!-\!\!\bullet$ in its labeled version G^{ℓ}. Then, every line a —— b in G such that a $*\!\!-\!\!*$ b in G^{ℓ} is a line in every CG F which is triplex equivalent to G.*

The proof is again postponed to the Appendix. Proposition 20 is not valid if the assumption on G is omitted. A counterexample is given in Figure 8 where $G = \widehat{G}_0$ and $F = \widehat{G}$. A consequence of Proposition 20 is that every flag in $\widehat{G}$ is an essential flag.

Corollary 21 *Given a CG G, the graph $\widehat{G} = \texttt{DeflaggingProcedure}(G)$ is a deflagged graph, formally $\widehat{G}$ in $\mathcal{H}^{\downarrow}$.*

Proof By Theorem 19, $\widehat{G}$ belongs to the same AMP equivalence class $\mathbb{H}$ as G. Owing to Proposition 8, we need to show that if $a \longrightarrow b$ —— d is a flag in $\widehat{G}$ then it is an essential flag. By the blocking rule (a) $a \longrightarrow b$ —— d in $\widehat{G}$ implies $a \longrightarrow b \;*\!\!-\!\!*\; d$ in $\widehat{G}^{\ell}$. Since there are no lines of the form $\bullet\!\!-\!\!*$ in $\widehat{G}^{\ell}$, it necessitates $a \longrightarrow b \;*\!\!-\!\!*\; d$ in $\widehat{G}^{\ell}$. It follows from Proposition 20 that b —— d in H for every $H \in \mathbb{H}$. As $a \longrightarrow b$ —— d has to correspond to a triplex in H, one can conclude that $a \longrightarrow b$ —— d in H. ■

Note that the arguments in the proof above actually imply that a simple sufficient condition for a CG to be deflagged is that its labelled version has no line of asymmetric form.

7. Conclusions

This paper is devoted to the problem of choosing a graphical representative of the statistical model ascribed to a CG under AMP interpretation. As a matter of fact, any CG from the respective AMP Markov equivalence class provides a graphical representative of the corresponding model. However, a representative only makes sense if it complies with some properties that uniquely identify it within each class. Furthermore, in the framework of structural learning, the usefulness of a graphical representative is related to the availability of procedures which can be practically dealt with. That means, for instance, that an implementable construction procedure to obtain the representative (on the basis of any other graph in the Markov equivalence class) should be at our disposal.

Nevertheless, from the point of view of interpretation, a representative should be chosen on the basis of the information carried with respect to the corresponding statistical model. Hereafter, we address the issue of the information contained in the largest deflagged graph, which is the representative for an AMP chain graph model we have proposed.

Andersson et al. (2001, Theorem 4) showed that, for a CG H, the AMP and the LWF Markov properties coincide iff H has no flags. Thus, if there exists a CG without flags in $\mathbb{H}$ then formal distinction between the two Markov properties is not necessary. In this case, all the results derived in the LWF case can be applied. For instance, the useful factorization of conditional densities into 'potentials' given by Frydenberg (1990, Theorem 4.1(iii)) can be applied in the AMP case only with respect to CGs without flags. Clearly, there is a strong connection between the set of CGs without flags and the set $\mathcal{H}^{\downarrow}$ of deflagged graphs. More specifically, $\mathbb{H}$ has a CG without flags iff

there are no essential flags in $\mathbb{H}$. In this case, the class of deflagged graphs $\mathcal{H}^{\downarrow}$ is just the class of CGs without flags in $\mathbb{H}$. Conversely, if there exists some essential flag in $\mathcal{H}^{\downarrow}$ then one can conclude that there is no CG in $\mathbb{H}$ for which the two Markov properties coincide. Because deflagged graphs only contain essential flags, they eliminate the ambiguity resulting from the non-unique graphical representation of triplexes, and allow an immediate comparison with the LWF case.

The above reasons justify our restriction to the class of deflagged graphs. Now we justify the choice of the largest deflagged graph in $\mathcal{H}^{\downarrow}$. If $\mathcal{H}^{\downarrow}$ contains no flags then $H^{\downarrow}$ is the largest CG without flags in $\mathbb{H}$. Thus, if $\mathbb{H}$ contains an undirected graph then the largest deflagged graph $H^{\downarrow}$ coincides with that undirected graph. Analogously, if $\mathbb{H}$ contains an acyclic directed graph D then $H^{\downarrow}$ coincides with the essential graph D^{*} for D (Andersson et al., 1997; Studený, 2004; Roverato, 2005). We remark that the AMP essential graph H^{*} proposed by Andersson et al. (2001) is a deflagged graph (see Andersson and Perlman, 2006, Lemma 3.2(a)) so that $H^{*} \leq H^{\downarrow}$. Nevertheless, in general, the largest deflagged graph $H^{\downarrow}$ is different from the AMP essential graph H^{*}: for instance, if $\mathbb{H}$ contains an undirected graph then H^{*} may even have some arrows (see Andersson et al., 2001, Figure 14).

Another issue related to the problem of representative choice is the topic of causal discovery in CGs (see Section 11.2 of Lauritzen, 2001). This is a controversial topic (see Section 3 of Dawid, 2002, for more discussion). The disputable question is whether one can identify some causal relationships between variables on the basis of data. Nevertheless, what we think that what is generally accepted in the field of causal discovery is the following proposition:

> If data are "generated" from a distribution which is "faithful" with respect to a CG and if an arrow $a \longrightarrow b$ is *not* invariant across the respective Markov equivalence class, then one *cannot* reveal possible causal relationship from a to b on basis of data.

In short, one cannot make causal discovery between a and b if there is an *undirected* edge between a and b in at least one of the chain graphs from the Markov equivalence class, or if there are two chain graphs such that $a \longrightarrow b$ in the one of them first and $b \longrightarrow a$ in the latter one. On the other hand, if an arrow $a \longrightarrow b$ is invariant across the respective Markov equivalence class then causal discovery *could* be possible. Consequently, from the point of view of causal discovery in chain graphs, a good representative of a Markov equivalence class should indicate that the corresponding edge is not an invariant arrow by the presence of a line. Standard representatives in the LWF case, such as the largest CGs (Studený, 1997), the essential graphs for acyclic directed graphs (Andersson et al., 1997), and the $\mathcal{B}$-essential graphs (Roverato and La Rocca, 2006), are fully informative from this point of view because they have the largest number of lines and, furthermore, they contain an arrow if and only if it is invariant. As the examples in Figures 2 and 4 show, a CG with this property may not exist in an AMP equivalence class and therefore both the AMP essential graph and the largest deflagged graph may contain some arrows that are not invariant. However, the largest deflagged graph is more informative than the AMP essential graph because it is a larger chain graph and, therefore, it has more lines.

We have not mentioned this explicitly but, in this paper, we have actually provided an algorithmic characterization of the largest deflagged graphs. More specifically, a CG G is the largest deflagged graph iff it is again obtained by the consecutive application of two procedures: the deflagging procedure is applied to G and the component merging procedure to its result $\widehat{G}$.

The results of the paper also lead to some natural open problems. For instance, we would like to know whether the converse of Proposition 20 is valid. More specifically, does the deflagging procedure identify all essential lines in $\mathbb{H}$ as double-blocked lines? Further conjecture is that the AMP essential graph is obtained if the deflagging procedure is applied to the largest deflagged graph. Another issue is as follows. We know that both LWF and AMP Markov equivalence are associated to Markov properties for CGs. Is there any Markov property for CGs which gives rise to the strong equivalence of CGs?

Acknowledgements

We are indebted to Nanny Wermuth who invited both of us to a workshop held in Wiesbaden in September 2002. We started there a discussion, which resulted in this paper. We also thank Michael Perlman for useful comments during the Barcelona meeting in 2004 and the anonymous referees for useful suggestions. Financial support to the fist author has been provided by Miur, PRIN03 and PRIN05 n. 134079. The research of the second author has been supported by GAČR n. 201/04/0393.

Appendix A. Proofs

Proof of Proposition 12

Throughout the proof we assume that G and H are strongly equivalent CGs. Let $G \cup H$ and $G \vee H$ denote the graphs introduced in Lemma 11. We start with an auxiliary observation.

Fact 1 *Let $d_0 \longrightarrow d_1$ be a cyclic arrow in $G \cup H$ and $\rho : d_0, d_1, \ldots, d_m \equiv d_0$, $m \geq 3$ a semi-directed cycle in $G \cup H$ containing it which cannot be shortened (to a semi-directed cycle in $G \cup H$ containing $d_0 \longrightarrow d_1$ of the length $l < m$). Then $d_2 \longrightarrow d_1$ in one of the graphs G and H while $d_0 \longrightarrow d_2$ in the other graph.*

Proof Since G is a CG, there exist $2 \leq j \leq m$ with $d_{j-1} \longleftarrow d_j$ in G and the same conclusion holds for H. Let us put

$$s = \min\{\, 2 \leq j \leq m;\, d_{j-1} \longleftarrow d_j \text{ either in } G \text{ or in } H \,\}.$$

Let us, without loss of generality, assume that $d_{s-1} \longleftarrow d_s$ in G. Then $d_0, \ldots, d_{s-1}$ is a descending path G. Moreover, observe that $d_1, \ldots, d_s$ is necessarily a descending path in the other graph, namely in H. This implies $s < m$ for otherwise ρ is a semi-directed cycle in a CG H.

The next step is to verify that $[d_{s-2}, d_s]$ is an edge in $G \cup H$. This is because otherwise $d_s \longrightarrow d_{s-1} \longleftarrow d_{s-2}$ is an immorality in G or $d_s \longrightarrow d_{s-1} \longrightarrow d_{s-2}$ is a flag in G, which, by strong equivalence of G and H, implies that $d_s \longrightarrow d_{s-1}$ in H and this contradicts the assumption that ρ is a semi-directed cycle in $G \cup H$.

Since H is a CG and d_{s-2}, d_{s-1}, d_s a descending path in H, one has either $d_s \longleftarrow d_{s-2}$ or $d_s \longrightarrow d_{s-2}$ in H, and, therefore, in $G \cup H$.

Thus, necessarily $s = 2$; otherwise ρ could be shortened in $G \cup H$ by the edge $[d_{s-2}, d_s]$ to get a shorter semi-directed cycle containing $d_0 \longrightarrow d_1$ which would contradict its definition. Thus, $d_2 \longrightarrow d_1$ in G. The facts that H is a CG, $[d_0, d_2] = [d_{s-2}, d_s]$ is an edge in H, $d_0 \longrightarrow d_1$ in H and either $d_1 \longrightarrow d_2$ or $d_1 \longrightarrow d_2$ in H imply that $d_0 \longrightarrow d_2$ in H. ■

Fact 2 *There is no cyclic arrow $a \longrightarrow c$ in $G \cup H$ which belongs either to an immorality $a \longrightarrow c \longleftarrow b$ or to a flag $a \longrightarrow c \longrightarrow\!\!\!\!\!- \; b$ in $G \cup H$.*

Proof For a contradiction, suppose that at least one such cyclic arrow exists. Choose a semi-directed cycle $\rho : d_0, d_1, \ldots, d_m \equiv d_0$, $m \geq 3$ in $G \cup H$ of shortest possible length among all semi-directed cycles containing an arrow of this kind. Assume that $d_0 = a \longrightarrow c = d_1$ is that arrow in $G \cup H$ and, using Fact 1, observe that $d_2 \longrightarrow d_1$ in one of the graph, say in G, while $d_0 \longrightarrow d_2$ in the other graph H.

Consider the induced subgraph over $\{a, c, b\}$ mentioned in the formulation of Fact 2. As $[a, b]$ is not an edge in $G \cup H$ whereas $[d_0, d_2] = [a, d_2]$ is an edge in H, one has $d_2 \neq b$. Observe that $c \longleftarrow b$ or $c \longrightarrow\!\!\!\!\!- \; b$ in G. Indeed, otherwise $c \longrightarrow b$ in G implies $\neg(c \longrightarrow b \text{ in } H)$ by the assumption of Fact 2, and H has either an immorality $a \longrightarrow c \longleftarrow b$ or a flag $a \longrightarrow c \longrightarrow\!\!\!\!\!- \; b$. By strong equivalence of G and H, G has the same induced subgraph for $\{a, c, b\}$, which contradicts the fact $c \longrightarrow b$ in G. By interchange of G and H derive that $c \longleftarrow b$ or $c \longrightarrow\!\!\!\!\!- \; b$ in H as well.

This allows one to see that $[b, d_2]$ is an edge in $G \cup H$ as otherwise the induced subgraph of G for $\{d_2, d_1 = c, b\}$ having $d_2 \longrightarrow d_1$ coincides, by strong equivalence of G and H, with the subgraph of H and the conclusion $d_2 \longrightarrow d_1$ in H contradicts the assumption that ρ is a semi-directed cycle in $G \cup H$. Since b, c, d_2 is a descending path in H one has either $b \longrightarrow d_2$ or $b \longrightarrow\!\!\!\!\!- \; d_2$ in H.

Thus, H has either an immorality $d_0 \longrightarrow d_2 \longleftarrow b$ or a flag $d_0 \longrightarrow d_2 \longrightarrow\!\!\!\!\!- \; b$. Since G and H are strongly equivalent, G has the same induced subgraph for $\{d_0, d_2, b\}$. Of course, the same conclusion holds for $G \cup H$ and $d_0 \longrightarrow d_2$ is an arrow in $G \cup H$ belonging to a triplex.

Hence, it is impossible that $m > 3$ as otherwise ρ can be shortened to $d_0, d_2, \ldots, d_m = d_0$ by a cyclic arrow $d_0 \longrightarrow d_2$ of the considered type which contradicts its definition. However, if $m = 3$ then the fact $d_3 \equiv d_0 \longrightarrow d_2$ in $G \cup H$ contradicts the assumption that ρ is a semi-directed cycle in $G \cup H$. ■

Observe easily by contradiction that if G and H are strongly equivalent then

[d] if $a \longrightarrow c$ both in G and in H then an induced subgraph $a \longrightarrow c \longrightarrow b$ occurs in H iff it occurs in G.

This observation is used in the proof of the following fact and also later.

Fact 3 *There is no cyclic arrow $c \longrightarrow b$ in $G \cup H$ which belongs to an induced subgraph $a \longrightarrow c \longrightarrow b$ in $G \cup H$.*

Proof For a contradiction, suppose that such an arrow exists. Choose a semi-directed cycle $\rho : d_0, d_1, \ldots, d_m \equiv d_0$, $m \geq 3$ in $G \cup H$ of shortest possible length among all semi-directed cycles containing an arrow of this kind. More specifically, assume that $d_0 = c \longrightarrow b = d_1$ in $G \cup H$. By Fact 1 observe that one can assume $d_2 \longrightarrow d_1$ in G and $d_0 \longrightarrow d_2$ in H. One has either $d_2 \longleftarrow d_1$ or $d_2 \longrightarrow\!\!\!\!\!- \; d_1$ in H for otherwise $d_2 \longrightarrow d_1$ in $G \cup H$ contradicts the assumption that ρ is a semi-directed cycle. As $a \longrightarrow c$ in H whereas $d_2 \longleftarrow d_0 = c$ in H, one has $a \neq d_2$.

Observe, by contradiction, that $[a, d_2]$ is not an edge in $G \cup H$. Indeed, otherwise $a \longrightarrow c = d_0 \longrightarrow d_2$ in a CG H implies $a \longrightarrow d_2$ in H and H has either an immorality $a \longrightarrow d_2 \longleftarrow d_1$ or a flag $a \longrightarrow d_2 \longrightarrow\!\!\!\!\!- \; d_1$. Thus, G has the same subgraph for $\{a, d_2, d_1\}$ which contradicts the fact $d_2 \longrightarrow d_1$ in G.

Thus, H has an induced subgraph $a \longrightarrow c = d_0 \longrightarrow d_2$, which implies, by the condition [d] mentioned above Fact 3, that G has the same induced subgraph. In particular, $G \cup H$ has this induced subgraph as well. Thus, necessarily $m \leq 3$ for otherwise ρ can be shortened in $G \cup H$ by $d_0 \longrightarrow d_2$ to get a shorter cycle of the required type. If $m = 3$ then $d_3 = d_0 \longrightarrow d_2$ in $G \cup H$ implies a contradictory conclusion that ρ is not a semi-directed cycle in $G \cup H$. ■

Now, the proof of Proposition 12 follows directly from Lemma 11 and Facts 2 and 3. Since G and H are strongly equivalent an immorality or a flag in G occurs also in H and, therefore, in $G \cup H$. By Fact 2 it is preserved in $G \vee H$. Conversely, if $a \longrightarrow c \longleftarrow b$ is an immorality in $G \vee H$ then it is also in G. If $a \longrightarrow c \longrightarrow b$ is a flag in $G \vee H$ then $a \longrightarrow c$ both in G and in H. The option $c \longleftarrow b$ in one of the graphs G and H is excluded because then the graph has an immorality $a \longrightarrow c \longleftarrow b$, which is saved in $G \vee H$. If $c \longrightarrow b$ in both graphs then $G \cup H$ has an induced subgraph $a \longrightarrow c \longrightarrow b$. By Fact 3 the arrow $c \longrightarrow b$ remains in $G \vee H$ which contradicts the assumption. Thus, $c \longrightarrow\!\!\!\!\!- \; b$ either in G or in H and this implies, by their strong equivalence, that the flag $a \longrightarrow c \longrightarrow\!\!\!\!\!- \; b$ is in G.

Proof of Proposition 15

This proposition is analogous to the result on LWF equivalence and feasible merging given in Theorem 8 of Roverato (2005). It says this:

> Given a CG G and a meta-arrow $U \rightrightarrows L$ in G, the conditions (i) and (ii) from Definition 3 form together a necessary and sufficient condition for the graph G' obtained by merging U and L to be a CG which is complex equivalent to G.

In fact, we utilize this result in our proof of Proposition 15. Recall that the condition [i] from Definition 14 is identical to the condition (i) from Definition 3 and the condition [ii] from Definition 14 is stronger than (ii) from Definition 3.

Proof First, we are going to verify the necessity of the conditions [i]-[iii]. Since strong equivalence of CGs implies their complex equivalence the necessity of conditions (i)-(ii) follows from Theorem 8 in Roverato (2005). The conditions [i] and (i) are identical, but [ii] is stronger than (ii). Indeed, [ii] requires equality of sets $\mathrm{pa}_G(L) \setminus U$ and $\mathrm{pa}_G(b)$ for every $b \in K$ whereas (ii) only requires $\mathrm{pa}_G(L) \setminus U \subseteq \mathrm{pa}_G(b)$.

Thus, to verify [ii] it suffices to show

- $\forall b \in K \quad \mathrm{pa}_G(b) \subseteq \mathrm{pa}_G(L) \setminus U$.

Suppose for contradiction that $b \in K$ and $a \in \mathrm{pa}_G(b)$ exists with $a \notin \mathrm{pa}_G(L) \setminus U$. Then $d \in L$ exists such that $b \longrightarrow d$ in G. Of course, $a \neq d$ and, since G is a CG, the options $a \longleftarrow d$ and $a \longrightarrow\!\!\!\!\!- \; d$ in G cannot occur. The option $a \longrightarrow d$ is excluded by the assumption $a \notin \mathrm{pa}_G(L) \setminus U$. If $[a,d]$ is not an edge in G then G has an induced subgraph $a \longrightarrow b \longrightarrow d$ while G' has a flag $a \longrightarrow b \longrightarrow\!\!\!\!\!- \; d$ which contradicts the assumption that they are strongly equivalent.

The next step is to verify the necessity of the condition

[$\widetilde{\text{iii}}$] for every $d \in L$ one has $\mathrm{pa}_G(L) \subseteq \mathrm{pa}_G(d)$,

which is an equivalent formulation of [iii]. Let us fix $d \in L$. Given $b \in \mathrm{pa}_G(L)$, to show that $b \longrightarrow d$ in G two cases can be distinguished.

- $b \in U$, that is, $b \in K$.
  Then there exists $g \in L$ with $b \longrightarrow g$ in G. If $g \neq d$ then one can consider a path $b \longrightarrow g \multimap \ldots \multimap d$ in G and its shortening ρ which cannot be shortened any more. If $[b,d]$ is not an edge in G then G has a flag $b \longrightarrow e \multimap f$ composed of nodes of ρ. This contradicts the assumption that G and G' are strongly equivalent since one has $b \multimap e$ in G' by definition of merging. Thus, $[b,d]$ is an edge and $b \longrightarrow d$ in G since G is a CG.

- $b \in \mathrm{pa}_G(L) \setminus U$.
  Observe that $a \in K$ exists by Definition 1 and $b \longrightarrow a$ follows from (ii). Moreover, one has $a \longrightarrow d$ in G by the previous case. If $[b,d]$ is not an edge then G has an induced subgraph $b \longrightarrow a \longrightarrow d$ while G' has a flag $b \longrightarrow a \multimap d$ which contradicts the assumption that they are strongly equivalent. Thus, $[b,d]$ is an edge, namely $b \longrightarrow d$ in G because G is a CG.

This concludes the proof of the necessity of conditions [i]-[iii].

Second, we prove the sufficiency of those conditions. Since they imply the conditions (i)-(ii) from Definition 3, it follows from Theorem 8 in Roverato (2005) that G' is a CG which is complex equivalent to G but strictly larger. In particular, G and G' have the same immoralities and, to show that they are strongly equivalent, it suffices to verify that they have identical flags.

If $a \longrightarrow b \multimap d$ is a flag in G then we are to show that it is a flag in G'. The only option which avoids the desired conclusion is $a \in U$ and $b \in L$. However, then $d \in L$ and by [iii] observe $a \longrightarrow d$ in G which contradicts the assumption.

If $a \longrightarrow b \multimap d$ is a flag in G' then the fact $G' \geq G$ implies $a \longrightarrow b$ in G and the only option which avoids the desired conclusion that $a \longrightarrow b \multimap d$ is a flag in G is that $[b,d]$ was modified. There are basically two cases.

- If $b \in L$ and $d \in U$ then $a \longrightarrow b \longleftarrow d$ is an immorality in G and, because of complex equivalence of graphs, also in G'. This contradicts the assumption.

- If $b \in U$ and $d \in L$ then observe $b \in K$ and by [ii] $a \in \mathrm{pa}_G(b) \subseteq \mathrm{pa}_G(L) \setminus U$. By [iii] get $a \in \mathrm{pa}_G(d)$ which contradicts the assumption.

Thus, the sufficiency proof is finished. ■

Proof of Proposition 16

Basic observation which is needed is as follows.

Fact 4 *Let E, F, G be CGs over N with the same underlying graph such that $E \geq F \geq G$ and the following condition holds for any $c \in N$:*

[e] *if there exists $a \in N$ with $a \multimap c$ in E and $a \longrightarrow c$ in F then for every $b \in N$ with $c \multimap b$ in F one has $c \multimap b$ in G.*

If E and G are strongly equivalent then F is strongly equivalent to them as well.

Note that the conclusion of Fact 4 need not be valid if the condition [e] is omitted: consider $N = \{a, b, c\}$, E an undirected graph with $a \multimap c \multimap b$, F a CG with $a \longrightarrow c \multimap b$ and G a directed graph with $a \longrightarrow c \longrightarrow b$.

Proof We can show that F is strongly equivalent to E. If $a \longrightarrow c \longleftarrow b$ is an immorality in E then $E \geq F$ implies that it is an immorality in F. Conversely, if $a \longrightarrow c \longleftarrow b$ is an immorality in F then $F \geq G$ implies that it is an immorality in G and, therefore, in E.

If $a \longrightarrow c \multimap b$ is a flag in E, then it is a flag in G which implies $c \multimap b$ in F by $F \geq G$. Since $a \longrightarrow c$ in F by $E \geq F$ the graph F has a flag $a \longrightarrow c \multimap b$.

If $a \longrightarrow c \multimap b$ is a flag in F, then $F \geq G$ implies $a \longrightarrow c$ in G. We first verify $a \longrightarrow c$ in E by excluding two other variants of the edge $[a, e]$ in E. Since $E \geq F$ the case $a \longleftarrow c$ in E is excluded. The case $a \multimap c$ in E is also excluded, this time owing to the condition [e] from the assumption of Fact 4. Indeed, [e] says $c \multimap b$ in G, which implies that $a \longrightarrow c \multimap b$ is a flag in G and, therefore, in E, which contradicts the assumption $a \multimap c$ in E. Thus, $a \longrightarrow c$ in E and the aim is to show $c \multimap b$ in G. It can be shown by contradiction.

- If $c \longleftarrow b$ in G then $a \longrightarrow c \longleftarrow b$ is an immorality in G and, therefore, in E, which implies, by $E \geq F$, a contradictory conclusion $c \longleftarrow b$ in F.

- If $c \longrightarrow b$ in G then $a \longrightarrow c \longrightarrow b$ is an induced subgraph in G. By the condition [d] mentioned above Fact 3 applied to G and E, it is also an induced subgraph in E. The assumption $E \geq F$ then implies a contradictory conclusion $c \longrightarrow b$ in F.

Hence, $a \longrightarrow c \multimap b$ is a flag in G, and therefore in E. ■

The main step is the following 'sandwich lemma'.

Fact 5 *Let G, E be strongly equivalent CGs, $E \geq G$, $E \neq G$. Then there exists a CG F which is strongly equivalent to G and E, such that $E \geq F \geq G$ and E is obtained from F by legal merging of components.*

Note that the idea of the proof of this proposition is analogous to the proof of Theorem 7 in Roverato (2005).

Proof Since $E \geq G$, every component in E is the union of components in G and the assumption $E \neq G$ implies that there exists a component C in E containing at least two components in G. As G_C is a CG one can find a terminal component T in it. By the construction $C \setminus T \neq \emptyset$ and there is an arrow from $C \setminus T$ to T in G. Let us construct a hybrid graph F from E by replacement of all lines between $C \setminus T$ and T in E by arrows from $C \setminus T$ to T. Observe the following facts.

{**a**} F is a CG.

Assume for contradiction that F has a semi-directed cycle ρ. Since $F_{N \setminus C} = E_{N \setminus C}$ is a CG and F_C is a CG by construction, ρ has an edge between $N \setminus C$ and C, namely an arrow. This arrow is also an arrow in E (with the same direction); the other arrows of ρ either are kept in E or become lines, the lines of ρ retain in E. Therefore, ρ has to be a semi-directed cycle in E, which contradicts the assumption.

{**b**} $E \geq F \geq G$ and $F \neq E$.
The fact $E \geq F$ is evident. To see $F \geq G$ observe that if $a \longrightarrow b$ in F then either $a \longrightarrow b$ in E in which case $E \geq G$ implies $a \longrightarrow b$ in G, or $a \multimap\!\!-\!\!\multimap b$ in E. In the latter case $a \in C \setminus T$ and $b \in T$ which also implies, by the definition of T, that $a \longrightarrow b$ in G.

{**c**} $C \setminus T$ is a connected set in F and, therefore, it is a component in F.
Indeed, suppose for contradiction that distinct $a, b \in C \setminus T$ exist which are not connected by an undirected path in $F_{C \setminus T} = E_{C \setminus T}$. Since C is a connected set in E, one can construct a path $\tilde{a} \longrightarrow c_1 \multimap\!\!-\!\!\multimap \ldots \multimap\!\!-\!\!\multimap c_m \longleftarrow \tilde{b}$, $m \geq 1$ in F with some $c_1, \ldots, c_m \in T$ and $\tilde{a}, \tilde{b} \in C \setminus T$ such that $[\tilde{a}, \tilde{b}]$ is not an edge in F. This path has the same form in G and can be shortened to a complex in G. This complex is not in E which contradicts the assumption that E and G are strongly equivalent since strong equivalence implies complex equivalence.

{**d**} F is strongly equivalent to G and E.
This follows from Fact 4 owing to {a} and {b}. The condition [e] from Fact 4 holds because of the construction of F: if $a \multimap\!\!-\!\!\multimap c$ in E and $a \longrightarrow c$ in F then $c \in T$ and $c \multimap\!\!-\!\!\multimap b$ in F implies $b \in T$ for which reason $c \multimap\!\!-\!\!\multimap b$ in G.

Now, the conclusion that E is made of F by legal merging of components is easy to see. The condition {c} implies that both $C \setminus T$ and T are components in F and E is obtained from F by merging of the upper component $U \equiv C \setminus T$ and the lower component $L \equiv T$. Since E and F are strongly equivalent is follows from Proposition 15 that the merging is legal. ■

Now, the proof of Proposition 16 is easy. The required sequence $G = F_1, \ldots, F_m = H$, $m \geq 1$ can be constructed backwards by consecutive application of Fact 5 to G and $E \equiv F_i$ to get $F_{i-1} = F$ until F_{i-1} is the graph G. Of course, one starts with $F_m = H$, where $m-1$ is the difference between the numbers of components of G and H.

Proof of Proposition 18

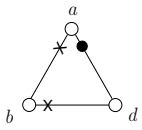
Assume for contradiction that two different orderings of applications of blocking rules leads to two different labeled graphs $G^{\ell(1)}$ and $G^{\ell(2)}$. Since they only differ in their labels, one can assume without loss of generality that $G^{\ell(1)}$ has at least one blocked label that is 'free' in $G^{\ell(2)}$. Let us fix a sequence of iterations $G_0^{\ell(1)}, G_1^{\ell(1)}, \ldots, G_n^{\ell(1)} = G^{\ell(1)}$, $n \geq 2$ leading to $G^{\ell(1)}$. Let $G_i^{\ell(1)}$ be the first graph in this sequence which has a blocked label that is 'free' in $G^{\ell(2)}$, say $a \star\!\!-\!\!\multimap d \in G_i^{\ell(1)}$ and $a \bullet\!\!-\!\!\multimap d \in G^{\ell(2)}$. In particular, $b \star\!\!-\!\!\multimap c$ in $G_j^{\ell(1)}$ for $j < i$ implies $b \star\!\!-\!\!\multimap c$ in $G^{\ell(2)}$.

We now show that $a \star\!\!-\!\!\multimap d \in G_i^{\ell(1)}$ and $a \bullet\!\!-\!\!\multimap d$ in $G^{\ell(2)}$ implies that $G^{\ell(2)}$ has a forbidden configuration, which contradicts the assumption. There are three possible cases.

1. If $a \star\!\!-\!\!\multimap d$ in $G_i^{\ell(1)}$ is blocked at a by the rule (a) then there exists a vertex b such that $b \longrightarrow d \multimap\!\!-\!\!\multimap a$ is a flag in G (cf. Algorithm 1). In particular, $b \longrightarrow d \multimap\!\!-\!\!\bullet\, a$ in $G^{\ell(2)}$ is a forbidden configuration in $G^{\ell(2)}$.

2. If $a \star\!\!-\!\!\multimap d$ in $G_i^{\ell(1)}$ is blocked at a by (b) then there exists a vertex b with $b \multimap\!\!-\!\!\multimap a \multimap\!\!-\!\!\multimap d$ in G, while $[b, d]$ is not an edge in G. Then $b \multimap\!\!-\!\!\multimap a \bullet\!\!-\!\!\multimap d$ is a forbidden configuration in $G^{\ell(2)}$.

3. If $a \mathrel{*\!\!-\!\!*} d$ in $G_i^{\ell(1)}$ is blocked at a by the rule (c) then there exists a vertex b such that the following forbidden configuration



appears in $G_{i-1}^{\ell(1)}$. As mentioned above, blocking labels in $G_j^{\ell(1)}$ for $j < i$ also occur in $G^{\ell(2)}$. Thus, that forbidden configuration is also present in $G^{\ell(2)}$.

Proof of Theorem 19

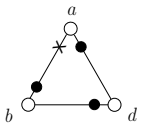
Throughout the proof we assume that G is a CG, G^ℓ the labeled version of G obtained from G by the labeling algorithm and G' the hybrid graph obtained from G^ℓ by the directing algorithm. The overall aim is to show that G' is a CG triplex equivalent to G. To improve the readability of the proof, we split it into more elementary facts. The first goal is to show that G' has no semi-directed cycle of the length 3. This is the main step to show that it has no semi-directed cycles at all, that is, it is a CG. Finally, we prove that G' is triplex equivalent to G.

We start with two auxiliary facts.

Fact 6 *If there is a semi-directed cycle in G' then it is undirected in G.*

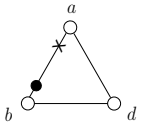
Proof Assume for contradiction that $\rho : d_0, \ldots, d_{n-1}, d_n = d_0$, $n \geq 3$ is a semi-directed cycle in G' which has an arrow $d_0 \longrightarrow d_1$ in G. Since G is a CG, there exists an arrow $d_{i-1} \longleftarrow d_i$, $2 \leq i \leq n$ in G. Basic observation is that arrows in G are kept in G' with the same direction. In particular, $d_0 \longrightarrow d_1$ and $d_{i-1} \longleftarrow d_i$ in G', which contradicts the assumption that ρ is a semi-directed cycle in G'. ■

Fact 7 *If $\rho : a, b, d, a$ is a semi-directed cycle of the length 3 in G' with $a \longrightarrow b$ in G' then it corresponds to the following configuration in G^ℓ:*

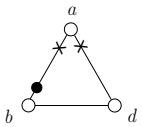


(5)

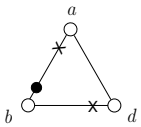
Proof By Fact 6, ρ consists of lines in G. As $a \longrightarrow b$ in G', it follows from Algorithm 2 that ρ corresponds to the following configuration



in G^ℓ. We only need to show that ρ cannot occur in either of the following two configurations in G^ℓ:

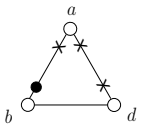


or

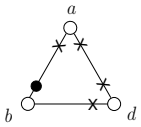


(A) (B)

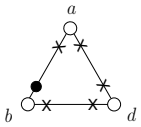
Consider the case (A) and observe that $a \multimap\!\!-\!\!\multimap d$ also has to have a blocked ending at d in G^{ℓ}. Indeed, otherwise by Algorithm 2 $a \longrightarrow d$ in G' and ρ is not a semi-directed cycle in G', which contradicts the assumption. Hence, we have



in G^{ℓ}. Now, it follows from Algorithm 1 that $b \multimap\!\!-\!\!\multimap d$ has a blocked ending at d. Indeed, otherwise a forbidden configuration $b \multimap\!\!\!\rightarrow a \multimap\!\!\!\rightarrow d \bullet\!\!-\!\!\multimap b$ of type (c) exists in G^{ℓ}. Thus, the situation is as follows:



Again, $b \multimap\!\!-\!\!\multimap d$ has a blocked ending at b for otherwise, by Algorithm 2, $b \longleftarrow d$ in G' contradicts the assumption that ρ is a semi-directed cycle. Thus, $G^{\ell}_{\{a,b,d\}}$ looks like



which is, however, also impossible because $a \multimap\!\!\!\rightarrow d \multimap\!\!\!\rightarrow b \bullet\!\!-\!\!\multimap a$ is a forbidden configuration of type (c) in G^{ℓ}. Hence, the configuration (A) cannot occur. Using the same kind of reasoning, it is also easy to check that the configuration (B) in G^{ℓ} is impossible. This is left to the reader. ■

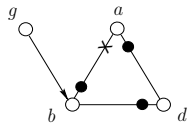
Fact 8 *G' has no semi-directed cycle of the length 3.*

Proof Suppose for contradiction that G' has a semi-directed cycle of the length 3. Thus, the set $\mathcal{A}'$ of arrows in G' belonging to (at least one of) those cycles is assumed to be non-empty. By Fact 6, every arrow $e \longrightarrow f$ in $\mathcal{A}'$ corresponds to a line $e \multimap\!\!-\!\!\multimap f$ in G, and, therefore, by the directing algorithm, to a labeled line $e \multimap\!\!\!\rightarrow f$ in G^{ℓ}. Let us fix a sequence $G^{\ell}_0, \ldots, G^{\ell}_n$, $n \geq 1$ of labeled CGs generated by the labeling algorithm. Clearly, every $e \longrightarrow f$ in $\mathcal{A}'$ is assigned the unique $1 \leq i \leq n$ such that $e \multimap\!\!\!\rightarrow f$ in G^{ℓ}_i and $e \bullet\!\!-\!\!\multimap f$ in G^{ℓ}_j for $j < i$. Let $a \longrightarrow b$ denote that arrow in $\mathcal{A}'$ which has assigned the least such i. In particular, if $e \multimap\!\!\!\rightarrow f$ in G^{ℓ}_{i-1} then $e \longrightarrow f$ does not belong to $\mathcal{A}'$.

Let us fix a semi-directed cycle $\rho : a, b, d, a$ of the length 3 in G' containing $a \longrightarrow b$. By Fact 7, the subset of vertices $\{a, b, d\}$ corresponds to the configuration (5) in G^{ℓ} and, because of the

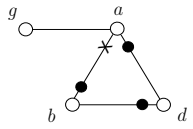
construction of G^ℓ, also in G_i^ℓ. Now, we show that the occurrence of (5) in G_i^ℓ leads to contradiction because the line $a \circ\!\!-\!\!\circ b$ in the previous iteration G_{i-1}^ℓ cannot be blocked at a by any of the blocking rules from Algorithm 1.

1. If $b \circ\!\!-\!\!* a$ is blocked at a on basis of the rule (a) then there exists a vertex $g \in N$, not adjacent to a, such that $g \longrightarrow b$ in G. If we add this arrow to the configuration (5) in G^ℓ above then we obtain (possibly omitting an edge between g and d)



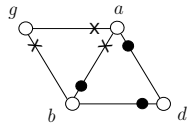
   Nodes g and d are necessarily adjacent for otherwise G^ℓ has a forbidden configuration $g \longrightarrow b \circ\!\!-\!\!\circ d$ of the type (a). Actually, one has $g \longrightarrow d$ in G as otherwise G has a semi-directed cycle g,b,d,g. However, then $g \longrightarrow d \circ\!\!-\!\!\circ a$ is a forbidden configuration of type (a) in G^ℓ, which is impossible.

2. If $a *\!\!-\!\!\circ b$ is blocked at a on basis of the rule (b) then there exists a vertex $g \in N$, not adjacent to b, such that $g \longrightarrow a$ in G. If we add this line to the configuration (5) in G^ℓ and obtain (possibly omitting an edge between g and d)

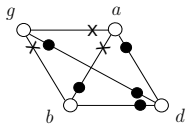


   Nodes g and d have to be adjacent for otherwise a forbidden configuration $g \longrightarrow a \circ\!\!-\!\!\circ d$ of the type (b) exists in G^ℓ. As G is a CG, one has $g \longrightarrow d$ in G. However, then $g \longrightarrow d \circ\!\!-\!\!\circ b$ is a forbidden configuration of type (b) in G^ℓ, which is impossible.

3. If $a *\!\!-\!\!\circ b$ is blocked at a on basis of the rule (c) then there exists a vertex g such that $b \longrightarrow g \longrightarrow a \circ\!\!-\!\!\circ b$ in G_{i-1}^ℓ. As $g \longrightarrow a$ and $d \longrightarrow a$ in G^ℓ one has $g \neq d$. Thus, the following configuration occurs in G^ℓ, where the possible edge between g and d is omitted:



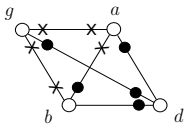
   The nodes g and d have to be adjacent for otherwise $g \longrightarrow a \circ\!\!-\!\!\circ d$ would be a forbidden configuration of type (b) in G^ℓ. Since G is a CG, one has $g \longrightarrow d$ in G. The ending of $g \longrightarrow d$ at g in G^ℓ has to be free as otherwise $d \longrightarrow g \longrightarrow a \circ\!\!-\!\!\circ d$ would be a forbidden configuration of type (c) in G^ℓ. Analogously, its ending at d in G^ℓ is also free for otherwise $b \longrightarrow g \longrightarrow d \circ\!\!-\!\!\circ b$ would be a forbidden configuration of type (c) in G^ℓ. Thus, $g \longrightarrow d$ has both endings free in G^ℓ and



To show that $a \multimap\!\!-\!\!\multimap g$ is blocked at g in G^ℓ recall that $a \ast\!\!-\!\!\ast g$ in G^ℓ_{i-1}. If $a \ast\!\!-\!\!\bullet g$ in G^ℓ then Algorithm 2 implies that a, g, d, a is a semi-directed cycle in G' (note that $d \longrightarrow\!\!\!\!\!\bullet\, a$ in G^ℓ implies that either $d \longrightarrow a$ or $d \multimap\!\!-\!\!\multimap a$ in G'). This, however, means that $a \longrightarrow g$ belongs to $\mathcal{A}'$, which contradicts the choice of $a \longrightarrow b$: as mentioned above, that choice ensures that if $e \ast\!\!-\!\!\ast f$ in G^ℓ_{i-1} then $e \longrightarrow f$ does not belong to $\mathcal{A}'$.

The conclusion that $g \multimap\!\!-\!\!\multimap b$ is blocked at b in G^ℓ can be derived analogously. If $g \ast\!\!-\!\!\bullet b$ in G^ℓ then Algorithm 2 implies that g, b, d, g is a semi-directed cycle in G'. Then $g \longrightarrow b$ belongs to $\mathcal{A}'$ which is not possible because of the fact $g \ast\!\!-\!\!\ast b$ in G^ℓ_{i-1}.

Hence, one has both $g \ast\!\!-\!\!\ast b$ and $g \ast\!\!-\!\!\ast a$ in G^ℓ, and the situation is as follows:



(6)

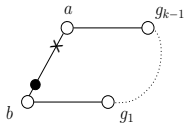
However, the configuration (6) has a subconfiguration $a \ast\!\!\longrightarrow g \ast\!\!\longrightarrow b \bullet\!\!- a$ which is a forbidden configuration of type (c) in G^ℓ. This contradicts the assumptions.

This completes the proof. ■

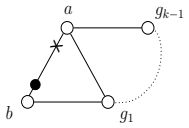
Fact 9 *The graph G' has no semi-directed cycle.*

Proof We show that if G' has a semi-directed cycle of the length $k+1$, where $k \geq 3$ then it has a semi-directed cycle of the length l, $3 \leq l \leq k$. This, together with Fact 8, implies what is desired.

Assume that $\rho : a, b, g_1, \ldots, g_{k-1}, a$, $k \geq 3$ is a semi-directed cycle in G' with $a \longrightarrow b$ in G'. By Fact 6, $a \multimap\!\!-\!\!\multimap b$ in G and Algorithm 2 implies that ρ corresponds to the following configuration



in G^ℓ, where the dotted connection stands for an undirected path and some edges are possibly omitted. It follows from Algorithm 1 that a and g_1 are adjacent in G for otherwise $g_1 \multimap\!\!-\!\!\multimap b \bullet\!\!- a$ is a forbidden configuration of the type (b) in G^ℓ. As G is a CG, $a \multimap\!\!-\!\!\multimap g_1$ in G and the situation is as follows:



If either $a \longleftarrow g_1$ or $a \longrightarrow\!\!\!\!- g_1$ in G' then a, b, g_1, a is a semi-directed cycle of the length 3 in G'. On the other hand, if $a \longrightarrow g_1$ in G' then $a, g_1, g_2, \ldots, g_{k-1}, a$ is a semi-directed cycle of the length k in G'. ■

Thus, we have verified that G' is a CG. The last step is to show that it is in the AMP equivalence class containing G.

Fact 10 *G' is triplex equivalent to G.*

Proof We already know that G and G' are CGs with the same underlying graph. Moreover, it follows from the construction of G' that $G \geq G'$.

In particular, every immorality in G remains in G'. Thus, to verify that triplexes in G are also in G' it is enough to show that every flag $a \longrightarrow b \longrightarrow\!\!\!\!- d$ in G remains a triplex in G'. As $a \longrightarrow b$ in G', the only option of canceling the triplex $\langle \{a, d\}, b \rangle$ is if $b \longrightarrow d$ in G'. Then Algorithm 2 implies $a \longrightarrow b \mathrel{*\!\!-\!\!\bullet} d$ in G^ℓ, which is, however, a forbidden configuration of type (a) in G^ℓ (cf. Algorithm 1).

Now, we show that triplexes in G' are also in G. Realize that $G \geq G'$ implies that an arrow in G' cannot be an arrow with the opposite direction in G. Thus, if $a \longrightarrow b \longrightarrow\!\!\!\!- d$ is a flag in G' then, by (2), either $a \longrightarrow b \longrightarrow\!\!\!\!- d$ or $a \longrightarrow\!\!\!\!- b \longrightarrow\!\!\!\!- d$ in G. By Algorithm 2, the latter case means $a \mathrel{*\!\!-\!\!\bullet} b \longrightarrow\!\!\!\!- d$ in G^ℓ, which is a forbidden configuration of type (b). Analogously, if $a \longrightarrow b \longleftarrow d$ is an immorality in G' that does not correspond to a triplex in G then $a \longrightarrow\!\!\!\!- b \longrightarrow\!\!\!\!- d$ in G. Hence, $a \mathrel{*\!\!-\!\!\bullet} b \mathrel{\bullet\!\!-\!\!*} d$ in G^ℓ, which is also a forbidden configuration of type (b). ■

Proof of Proposition 20

Recall that G is a CG such that there is no line of the form $\mathrel{*\!\!-\!\!\bullet}$ in its labeled version G^ℓ. Let F be a CG which is triplex equivalent to G. We are to show that $a \longrightarrow\!\!\!\!- b$ in F whenever $a \mathrel{*\!\!-\!\!*} b$ in G^ℓ. Suppose for contradiction that there exists (at least one) line of the form $e \mathrel{*\!\!-\!\!*} f$ in G^ℓ such that $e \longrightarrow f$ in F. Thus, the set $\mathcal{A}_F$ of arrows $e \longrightarrow f$ in F of the form $e \mathrel{*\!\!-\!\!*} f$ in G^ℓ is assumed to be non-empty.

Let us fix a chain of components $C_1, \ldots, C_m$, $m \geq 1$ in F. Let k be the highest $1 \leq k \leq m$ such that there exists an arrow $e \longrightarrow f$ from $\mathcal{A}_F$ with $f \in C_k$. Denote by $\mathcal{A}'_F$ the subset of $\mathcal{A}_F$ consisting of arrows $e \longrightarrow f$ with $f \in C_k$. Clearly, $\mathcal{A}'_F \neq \emptyset$.

The next step is to fix a sequence $G^\ell_0, \ldots G^\ell_n$, $n \geq 1$ of labeled CGs generated by Algorithm 1. Every $e \longrightarrow f$ from $\mathcal{A}'_F$ is assigned unique $1 \leq i \leq n$ such that $e \longrightarrow\!\!\!\!* f$ in G^ℓ_i and $e \longrightarrow\!\!\!\!\bullet f$ in G^ℓ_j for $j < i$. Let $a \longrightarrow b$ denote the arrow from $\mathcal{A}'_F$ which has assigned the least such i. Observe that this choice of $a \longrightarrow b$ ensures that the following two conditions are valid.

(I) If $b \longrightarrow d$ in F for some node d then $b \longrightarrow d$ does not belong to $\mathcal{A}_F$.

This is because $b \in C_k$. The fact that $C_1, \ldots, C_m$ is a chain for F implies $d \in C_l$ with $l > k$. However, k was chosen so that no arrow $e \longrightarrow f$ from $\mathcal{A}_F$ with $f \in C_l$ for $l > k$ exists.

(II) Whenever $e \longrightarrow\!\!\!\!* f$ in G^ℓ_{i-1} then $\mathcal{A}'_F$ does not contain $e \longrightarrow f$.

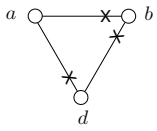
This follows from the choice of i: a necessary condition for $e \longrightarrow f$ to belong to $\mathcal{A}'_F$ is $e \longrightarrow\!\!\!\!* \; f$ in G^ℓ_j only for $j \geq i$, that is, $e \longrightarrow\!\!\!\!\bullet \; f$ in G^ℓ_{i-1}.

Now, we are going to derive a contradictory conclusion that $a \; *\!\!-\!\!* \; b$ cannot be blocked at b by any of the blocking rules from Algorithm 1.

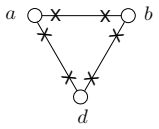
1. If $a \; *\!\!-\!\!* \; b$ is blocked at b on basis of the blocking rule (a) then there exists a vertex d such that $d \longrightarrow a - b$ in G and b is not adjacent to d. Hence, $d \longrightarrow a - b$ is a flag in G. As G and F are triplex equivalent, F has a triplex $\langle\{b,d\},a\rangle$, which, however, contradicts the assumption $a \longrightarrow b$ in F.

2. If $a \; *\!\!-\!\!* \; b$ is blocked at b on basis of the blocking rule (b) then there exists a vertex d such that $a - b - d$ in G and a is not adjacent to d. This implies $b \longrightarrow d$ in F for otherwise the fact $a \longrightarrow b$ in F implies that $\langle\{a,d\},b\rangle$ is a triplex in F which is not in G. Moreover, $a - b - d$ in G implies, by the blocking rule (b) from Algorithm 1, that $a \longrightarrow\!\!\!\!* \; b \; *\!\!-\!\!* \; d$ in G^ℓ. Because G^ℓ has no lines of the form $*\!\!-\!\!\bullet$ this means $a \; *\!\!-\!\!* \; b \; *\!\!-\!\!* \; d$ in G^ℓ. Thus, $b \longrightarrow d$ belongs to $\mathcal{A}_F$, contradicting the condition (I) above.

3. If $a \; *\!\!-\!\!* \; b$ is blocked at b on basis of the blocking rule (c) then there exists a vertex d such that $a \longrightarrow\!\!\!\!* \; d \longrightarrow\!\!\!\!* \; b \; \bullet\!\!- \; a$ in $\widehat{G}^\ell_{i-1}$. Thus, we have

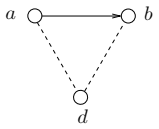


in G^ℓ. Since there is no line of the type $*\!\!-\!\!\bullet$ in G^ℓ, we have



(7)

in G^ℓ, whereas the corresponding subgraph in F is



(8)

where the dashed connection means that the nodes are adjacent. However, the configurations (7) and (8) cannot coexist because any possible type of the edge between d and b in F leads to a contradiction.

- If $d \longrightarrow b$ in F then (7) and the fact $b \in C_k$ imply $d \longrightarrow b$ is in $\mathcal{A}'_F$. As $d \longrightarrow\!\!\!\!* \; b$ in G^ℓ_{i-1} this contradicts the condition (II) above.
- If $d - b$ in F then $d \in C_k$ and $a \longrightarrow d$ in F for otherwise F has a semi-directed cycle. Hence, by (7) $a \longrightarrow d$ belongs to $\mathcal{A}'_F$. As $a \longrightarrow\!\!\!\!* \; d$ in $\widehat{G}^\ell_{i-1}$ it also contradicts the condition (II) above.
- If $b \longrightarrow d$ in F then (7) gives $b \longrightarrow d$ in $\mathcal{A}_F$ contradicting the condition (I) above.

This concludes the proof.

References

S. A. Andersson, D. Madigan and M. D. Perlman. An alternative Markov property for chain graphs. In *Uncertainty in Artificial Intelligence: Proceedings of the Twelfth Conference* (F. Jensen and E. Horvitz eds.), pages 40–48, Morgan Kaufmann, San Francisco, 1996.

S. A. Andersson, D. Madigan and M. D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25:505–541, 1997.

S. A. Andersson, D. Madigan and M. D. Perlman. Alternative Markov properties for chain graphs. *Scandinavian Journal of Statistics*, 28:33–85, 2001.

S. A. Andersson and M. D. Perlman. Characterizing Markov equivalence classes for AMP chain graph models. *The Annals of Statistics*, 34, 2006, forthcoming.

D. M. Chickering. Learning equivalence classes of Bayesian-network structure. *Journal of Machine Learning Research*, 2:445–498, 2002.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer–Verlag, New York, 1999.

A. P. Dawid. Influence diagrams for causal modelling and inference. *International Statistical Review*, 70:161–189, 2002.

M. Frydenberg. The chain graph Markov property. *Scandinavian Journal of Statistics*, 17:333–353, 1990.

S. L. Lauritzen and N. Wermuth. Mixed interaction models. Research Report R-84-8, Inst. Elec. Sys., University of Aalborg, Denmark, 1984.

S. L. Lauritzen and N. Wermuth. Graphical models for association between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17:31–57, 1989.

S. L. Lauritzen. Causal inference from graphical models. In *Complex Stochastic Systems*, (O. E. Barndorff-Nielsen, D. R. Cox and C. Klüppelberg eds.), pages 63–107, Chapman and Hall/CRC, 2001.

A. Roverato. A unified approach to the characterisation of equivalence classes of DAGs, chain graphs with no flags and chain graphs. *Scandinavian Journal of Statistics*, 32:295–312, 2005.

A. Roverato and L. La Rocca. On block ordering of variables in graphical modeling. *Scandinavian Journal of Statistics*, 33:65–81, 2006.

M. Studený. A recovery algorithm for chain graphs. *International Journal of Approximate Reasoning*, 17:265–293, 1997.

M. Studený, Characterization of essential graphs by means of an operation of legal component merging. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12:43–62, 2004.

M. Studený, A. Roverato and S. Štěpánová. Two operations of merging components in a chain graph. Submitted. Available electronically at `http://staff.utia.cas.cz/studeny/aa23.html`, 2006.

M. Volf and M. Studený. A graphical characterization of the largest chain graphs. *International Journal of Approximate Reasoning*, 20:209–236, 1999.

Journal of Machine Learning Research 7 (2006) 1079–1105 Submitted 2/05; Published 6/06

Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems*

Eyal Even-Dar EVENDAR@SEAS.UPENN.EDU
Department of Information and Computer Science
University of Pennsylvania
Philadelphia, PA 19104

Shie Mannor SHIE@ECE.MCGILL.CA
Department of Electrical & Computer Engineering
McGill University
H3A-2A7 Québec, Canada

Yishay Mansour MANSOUR@CS.TAU.AC.IL
School of Computer Science
Tel-Aviv University
Tel-Aviv, 69978, Israel

Editor: Sridhar Mahadevan

Abstract

We incorporate statistical confidence intervals in both the multi-armed bandit and the reinforcement learning problems. In the bandit problem we show that given n arms, it suffices to pull the arms a total of $O\left((n/\varepsilon^2)\log(1/\delta)\right)$ times to find an ε-optimal arm with probability of at least $1-\delta$. This bound matches the lower bound of Mannor and Tsitsiklis (2004) up to constants. We also devise action elimination procedures in reinforcement learning algorithms. We describe a framework that is based on learning the confidence interval around the value function or the Q-function and eliminating actions that are not optimal (with high probability). We provide a model-based and a model-free variants of the elimination method. We further derive stopping conditions guaranteeing that the learned policy is approximately optimal with high probability. Simulations demonstrate a considerable speedup and added robustness over ε-greedy Q-learning.

1. Introduction

Two of the most studied problems in control, decision theory, and learning in unknown environment are the multi-armed bandit (MAB) and reinforcement learning (RL). In this paper we consider both models under the probably approximately correct (PAC) settings and study several important questions arising in this model. The first question is when can an agent stop learning and start exploiting using the knowledge it obtained. The second question is which strategy leads to minimal learning time. Since the multi-armed bandit setup is simpler, we start by introducing it and later describe the reinforcement learning problem.

The Multi-armed bandit problem is one of the classical problems in decision theory and control. There is a number of alternative arms, each with a stochastic reward whose probability distribution is initially unknown. We try these arms in some order, which may depend on the sequence of rewards

*. Preliminary and partial results from this work appeared as extended abstracts in COLT 2002 and ICML 2003.

that have been observed so far. A common objective in this context is to find a policy for choosing the next arm to be tried, under which the sum of the expected rewards comes as close as possible to the ideal reward, i.e., the expected reward that would be obtained if we were to try the "best" arm at all times. One of the attractive features of the multi-armed bandit problem is that despite its simplicity, it encompasses many important decision theoretic issues, such as the tradeoff between exploration and exploitation.

The multi-armed bandit problem has been widely studied in a variety of setups. The problem was first considered in the 50's in the seminal work of Robbins (1952) that derives strategies that asymptotically attain an average reward that converges in the limit to the reward of the best arm. The multi-armed bandit problem was later studied in discounted, Bayesian, Markovian, expected reward, and adversarial setups. (See Berry and Fristedt, 1985, for a review of the classical results on the multi-armed bandit problem.) Most of the research so far has considered the expected regret, and devised strategies for minimizing it. The seminal work of Lai and Robbins (1985) provides tight bounds as a function of the Kullback-Leibler divergence between the arms reward distribution, and a logarithmic growth with the number of steps. The bounds of Lai and Robbins (1985) were shown to be efficient, in the sense that the convergence rates are optimal. The adversarial multi-armed bandit problem was considered in Auer et al. (1995, 2002), where it was shown that the expected regret grows proportionally to the square root of the number of steps.

We consider the classical multi-armed bandit problem, but rather than looking at the expected regret, we develop PAC style bounds. The agent's goal is to find, with high probability, a near optimal arm, namely, with probability at least $1-\delta$ output an ε-optimal arm. This naturally abstracts the case where the agent needs to choose one specific arm, and it is given only limited exploration initially. Our main complexity criterion, in addition to correctness, is the number of steps taken by the algorithm, which can be viewed as pure exploration steps. This is in contrast to most of the results for the multi-armed bandit problem, where the main aim is to maximize the expected cumulative reward while both exploring and exploiting. Therefore, methods which balance between exploration and exploitation such as softmax, and ε-greedy are not comparable to our methods. Following our initial conference publication, a lower bound on the number of steps needed to obtain a PAC solution was developed in Mannor and Tsitsiklis (2004); it matches the upper bound we develop in this paper.

The MAB problem models a situation where the environment is static and the same decision has to be made repeatedly. In many cases of practical interest, the model should represent a situation where the state of the system changes with time. This is encompassed in the Markov decision process model (MDP), that has been the subject of intensive research since the 1950's. When the model is known, and learning is not required, there are several standard methods for calculating the optimal policy - linear programming, value iteration, policy iteration, etc.; see Puterman (1994) for a review. When the model is not known a-priori, a *learning* scheme is needed. RL has emerged in the recent decade as unified discipline for adaptive control of dynamic environments (e.g., Sutton and Barto, 1998, Bertsekas and Tsitsiklis, 1996). A common problem with many RL algorithms is a slow convergence rate, even for relatively small problems. For example, consider the popular Q-learning algorithm (Watkins, 1989) which is essentially an asynchronous stochastic approximation algorithm (Bertsekas and Tsitsiklis, 1996). Generic convergence rate bounds for stochastic approximation (e.g., Borkar and Meyn, 2000) or specific rates for Q-learning (see, Kearns and Singh, 1998, Even-Dar and Mansour, 2003) are somewhat disappointing. However, the generic convergence rate is shown there to be almost tight for several particularly bad scenarios. The question that we ask

is: When is enough information gathered? When can the learning agent declare with reasonable confidence that the policy discovered is optimal, or at least approximately optimal? To summarize the differences, we are not concerned in the generic convergence rate (which must be slow), but we are rather interested in supplying rates which will be adjusted to the specific MDP parameters and as a result are much better for certain problems.

The problem of obtaining stopping conditions for learning in MDPs is a fundamental problem in RL. As opposed to supervised learning problems where typically a data set is given to the learner who has to commit to a classifier (or regressor in regression problem), in RL the decision maker can continue its interaction with the environment and obtain additional samples. The stopping rules that are currently employed in practice are based on ad-hoc rules, and may lead to premature stopping or to overly long trajectories.

When an action in a certain state can be determined to *not* belong to the optimal policy in an MDP, it can be discarded and disregarded in both planning and learning. This idea, commonly known as action elimination (AE), was proposed by MacQueen (1966) in the context of planning when the MDP parameters are known. In the planning case AE serves two purposes: reduce the size of the action sets to be searched at every iteration; identify optimal policies when there is a unique optimal policy. (In value iteration this is the only way to reach optimal policy rather than ε-optimal policy.) AE procedures are standard practice in solving large practical MDPs and are considered state-of-the-art; see Puterman (1994) for more details. We consider AE in the *learning* context when the model is not known a-priori.

In many applications the computational power is available but sampling of the environment is expensive. By eliminating sub-optimal actions early in the learning process, the total amount of sampling is reduced, leading to spending less time on estimating the parameters of sub-optimal actions. The main motivation for applying AE in RL is reducing the amount of samples needed from the environment. In addition to that, AE in RL enjoys the same advantages as in MDPs - convergence rate speedup and possibility to find an optimal policy (rather than ε-optimal).

Overview of the paper

After defining the settings in Section 2, we consider the MAB problem in Section 3. We start from a naive algorithm for the MAB problem, and present two improved algorithms. The first algorithm in the bandit settings, *Successive Elimination*, has the potential to exhibit an improved behavior in cases where the differences between the expected rewards of the optimal arm and sub-optimal arms are much larger than ε. The second algorithm, *Median Elimination*, achieves a better dependence on the number of arms. Namely, the total number of arm trials is $O(n/\varepsilon^2 \log(1/\delta))$, which improves the naive bound by a factor of $\log n$, and matches the lower bounds given in Mannor and Tsitsiklis (2004).

In Section 4 we consider AE in RL. The underlying idea is to maintain upper and lower estimates of the value (or Q) function. When the expected upper estimate of the return of a certain action falls below the expected lower estimate of another action, the obviously inferior action is eliminated. We suggest both, a model-based and a Q-learning style AE algorithms. The upper and lower bounds are based on a large deviations inequality, so that when an action is eliminated, it is not optimal with high probability.

Stopping conditions that are based on generic convergence rate bounds (as in Even-Dar and Mansour, 2003) are overly conservative. We suggest a stopping time based on the difference be-

tween the upper and lower bounds of the value (or Q) function. We show that if the difference is small, then the greedy policy with respect to the lower estimate is almost optimal.

In Section 5 we present the results of several experiments with AE in toy problems as well as in non-trivial problems. Significant speedup with negligible computational overhead is observed as compared to ε-greedy Q-learning.

2. Model and Preliminaries

In this section we define the models considered in this paper. We start from the MAB model in Section 2.1. We then describe the MDP model in Section 2.2. While both models are well studied we prefer to recapitulate them in the PAC setup, to avoid confusion. We finally recall Hoeffding's inequality which is a central tool in this work in Section 2.3.

2.1 Multi-Armed Bandit

The model is comprised of a set of arms A with $n = |A|$. When sampling arm $a \in A$ a reward which is a random variable $R(a)$ is received. We assume that the reward is binary, i.e., for every arm $a \in A$ the reward $R(a) \in \{0,1\}$ (all the results apply without change if the reward is bounded in $[0,1]$ and in general as long as the reward is bounded with appropriate modifications). Denote the arms by $a_1, \cdots, a_n$ and $p_i = \mathbb{E}[R(a_i)]$. For simplicity of notations we enumerate the arms according to their expected reward $p_1 > p_2 > ... > p_n$.

An arm with the highest expected reward is called the *best arm*, and denoted by a^*, and its expected reward r^* is the *optimal reward*. An arm whose expected reward is strictly less than r^*, the expected reward of the best arm, is called a *non-best arm*. An arm a is called an ε-*optimal arm* if its expected reward is at most ε from the optimal reward, i.e., $\mathbb{E}[R(a)] \geq r^* - \varepsilon$.

An algorithm for the MAB problem, at each time step t, samples an arm a_t and receives a reward r_t (distributed according to $R(a_t)$). When making its selection the algorithm may depend on the history (i.e., the actions and rewards) up to time $t-1$. Eventually the algorithm must commit to a single arm and select it.

Next we define the desired properties of an algorithm formally.

Definition 1 *An algorithm is a* (ε, δ)*-PAC algorithm for the multi armed bandit with* sample complexity T*, if it outputs an* ε*-optimal arm,* a'*, with probability at least* $1-\delta$*, when it terminates, and the number of time steps the algorithm performs until it terminates is bounded by* T.

Remark 2 The MAB algorithm may terminate before T steps passed. The sample complexity we consider is the complexity of the *worst* trajectory. The expected sample complexity (where the expectation is taken with respect to both the model and the algorithm) was considered in Mannor and Tsitsiklis (2004). The expected sample complexity behaves like $\Omega((n+\log(1/\delta))/\varepsilon^2)$, which is different than the complexity we prove below in Theorem 10. We note that the running time of the algorithm from Mannor and Tsitsiklis (2004) is not bounded in the worst case.

2.2 Markov Decision Processes

We define an MDP as follows:

Definition 3 *A Markov Decision process (MDP) M is a 4-tuple (S,A,P,R), where S is a set of the states, A is a set of actions, $P^a_{s,s'}$ is the transition probability from state s to state s' when performing action $a \in A$ in state s, and $R(s,a)$ is the reward received when performing action a in state s.*

A strategy for an MDP assigns, at each time t, for each state s a probability for performing action $a \in A$, given a history $F_{t-1} = \{s_1, a_1, r_1, ..., s_{t-1}, a_{t-1}, r_{t-1}\}$ which includes the states, actions and rewards observed until time $t-1$. While executing a strategy π we perform at time t action a_t in state s_t and observe a reward r_t (distributed according to $R(s_t,a_t)$), and the next state s_{t+1} distributed according to $P^{a_t}_{s_t,\cdot}$. We combine the sequence of rewards into a single value called the *return*. Our goal is to maximize the return. In this work we focus on the *discounted return*, which has a parameter $\gamma \in (0,1)$, and the discounted return of policy π is $V^\pi = \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward observed at time t. We also consider the *finite horizon* return, $V^\pi = \sum_{t=0}^{H} r_t$ for a given horizon H.

We assume that $R(s,a)$ is non-negative and bounded by R_{max}, i.e., for every s,a : $0 \le R(s,a) \le R_{max}$. This implies that the discounted return is bounded by $V_{max} = R_{max}/(1-\gamma)$; for the finite horizon the return is bounded by HR_{max}. We define a value function for each state s, under policy π, as $V^\pi(s) = \mathbb{E}^\pi[\sum_{i=0}^{\infty} r_i \gamma^i]$, where the expectation is over a run of policy π starting at state s. We further denote the state-action value function as using action a in state s and then following π as:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P^a_{s,s'} V^\pi(s').$$

Similarly, we define the value functions for the finite horizon model.

Let π^* be an optimal policy which maximizes the return from any start state. For discounted return criterion, there exists such a policy which is deterministic and stationary (see, e.g., Puterman, 1994). This implies that for any policy π and any state s we have $V^{\pi^*}(s) \ge V^\pi(s)$, and $\pi^*(s) = \text{argmax}_a(R(s,a) + \gamma(\sum_{s'} P^a_{s,s'} V^{\pi^*}(s'))$. We use V^* and Q^* for V^{π^*} and Q^{π^*}, respectively. We say that a policy π is ε-optimal if $\|V^* - V^\pi\|_\infty \le \varepsilon$. We also define the policy $Greedy(Q)$ as the policy that prescribes in each state the action that maximizes the Q-function in the state, i.e., $\pi(s) = \text{argmax}_a Q(s,a)$.

For a given trajectory let: $T^{s,a}$ be the set of times in which we perform action a in state s and $T^{s,a,s'}$ be a subset of $T^{s,a}$ in which we reached state s'. Also, $\#(s,a,t)$ is the number of times action a is performed in state s up to time t, i.e., $|T^{s,a} \cap \{1,2,3,\ldots,t\}|$. We similarly define $\#(s,a,s',t)$ as $|T^{s,a,s'} \cap \{1,2,3,\ldots,t\}|$. Next we define the empirical model at time t. Given that $\#(s,a,t) > 0$ we define the empirical next state distribution at time t as

$$\hat{P}^a_{s,s'} = \frac{\#(s,a,s',t)}{\#(s,a,t)} \quad \text{and} \quad \hat{R}(s,a) = \frac{\sum_{t \in T^{s,a}} r_t}{\#(s,a,t)}.$$

If $\#(s,a,t) = 0$ the empirical model and the reward can be chosen arbitrarily. We define the expectation of the empirical model as $\hat{\mathbb{E}}_{s,s',a}[V(s')] = \sum_{s' \in S} \hat{P}^a_{s,s'} V(s')$. To simplify the notations we omit s,a in the notations $\hat{\mathbb{E}}_{s'}$ whenever evident.

2.3 A Concentration Bound

We often use large deviation bounds in this paper. Since we assume boundedness we can rely on Hoeffding's inequality.

Lemma 4 *(Hoeffding, 1963) Let X be a set, D be a probability distribution on X, and $f_1,...,f_m$ be real-valued functions defined on X with $f_i : X \to [a_i,b_i]$ for $i = 1,...,m$, where a_i and b_i are real numbers satisfying $a_i < b_i$. Let $x_1,\ldots,x_m$ be independent identically distributed samples from D. Then we have the following inequality*

$$\begin{aligned}
\mathbf{P}\left[\frac{1}{m}\sum_{i=1}^{m} f_i(x_i) - \left(\frac{1}{m}\sum_{i=1}^{m}\int_{a_i}^{b_i} f_i(x)D(x)\right) \geq \varepsilon\right] &\leq e^{-\frac{2\varepsilon^2 m^2}{\sum_{i=1}^{m}(b_i-a_i)^2}} \\
\mathbf{P}\left[\frac{1}{m}\sum_{i=1}^{m} f_i(x_i) - \left(\frac{1}{m}\sum_{i=1}^{m}\int_{a_i}^{b_i} f_i(x)D(x)\right) \leq -\varepsilon\right] &\leq e^{-\frac{2\varepsilon^2 m^2}{\sum_{i=1}^{m}(b_i-a_i)^2}}.
\end{aligned}$$

Remark 5 We note that the boundedness assumption is not essential and can be relaxed in certain situations. We also note that sometimes tighter bounds can be obtained using the relative Chernoff bound (Angluin and Valiant, 1979).

3. PAC Bounds for the Multi-Armed Bandit Problem

In this section we investigate an (ε,δ)-PAC algorithms for the MAB problem. Such algorithms are required to output with probability $1-\delta$ an ε-optimal arm. We start with a naive solution that samples each arm $1/(\varepsilon/2)^2\ln(2n/\delta)$ and picks the arm with the highest empirical reward. The sample complexity of this naive algorithm is $O(n/\varepsilon^2\log(n/\delta))$. The naive algorithm is described in Algorithm 1. In Section 3.1 we consider an algorithm that eliminates one arm after the other. In Section 3.2 we finally describe the Median Elimination algorithm whose sample complexity is optimal in the worst case.

Input : $\varepsilon > 0$, $\delta > 0$
Output : An arm
foreach Arm $a \in A$ **do**
  Sample it $\ell = \frac{4}{\varepsilon^2}\ln(\frac{2n}{\delta})$ times;
  Let $\hat{p}_a$ be the average reward of arm a;
end
Output $a' = \arg\max_{a\in A}\{\hat{p}_a\}$;

Algorithm 1: Naive Algorithm

Theorem 6 *The algorithm* Naive(ε,δ) *is an* (ε,δ)*-PAC algorithm with arm sample complexity* $O\left((n/\varepsilon^2)\log(n/\delta)\right)$.

Proof The sample complexity is immediate from the definition of the algorithm. We now prove it is an (ε,δ)-PAC algorithm. Let a' be an arm for which $\mathbb{E}(R(a')) < r^* - \varepsilon$. We want to bound the probability of the event $\hat{p}_{a'} > \hat{p}_{a^*}$.

$$\begin{aligned}
P(\hat{p}_{a'} > \hat{p}_{a^*}) &\leq P\left(\hat{p}_{a'} > \mathbb{E}[R(a')] + \varepsilon/2 \text{ or } \hat{p}_{a^*} < r^* - \varepsilon/2\right) \\
&\leq P\left(\hat{p}_{a'} > \mathbb{E}[R(a')] + \varepsilon/2\right) + P\left(\hat{p}_{a^*} < r^* - \varepsilon/2\right) \\
&\leq 2\exp(-2(\varepsilon/2)^2\ell),
\end{aligned}$$

where the last inequality uses the Hoeffding inequality. Choosing $\ell = (2/\varepsilon^2)\ln(2n/\delta)$ assures that $P(\hat{p}_{a'} > \hat{p}_{a^*}) \leq \delta/n$. Summing over all possible a' we have that the failure probability is at most $(n-1)(\delta/n) < \delta$. ■

3.1 Successive Elimination

The successive elimination algorithm attempts to sample each arm a minimal number of times and eliminate the arms one after the other. To motivate the successive elimination algorithm, we first assume that the expected rewards of the arms are known, but the matching of the arms to the expected rewards is unknown. Let $\Delta_i = p_1 - p_i > 0$. Our aim is to sample arm a_i for $(1/\Delta_i^2)\ln(n/\delta)$ times, and then eliminate it. This is done in phases. Initially, we sample each arm $(1/\Delta_n^2)\ln(n/\delta)$ times. Then we eliminate the arm which has the lowest empirical reward (and never sample it again). At the i-th phase we sample each of the $n-i$ surviving arms

$$O\left(\left(\frac{1}{\Delta_{n-i}^2} - \frac{1}{\Delta_{n-i+1}^2}\right)\log\left(\frac{n}{\delta}\right)\right)$$

times and then eliminate the empirically worst arm. The algorithm described as Algorithm 2 below. In Theorem 7 we prove that the algorithm is $(0,\delta)$-PAC and compute its sample complexity.

Input : $\delta > 0$, bias of arms $p_1, p_2, \ldots, p_n$
Output : An arm
Set $S = A$; $t_i = (8/\Delta_i^2)\ln(2n/\delta)$; and $t_{n+1} = 0$, for every arm a: $\hat{p}_a = 0$, $i = 0$;
while $i < n-1$ **do**
  Sample every arm $a \in S$ for $t_{n-i} - t_{n-i+1}$ times;
  Let $\hat{p}_a$ be the average reward of arm a (in all rounds);
  Set $S = S \setminus \{a_{\min}\}$, where $a_{\min} = \arg\min_{a \in S}\{\hat{p}_a\}$, $i = i+1$;
end
Output S;

Algorithm 2: Successive Elimination with Known Biases

Theorem 7 *Suppose that $\Delta_i > 0$ for $i = 2,3,\ldots,n$. Then the Successive Elimination with Known Biases algorithm is an $(0,\delta)$-PAC algorithm and its arm sample complexity is*

$$O\left(\log\left(\frac{n}{\delta}\right)\sum_{i=2}^{n}\frac{1}{\Delta_i^2}\right). \tag{1}$$

Proof The sample complexity of the algorithm is as follows. In the first round we sample n arms t_n times. In the second round we sample $n-1$ arms $t_{n-1} - t_n$ times. In the kth round ($1 \leq k < n$) we sample $n-k+1$ arms for $t_{n-k} - t_{n-k+1}$ times. The total number of arms samples is therefore $t_2 + \sum_{i=2}^{n} t_i$ which is of the form (1).
We now prove that the algorithm is correct with probability at least $1-\delta$. Consider first a simplified

algorithm which is similar to the naive algorithm, suppose that each arm is pulled $8/(\Delta_2^2)\ln(2n/\delta)$ times. For every $2 \le i \le n-1$ we define the event

$$E_i = \left\{ \hat{p}_1^{t_j} \ge \hat{p}_i^{t_j} | \forall t_j \text{ s.t. } j \ge i \right\},$$

where $\hat{p}_i^{t_j}$ is the empirical value the ith arm at time t_j. If the events E_i hold for all $i > 1$ the algorithm is successful.

$$\begin{aligned}
\mathbf{P}[\text{not}(E_i)] &\le \sum_{j=i}^{n} \mathbf{P}[\hat{p_n}^{t_j} < \hat{p}_i^{t_j}] \\
&\le \sum_{j=i}^{n} 2\exp(-2(\Delta_i/2)^2 t_j) \le \sum_{j=i}^{n} 2\exp(-2(\Delta_i/2)^2 8/\Delta_j^2 \ln(2n/\delta)) \\
&\le \sum_{j=i}^{n} 2\exp(-\ln(4n^2/\delta^2)) \\
&\le (n-i+1)\delta^2/n^2 \le \frac{\delta}{n}.
\end{aligned}$$

Using the union bound over all E_i's we obtain that the simplified algorithm satisfies all E_i with probability at least $1-\delta$. Consider the original setup. If arm 1 is eliminated at time t_j for some is implies that some arm $i < j$ has higher empirical value at time t_j. The probability of failure of the algorithm is bounded by the probability of failure in the simplified setting. ■

Next, we relax the requirement that the expected rewards of the arms are known in advance, and introduce the Successive Elimination algorithm that works with any set of biases. The algorithm we present as Algorithm 3 finds the best arm (rather than ε-best) with high probability. We later explain in Remark 9 how to modify it to be an (ε, δ)-PAC algorithm.

Input : $\delta > 0$
Output : An arm
Set $t = 1$ and $S = A$;
Set for every arm a: $\hat{p}_a^1 = 0$;
Sample every arm $a \in S$ once and let $\hat{p}_a^t$ be the average reward of arm a by time t;
repeat
  Let $\hat{p}_{max}^t = \max_{a \in S} \hat{p}_a^t$ and $\alpha_t = \sqrt{\ln(cnt^2/\delta)/t}$, where c is a constant;
  foreach arm $a \in S$ such that $\hat{p}_{max}^t - \hat{p}_a^t \ge 2\alpha_t$ **do**
    set $S = S \setminus \{a\}$;
  end
  $t = t + 1$;
until $|S| > 1$;

Algorithm 3: Successive elimination with unknown biases

Theorem 8 *Suppose that $\Delta_i > 0$ for $i = 2, 3, \ldots, n$. Then the Successive Elimination algorithm (Algorithm 3) is a $(0, \delta)$-PAC algorithm, and with probability at least $1-\delta$ the number of samples*

is bounded by

$$O\left(\sum_{i=2}^{n}\frac{\ln(\frac{n}{\delta\Delta_i})}{\Delta_i^2}\right).$$

Proof Our main argument is that, at any time t and for any action a, the observed probability $\hat{p}_a^t$ is within α_t of the true probability p_a. For any time t and action $a \in S_t$ we have that,

$$\mathbf{P}[|\hat{p}_a^t - p_a| \geq \alpha_t] \leq 2e^{-2\alpha_t^2 t} \leq \frac{2\delta}{cnt^2}.$$

By taking the constant c to be greater than 4 and from the union bound we have that with probability at least $1-\delta/n$ for any time t and any action $a \in S_t$, $|\hat{p}_a^t - p_a| \leq \alpha_t$. Therefore, with probability $1-\delta$, the best arm is never eliminated. Furthermore, since α_t goes to zero as t increases, eventually every non-best arm is eliminated. This completes the proof that the algorithm is $(0,\delta)$-PAC.

It remains to compute the arm sample complexity. To eliminate a non-best arm a_i we need to reach a time t_i such that,

$$\hat{\Delta}_{t_i} = \hat{p}_{a_1}^{t_i} - \hat{p}_{a_i}^{t_i} \geq 2\alpha_{t_i}.$$

The definition of α_t combined with the assumption that $|\hat{p}_a^t - p_a| \leq \alpha_t$ yields that

$$\Delta_i - 2\alpha_t = (p_1 - \alpha_t) - (p_i + \alpha_t) \geq \hat{p}_1 - \hat{p}_i \geq 2\alpha_t,$$

which holds with probability at least $1-\frac{\delta}{n}$ for

$$t_i = O\left(\frac{\ln(n/\delta\Delta_i)}{\Delta_i^2}\right).$$

To conclude, with probability of at least $1-\delta$ the number of arm samples is $2t_2 + \sum_{i=3}^{n} t_i$, which completes the proof. ■

Remark 9 One can easily modify the successive elimination algorithm so that it is (ε,δ)-PAC. Instead of stopping when only one arm survives the elimination, it is possible to settle for stopping when either only one arm remains or when each of the k surviving arms were sampled $O(\frac{1}{\varepsilon^2}\log(\frac{k}{\delta}))$. In the latter case the algorithm returns the best arm so far. In this case it is not hard to show that the algorithm finds an ε-optimal arm with probability at least $1-\delta$ after

$$O\left(\sum_{i:\Delta_i>\varepsilon}\frac{\log(\frac{n}{\delta\Delta_i})}{\Delta_i^2} + \frac{N(\Delta,\varepsilon)}{\varepsilon^2}\log\left(\frac{N(\Delta,\varepsilon)}{\delta}\right)\right),$$

where $N(\Delta,\varepsilon) = |\{i \mid \Delta_i < \varepsilon\}|$ is the number of arms which are ε-optimal.

3.2 Median Elimination

The following algorithm substitutes the term $O(\log(1/\delta))$ for $O(\log(n/\delta))$ of the naive bound. The idea is to eliminate the worst half of the arms at each iteration. We do not expect the best arm to be empirically "the best", we only expect an ε-optimal arm to be above the median.

Input : $\varepsilon > 0, \delta > 0$
Output : An arm
Set $S_1 = A$, $\varepsilon_1 = \varepsilon/4$, $\delta_1 = \delta/2$, $\ell = 1$. **repeat**
  Sample every arm $a \in S_\ell$ for $1/(\varepsilon_\ell/2)^2 \log(3/\delta_\ell)$ times, and let $\hat{p}_a^\ell$ denote its empirical value;
  Find the median of $\hat{p}_a^\ell$, denoted by m_ℓ;
  $S_{\ell+1} = S_\ell \setminus \{a : \hat{p}_a^\ell < m_\ell\}$;
  $\varepsilon_{\ell+1} = \frac{3}{4}\varepsilon_\ell$; $\delta_{\ell+1} = \delta_\ell/2$; $\ell = \ell + 1$;
until $|S_\ell| = 1$;

Algorithm 4: Median Elimination

Theorem 10 *The Median Elimination(ε,δ) algorithm is an (ε, δ)-PAC algorithm and its sample complexity is*

$$O\left(\frac{n}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right).$$

First we show that in the ℓ-th phase the expected reward of the best arm in S_ℓ drops by at most ε_ℓ.

Lemma 11 *For the* Median Elimination(ε, δ) *algorithm we have that for every phase ℓ:*

$$\mathbf{P}[\max_{j \in S_\ell} p_j \leq \max_{i \in S_{\ell+1}} p_i + \varepsilon_\ell] \geq 1 - \delta_\ell.$$

Proof Without loss of generality we look at the first round and assume that p_1 is the reward of the best arm. We bound the failure probability by looking at the event $E_1 = \{\hat{p}_1 < p_1 - \varepsilon_1/2\}$, which is the case that the empirical estimate of the best arm is pessimistic. Since we sample sufficiently, we have that $\mathbf{P}[E_1] \leq \delta_1/3$.

In case E_1 does not hold, we calculate the probability that an arm j which is not an ε_1-optimal arm is empirically better than the best arm.

$$\mathbf{P}\left[\hat{p}_j \geq \hat{p}_1 \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2\right] \leq \mathbf{P}\left[\hat{p}_j \geq p_j + \varepsilon_1/2 \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2\right] \leq \delta_1/3$$

Let #bad be the number of arms which are not ε_1-optimal but are empirically better than the best arm. We have that $\mathbb{E}[\#\text{bad} \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2] \leq n\delta_1/3$. Next we apply Markov inequality to obtain,

$$\mathbf{P}[\#\text{bad} \geq n/2 \mid \hat{p}_1 \geq p_1 - \varepsilon_1/2] \leq \frac{n\delta_1/3}{n/2} = 2\delta_1/3.$$

Using the union bound gives us that the probability of failure is bounded by δ_1. ■

Next we prove that arm sample complexity is bounded by $O((n/\varepsilon^2)\log(1/\delta))$.

Lemma 12 *The sample complexity of the* Median Elimination(ε, δ) *is* $O\left((n/\varepsilon^2)\log(1/\delta)\right)$.

Proof The number of arm samples in the ℓ-th round is $4n_\ell \log(3/\delta_\ell)/\varepsilon_\ell^2$. By definition we have that

1. $\delta_1 = \delta/2$; $\delta_\ell = \delta_{\ell-1}/2 = \delta/2^\ell$
2. $n_1 = n$; $n_\ell = n_{\ell-1}/2 = n/2^{\ell-1}$
3. $\varepsilon_1 = \varepsilon/4$; $\varepsilon_\ell = \frac{3}{4}\varepsilon_{\ell-1} = \left(\frac{3}{4}\right)^{\ell-1}\varepsilon/4$

Therefore we have

$$\begin{aligned}
\sum_{\ell=1}^{\log_2(n)} \frac{n_\ell \log(3/\delta_\ell)}{(\varepsilon_\ell/2)^2} &= 4\sum_{\ell=1}^{\log_2(n)} \frac{n/2^{\ell-1}\log(2^\ell 3/\delta)}{((\frac{3}{4})^{\ell-1}\varepsilon/4)^2} \\
&= 64\sum_{\ell=1}^{\log_2(n)} n(\frac{8}{9})^{\ell-1}(\frac{\log(1/\delta)}{\varepsilon^2} + \frac{\log(3)}{\varepsilon^2} + \frac{\ell\log(2)}{\varepsilon^2}) \\
&\leq 64\frac{n\log(1/\delta)}{\varepsilon^2}\sum_{\ell=1}^{\infty}(\frac{8}{9})^{\ell-1}(\ell C' + C) = O(\frac{n\log(1/\delta)}{\varepsilon^2})
\end{aligned}$$

■

Now we can prove Theorem 10.

Proof From Lemma 12 we have that the sample complexity is bounded by $O\left(n\log(1/\delta)/\varepsilon^2\right)$. By Lemma 11 we have that the algorithm fails with probability δ_i in each round so that over all rounds the probability of failure is bounded by $\sum_{i=1}^{\log_2(n)}\delta_i \leq \delta$. In each round we reduce the optimal reward of the surviving arms by at most ε_i so that the total error is bounded by $\sum_{i=1}^{\log_2(n)}\varepsilon_i \leq \varepsilon$. ■

4. Learning in MDPs

In this section we consider algorithms for the RL problem, which are based on the MAB algorithms presented above. We start from model-based learning in Section 4.1, where the parameters of the models are learned. We describe algorithms which are based on the successive elimination algorithm and provide stopping conditions for these algorithms. In Section 4.2 we consider model-free learning and suggest a version of the Q-learning algorithm that incorporates action elimination and stopping conditions. In Section 4.3 we analyze the batched sampling setting of Kearns and Singh (2002) and provide a mechanism that can use any (ε,δ)-MAB algorithm to enhance the performance of the Phased Q-learning introduced in Kearns and Singh (2002). We also provide a matching lower bound.

4.1 Model-Based Learning

In this section we focus on model-based learning. In model-based methods, we first learn the model, i.e., estimate the immediate reward and the next state distribution. Then by either value iteration, policy iteration, or linear programming on the learned (empirical) model, we find the exact optimal policy for the empirical model. If enough exploration is done, this policy is almost optimal for the true model. We note that there is an inherent difference between the finite horizon and the infinite discounted return. Technically, the finite horizon return is simpler than the discounted return, as one can apply the concentration inequality directly. We provide model-based algorithms for both cases.

4.1.1 FINITE HORIZON

Let us first recall the classical optimality equations for finite horizon:

$$\begin{aligned} V^H(s) &= \max_a \{R(s,a) + \mathbb{E}_{s'}[V^{H-1}(s')]\}, \quad H > 0 \\ V^0(s) &= \max_a R(s,a), \end{aligned}$$

where $V^H(s)$ is the optimal value function for horizon H. We often abuse notation by using $\mathbb{E}_{s'}$ instead of $\mathbb{E}_{s',a}$. Given the empirical model by time t we define the upper estimate $\overline{V}_\delta$, which will be shown to satisfy for every horizon k and every state s, $\overline{V}^k_\delta(s) \geq V^k(s)$ with high probability. For horizon H we define:

$$\overline{V}^H_\delta(s) = \max_a \left\{ \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[\overline{V}^{H-1}_\delta(s')] + HR_{max}\sqrt{\frac{\ln(\frac{c|S||A|H^2}{\delta})}{|T^{s,a}|}} \right\}, \quad H > 0 \tag{2}$$

$$\overline{V}^0_\delta(s) = \max_a \left\{ \hat{R}(s,a) + R_{max}\sqrt{\frac{\ln(\frac{c|S||A|}{\delta})}{|T^{s,a}|}} \right\}, \tag{3}$$

for some constant $c \geq 4$. Similarly to the upper bound $\overline{V}^H_\delta$, a lower bound may be defined where the R_{max} is replaces by $-R_{max}$. We call this estimate the lower estimate $\underline{V}^H_\delta$. The following Lemma proves that $\overline{V}^H_\delta$ is an upper estimation for any horizon and that $\underline{V}^H_\delta$ is a lower estimation.

Theorem 13 *We have that $\overline{V}^k_\delta(s) \geq V^k(s) \geq \underline{V}^k_\delta(s)$ for all states s and horizons k, with probability at least $1-\delta$.*

Proof We prove the claim by induction. For the base of the induction, by a simple use of Hoeffding inequality we have that for every state s $\overline{V}^0_\delta(s) \geq \max_a \hat{R}(s,a)$ with probability $1-\delta/(c|S||A|)$. Next we assume that the claim holds for $i \leq k$ and prove for $k+1$ and for every action a. By definition $\overline{V}^{k+1}_\delta(s)$ satisfies for every a that

$$\begin{aligned} \overline{V}^{k+1}_\delta(s) &\geq \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[\overline{V}^k_\delta(s')] + (k+1)R_{max}\sqrt{\frac{\ln(\frac{c|S||A|(k+1)^2}{\delta})}{|T^{s,a}|}} \\ &\geq \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[V^k(s')] + (k+1)R_{max}\sqrt{\frac{\ln(\frac{c|S||A|(k+1)^2}{\delta})}{|T^{s,a}|}}, \end{aligned}$$

where the second inequality follows from the inductive hypothesis. Note that V^k is not a random variable, so we can bound the last expression using Hoeffding's inequality. We arrive at:

$$\begin{aligned} \mathbf{P}\left\{ \hat{R}(s,a) + \hat{\mathbb{E}}_{s'}[V^k(s')] + (k+1)R_{max}\sqrt{\frac{\ln(\frac{c|S||A|(k+1)^2}{\delta})}{|T^{s,a}|}} < R(s,a) + \mathbb{E}_{s'}[V^k(s')] \right\} \\ \leq e^{\frac{-\ln(\frac{c|S||A|(k+1)^2}{\delta})|T^{s,a}|\left(\frac{(k+1)R_{max}}{\sqrt{|T^{s,a}|}}\right)^2}{((k+1)R_{max})^2}} = \frac{\delta}{c|S||A|(k+1)^2}. \end{aligned}$$

Therefore, we have that with high probability the following holds

$$\begin{aligned}\overline{V}_{\delta}^{k+1}(s) &\geq \max_a\{R(s,a)+\hat{\mathbb{E}}_{s'}[V^k(s')]+kR_{max}\sqrt{\frac{\ln(\frac{c|S||A|k^2}{\delta})}{|T^{s,a}|}}\}\\ &\geq \max_a\{R(s,a)+\mathbb{E}_{s'}[V^k(s')]\}\\ &= V^{k+1}(s).\end{aligned}$$

Using the union bound over all state-action pairs and all finite horizons k, we obtain that the failure probability is bounded by $\delta/2$ for $c \geq 4$. Repeating the same argument for the lower estimate and applying the union bound completes the proof. ■

Consequently, a natural early stopping condition is to stop sampling when $\|\overline{V}^H - \underline{V}^H\|_\infty < \varepsilon$. We do not provide an algorithm here, however a detailed algorithm will be given in the following subsection.

4.1.2 DISCOUNTED RETURN - INFINITE HORIZON

In this subsection, we provide upper and lower estimates of the value function V for the infinite horizon case. The optimal value is the solution of the set of the equations:

$$V^*(s) = \max_a\{R(s,a)+\gamma\mathbb{E}_{s'}[V^*(s')]\}, \quad s \in S.$$

As in Subsection 4.1.1, we provide an upper value function $\overline{V}_\delta$, which satisfies with high probability $\overline{V}_\delta(s) \geq V^*(s)$. We define $\overline{V}_\delta^t$ at time t as the solution of the set of equations:

$$\overline{V}_\delta^t(s) = \max_a\left\{\hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta^t(s')]+V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|)}}\right\}$$

for some positive constant c and $\overline{Q}_\delta^t$ as:

$$\overline{Q}_\delta^t(s,a) = \hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta(s')]+V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|}}.$$

Similarly, we define $\underline{V}_\delta^t$ and $\underline{Q}_\delta^t$ as:

$$\underline{V}_\delta^t(s) = \max_a\left\{\hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}\underline{V}_\delta(s')-V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|)}}\right\}$$

$$\underline{Q}_\delta^t(s,a) = \hat{R}(s,a)+\gamma\hat{\mathbb{E}}_{s'}[\underline{V}_\delta^t(s')]-V_{max}\sqrt{\frac{\ln(\frac{ct^2|S|\,|A|}{\delta})}{|T^{s,a}|}}.$$

The next lemma shows that with high probability the upper and lower estimations are indeed correct.

Lemma 14 *With probability at least* $1-\delta$ *we have that* $\overline{Q}_\delta^t(s,a) \geq Q^*(s,a) \geq \underline{Q}_\delta^t(s,a)$ *for every state* s*, action* a *and time* t*.*

Proof Suppose we run a value iteration algorithm on the empirical model at time t. Let $\overline{V}_\delta^{t,k}$ be the kth iteration of the value function algorithm at time t, and let $\overline{Q}_\delta^{t,k}$ be the associated Q-function, that is

$$\overline{Q}_\delta^{t,k}(s,a) = \hat{R}(s,a) + \gamma \hat{\mathbb{E}}_{s'}[\overline{V}_\delta^{t,k}(s')] + V_{max}\sqrt{\frac{\ln(\frac{ct^2|S||A|}{\delta})}{|T^{s,a}|}}.$$

Assume that we start with $\overline{V}_\delta^{t,0} = V^*$. (The use of V^* is restricted to the proof and not used in the algorithm.) We need to prove that $\overline{Q}_\delta^t(s,a) \geq Q^*(s,a)$ for every s and a. Note that since the value iteration converges, $\overline{Q}_\delta^{t,k}$ converges to $\overline{Q}_\delta^t$. We prove by induction on the number of the iterations that by taking $\overline{V}_\delta^{t,0} = V^*$, with high probability for every k we have that $\overline{Q}_\delta^{t,k} \geq \overline{Q}_\delta^{t,k-1}$, i.e., $\mathbf{P}[\forall k\, \overline{Q}_\delta^k \geq \overline{Q}_\delta^{k-1}] \geq 1 - \frac{\delta}{ct^2}$. For the basis, since V^* is not a random variable we can apply Hoeffding's inequality and obtain that for every state action pair (s,a)

$$\begin{aligned}\mathbf{P}\Big\{\hat{R}(s,a) + \gamma\hat{\mathbb{E}}_{s'}[V^*(s')] + V_{max}\sqrt{\frac{\ln(\frac{ct^2|S||A|}{\delta})}{|T^{s,a}|}} < R(s,a) + \gamma \mathbb{E}_{s'}[V^*(s')]\Big\} \\ \leq e^{-\ln(\frac{ct^2|S||A|}{\delta})} = \frac{\delta}{ct^2|S||A|}.\end{aligned}$$

Since $\overline{V}_\delta^{t,0}(s) = V^*$ we have that $\overline{Q}_\delta^{t,1}(s,a) = \hat{R}(s,a) + \gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta^{t,0}(s')] + V_{max}\sqrt{\frac{\ln(\frac{ct^2|S||A|}{\delta})}{|T^{s,a}|}}$. Therefore, $\overline{Q}_\delta^{t,1} \geq \overline{Q}_\delta^{t,0}$ with probability $1 - \frac{\delta}{ct^2}$. For the induction step, we assume that the claim holds for $i < k$ and prove for k.

$$\overline{Q}_\delta^{t,k}(s,a) - \overline{Q}_\delta^{t,k-1}(s,a) \quad = \quad \gamma\hat{\mathbb{E}}_{s'}[\overline{V}_\delta^{t,k-1}(s') - \overline{V}_\delta^{t,k-2}(s')].$$

Since $\overline{V}_\delta^{t,k-1}(s') = \max_a \overline{Q}_\delta^{t,k-1}(s',a)$ we have by the induction that for every s,

$$V_\delta^{t,k-1}(s) = \max_a \overline{Q}_\delta^{t,k-1}(s,a) \geq \max_a \overline{Q}_\delta^{t,k-2}(s,a) = V_\delta^{t,k-2}(s).$$

So that $\overline{Q}_\delta^{t,k} - \overline{Q}_\delta^{t,k-1} \geq 0$. We conclude that $\mathbf{P}[\overline{Q}_\delta \geq Q^*] \geq 1 - \frac{\delta}{ct^2}$. Repeating the same argument for the lower estimate, $\underline{Q}_\delta$, and applying the union bound over both and over all times completes the proof for the appropriate c. ■

The AE procedure is demonstrated in the following algorithm, which also supplies a stopping condition for sampling the model and eliminates actions when they are sub-optimal with high probability.

Input : MDP M, $\varepsilon > 0$, $\delta > 0$
Output : A policy for M
Choose arbitrarily an initial state s_0, let $t = 0$,
and let $U_0 = \{(s,a) | s \in S, a \in A\}$
repeat
  At state s_t perform any action a s.t. $(s_t, a) \in U_t$
  Receive a reward r_t, and a next state s_{t+1}
  Compute, $\overline{Q}_\delta, \underline{Q}_\delta$ from all the samples
  $t = t+1$
  $U_t = \{(s,a) | \overline{Q}_\delta(s,a) \geq \underline{V}_\delta(s)\}$
until $\forall (s,a) \in U \quad |\overline{Q}_\delta(s,a) - \underline{Q}_\delta(s,a)| < \frac{\varepsilon(1-\gamma)}{2}$;
return *Greedy*$(\underline{Q}_\delta)$

Algorithm 5: Model-Based AE Algorithm

A direct corollary from Lemma 14, is a stopping time condition to the Model-Based algorithm using the following Corollary.

Corollary 15 *[Singh and Yee (1994)] If $\tilde{Q}$ is a function such that $|\tilde{Q}(s,a) - Q^*(s,a)| \leq \varepsilon$ for all $s \in S$ and $a \in A$. Then for all s*

$$V^*(s) - V^{\tilde{\pi}}(s) \leq \frac{2\varepsilon}{1-\gamma},$$

where $\tilde{\pi} = Greedy(\tilde{Q})$.

Theorem 16 *Supposed the Model-Based AE Algorithm terminates. Then the policy, π, the algorithm returns is ε-optimal with probability at least $1-\delta$.*

Proof By Lemma 14 we know that with probability at least $1-\delta$ for every s, a and time t we have that $\underline{Q}_\delta(s,a) \leq Q^*(s,a) \leq \overline{Q}_\delta(s,a)$. Therefore, with probability of at least $1-\delta$ the optimal action has not been eliminated in any state in any time t. Furthermore, any action b in state s that has not been eliminated satisfies $Q^*(s,b) - \underline{Q}_\delta(s,b) \leq \overline{Q}_\delta(s,b) - \underline{Q}_\delta(s,b) \leq \varepsilon(1-\gamma)/2$. The result follows by Corollary 15. ■

4.2 Model-Free Learning

In this section we describe a model-free algorithm. We use two functions $\underline{Q}^t$ and $\overline{Q}^t$, which provide lower and upper estimations on Q^*, respectively. We use these functions to derive an asynchronous algorithm, which eliminates actions and supplies stopping condition. This algorithm requires space which is proportional to the space used by Q-learning and converges under the same conditions. Let us first recall the Q-learning algorithm (Watkins, 1989). The Q-learning algorithm estimates the state-action value function (for discounted return) as follows:

$$\begin{aligned} Q^0(s,a) &= 0, \\ Q^{t+1}(s,a) &= (1-\alpha_t(s,a))Q^t(s,a) + \alpha_t(s,a)(r_t(s,a) + \gamma V^t(s')), \end{aligned}$$

where s' is the state reached from state s when performing action a at time t, and $V^t(s) = \max_a Q^t(s,a)$. Set $\alpha_t(s,a) = 1/\#(s,a,t)$ for $t \in T^{s',a'}$ and 0 otherwise. [1] We define the upper estimation process as:

$$\begin{aligned}
\overline{Q}^0_\delta(s,a) &= V_{max} \ln\left(\frac{c|S||A|}{\delta}\right), \\
\overline{Q}^{t+1}_\delta(s,a) &= (1-\alpha_t(s,a))\overline{Q}^t_\delta(s,a) + \alpha_t(s,a)\Big(R(s,a) + \gamma \overline{V}^t_\delta(s') + \beta(\#(s,a,t))\Big),
\end{aligned}$$

where $c > 4$ and s' is the state reached from state s when performing action a at time t, $\overline{V}^t_\delta(s) = \max_a \overline{Q}^t_\delta(s,a)$ and the function β, which maintains the upper estimate interval is defined as:

$$\beta(k) = k\left(\sqrt{k\ln(ck^2|S||A|/\delta)} - (1-1/k)\sqrt{(k-1)\ln(c(k-1)^2|S||A|/\delta)}\right) V_{max}.$$

Analogously, we define the lower estimate $\underline{Q}_\delta$ as :

$$\begin{aligned}
\underline{Q}^0_\delta(s,a) &= -V_{max} \ln\left(\frac{c|S||A|}{\delta}\right), \\
\underline{Q}^{t+1}_\delta(s,a) &= (1-\alpha_t(s,a))\underline{Q}^t_\delta(s,a) + \alpha_t(s,a)\Big(R(s,a) + \gamma \underline{V}^t_\delta(s') - \beta(\#(s,a,t))\Big),
\end{aligned}$$

where $\underline{V}_\delta(s) = \max_a \underline{Q}_\delta(s,a)$. We claim that these processes converge almost surely to Q^*. (The proof appears in Appendix A.)

Proposition 17 *If every state-action pair is performed infinitely often then the upper (lower) estimation process, $\overline{Q}^t_\delta$ ($\underline{Q}^t_\delta$), converges to Q^* with probability one.*

The following Proposition claims that $\overline{Q}^t_\delta$ upper bounds Q^* and $\underline{Q}^t_\delta$ lower bounds Q^* with high probability.

Proposition 18 *With probability at least $1-\delta$ we have that for every state action pair (s,a) and time t:*

$$\overline{Q}^t_\delta(s,a) \geq Q^*(s,a) \geq \underline{Q}^t_\delta(s,a).$$

Proof We start by defining disjoints events such that their union is the event of $\overline{Q}$ not always being an upper bound of Q^*. Let

$$\begin{aligned}
E_{k,s,a} = \{&\text{The first time for which } \overline{Q} \text{ is not an upper bound of} \\
&Q^* \text{ is when } a \text{ is performed at state } s \text{ at the } k\text{th time}\}.
\end{aligned}$$

Note that if $\overline{Q}$ does not upper bound Q^* it implies that one of the events $E_{k,s,a}$ occurred. Next we bound the probability that an event $E_{k,s,a}$ happens. Note that the only Q value that has changed where a was performed art the kth time at state s is $\overline{Q}(s,a)$. We let t' be the time of $E_{k,s,a}$ and note that $\overline{Q}^t(s',a') \geq Q^*(s,a)$ for any $t < t'$.

$$\mathbf{P}(E_{k,s,a}) = \mathbf{P}\left(\overline{Q}^{t'}(s,a) - Q^*(s,a) < 0\right)$$

1. This particular learning rate is especially convenient, since the recurrence $X_t = (1-1/t)X_{t-1} + (1/t)\theta_t$ has the solution $X_t = (1/t)\sum_{i=1}^t \theta_i$.

Input : MDP M, $\varepsilon > 0$, $\delta > 0$
Output : A policy for M
For every state action (s,a):
$\overline{Q}(s,a) = \quad V_{max} \ln(\frac{c|S||A|}{\delta})$
$\underline{Q}(s,a) = -V_{max} \ln(\frac{c|S||A|}{\delta})$
#(s,a) = 1
Choose an arbitrary initial state s
repeat
  Let $U(s) = \{a | \overline{Q}(s,a) \geq \underline{V}(s)\}$
  choose arbitrarily action $a \in U(s)$, perform it and observe the next state s'
  $\overline{Q}(s,a) := (1 - \frac{1}{\#(s,a)})\overline{Q}(s,a) + \frac{1}{\#(s,a)}\left(R(s,a) + \gamma\overline{V}(s') + \beta(\#(s,a))\right)$
  $\underline{Q}(s,a) := (1 - \frac{1}{\#(s,a)})\underline{Q}(s,a) + \frac{1}{\#(s,a)}\left(R(s,a) + \gamma\underline{V}(s') - \beta(\#(s,a))\right)$
  $\#(s,a) := \#(s,a) + 1$; $s = s'$
until $\forall s \in S \ \forall a \in U(s) \ \ |\overline{Q}(s,a) - \underline{Q}(s,a)| < \frac{\varepsilon(1-\gamma)}{2}$;
return *Greedy*$(\underline{Q})$

Algorithm 6: Model-Free AE Algorithm

$$
\begin{aligned}
&= \mathbf{P}\left(\frac{1}{k}\sum_{i=1}^{k}(r_i + \gamma\overline{V}^{t_i}(s_i) + \beta(i)) - Q^*(s,a) < 0\right) \\
&\leq \mathbf{P}\left(\frac{1}{k}\sum_{i=1}^{k}(r_i + \gamma V^*(s_i) + \beta(i)) - Q^*(s,a) < 0\right) \\
&\leq \frac{\delta}{c|S||A|k^2},
\end{aligned}
$$

where we could apply Hoeffding's inequality since V^* is not a random variable. Now taking the union bound over all pairs (s,a) and times k completes the proof for the upper estimate. A similar argument for the lower estimate completes the proof. ■

We combine the upper and lower estimates to an algorithm, which eliminates sub-optimal actions whenever possible. Furthermore, the algorithm supplies a stopping condition that assures a near optimal policy. The model free AE algorithm is described in Algorithm 6.

A direct corollary from Proposition 18 is a stopping condition to the model free AE algorithm. The following corollary follows from Corollary 15 and its proof is similar to the proof of Theorem 16.

Corollary 19 *Suppose the Model-Free AE Algorithm terminates. Then the policy, it returns is* ε*-optimal with probability at least* $1 - \delta$.

4.3 MAB Phased Q-learning Algorithm

In contrast to previous sections concerning learning in MDPs, we restrict the setup in this section. In this limited setup we can fully exploit the connection between the MAB problem and learning

in MDPs. The setup is that of parallel sampling where the decision maker can sample every state and action pair, as opposed to the typical Q-learning setup where a single trajectory is followed. We will focus on the phased Q-learning algorithm Kearns and Singh (2002) which partitions the learning to phases. We will use a MAB black-box to perform the updates for each state and phase of the phased Q-learning algorithm. Although the parallel sampling model is not a realistic model it is often considered in theory as a relaxation of the MDP model which still captures many important aspects of the original problem; see Kearns and Singh (2002), Szepesvri and Munos (2005). The parallel sampling model can represent a situation where sampling from different states is very cheap (for example, when a simulator is available), so there is no real need to follow a single trajectory. In this case, reducing the number of samples needed for finding an optimal (or approximately optimal) policy is the main concern.

In phased Q-learning the value of $V_k(s)$ is fixed during the kth phased and updated only at the end of the phase. This implies that for every state and action (s,a) we can define a random variable $Y_s(a)$ whose value is $R(s,a)+\gamma V_k(s')$, where $R(s,a)$ is the random variable representing the reward and s' is distributed using $P^a_{s,s'}$.

Our aim is to find, at each state, the action that maximizes the expected reward, and estimate its expected reward, where the rewards are $Y_s(a)$. The phased Q-Learning can now be viewed as using the naive algorithm for the MAB problem (Algorithm 1) in order to find the best arm. In the following we show how, using more sophisticated MAB algorithms, one can improve the convergence rate of the Phased Q-Learning.

Our algorithm uses any (ε,δ)-PAC Multi-armed bandit algorithm as a black box in the learning process. In order to use the MAB algorithm B as, a black box, we define a simple interface, which requires the following procedures:

- $Init_B(\varepsilon,\delta)$ - Initialize the parameters of B.
- $GetArm_B()$ - returns the arm a that B wants to sample next.
- $Update_B(a,r)$ - informs B the latest reward r of arm a.
- $Stop_B(a,v)$ - returns TRUE if B terminates, and in such a case a is the output of B and v is its estimated value. (We assume that on termination, with probability at least $1-\delta$, the arm a is an ε-optimal arm and $|r^*-v|\leq\varepsilon$.)

The MAB Phased Q-learning algorithm uses as a black box, an algorithm B for the MAB problem. It receives as input (ε,δ) and returns a policy π which is ε-optimal with probability at least $1-\delta$.

Suppose that we have some (ε,δ)-PAC MAB algorithm B, and assume B has arm sample complexity $T_B(\varepsilon,\delta)$. Namely, with probability $1-\delta$, algorithm B terminates after at most $T_B(\varepsilon,\delta)$ and outputs a policy π which is ε-optimal. The following theorem computes the sample complexity of MAB Phased Q-Learning algorithm as a function of T_B.

Theorem 20 *Assume B is an $(\hat{\varepsilon},\hat{\delta})$-PAC multi-armed bandit algorithm. Then the MAB Phased Q-Learning(ε,δ) algorithm outputs a policy π which is ε-optimal policy with probability at least $1-\delta$, and has sample complexity of*

$$T(\varepsilon,\delta)=|S|T_B(\hat{\varepsilon},\hat{\delta})\log_\gamma(\frac{\hat{\varepsilon}}{2V_{max}})=O\left(\frac{|S|}{1-\gamma}\ln(\frac{V_{max}}{(1-\gamma)\varepsilon})T_B\left(\frac{\varepsilon(1-\gamma)^2}{2},\frac{\delta(1-\gamma)}{|S|\ln(V_{max}/\varepsilon)}\right)\right).$$

Input : $\varepsilon, \delta > 0$ and B a multi-armed bandit algorithm
Output : A policy
Let $\hat{\varepsilon} = \frac{\varepsilon(1-\gamma)^2}{2}$; $n = \log_\gamma(\hat{\varepsilon}/2V_{max}) = O(\ln(\frac{V_{max}}{(1-\gamma)\varepsilon})/(1-\gamma))$; $\hat{\delta} = \frac{\delta}{|S|n}$;
Initialize for every $s \in S$: $V_0(s) = 0$;
for $i = 1 : n$ **do**
  foreach $s \in S$ **do**
    $Init_B(\hat{\varepsilon}, \hat{\delta})$;
    repeat
      $a = GetArm_B()$;
      $(s', r) = sample(s, a)$;
      $r' = r + \gamma V_i(s')$;
      $Update_B(a, r')$;
    until $Stop(a, v) = TRUE$;
    $V_{i+1}(s) = v$; $\pi(s) = a$;
  end
end

Algorithm 7: MAB Phased Q-Learning algorithm

First we show that in each phase the norm $\|V^* - V\|_\infty$ decreases.

Lemma 21 *Assume B is an $(\hat{\varepsilon}, \hat{\delta})$-PAC multi-armed bandit algorithm, and consider the MAB Phased Q-Learning(ε, δ) algorithm using B. Then with probability at least $1 - \delta$, for all $k \leq n$, $\|V^* - V_k\|_\infty$ is bounded by $\frac{\hat{\varepsilon}}{1-\gamma} + V_{max}\gamma^k$.*

Proof First we bound the probability that B outputs an arm which is not ε-optimal. We bound the failure probability by using the union bound on all the invocations of B. There are

$$|S|n = |S| \log_\gamma(\hat{\varepsilon}/V_{max}) = O\left(\frac{|S| \ln(\frac{V_{max}}{(1-\gamma)\varepsilon})}{1-\gamma}\right)$$

initializations of algorithm B and for each invocation the failure probability is bounded by $\delta/|S|n$. Thus, the failure probability is at most δ.

Next, we show that the error contracts in every phase. We compare the value vector, V_k, with the standard value iteration value vector $\hat{V}_k$ for the case of a known model (at the end of the k-th step). Formally,

$$\hat{V}_{k+1}(s) = \max_u \{\mathbb{E}[R(s,u)] + \gamma \mathbb{E}_{s'}[\hat{V}_k(s')]\},$$

where s' is distributed according to $P^u_{s,s'}$ and $\hat{V}_0 = 0$.

We show by induction on the number of phases, that $d_k = \|V_k - \hat{V}_k\|_\infty \leq \frac{\hat{\varepsilon}}{1-\gamma}$. The base of the induction, $t = 0$, for every state s we have $d_0 = |V_0(s) - \hat{V}_0(s)| = 0$. We assume that the induction assumption holds for $t < k$ and prove for k. Let $m_{s,a}$ denote the number of times the state action pair (s,a) was sampled in the k-th iteration.

$$|V_k(s) - \hat{V}_k(s)| \quad = \quad \left| \max_u [\frac{1}{m_{s,u}} \sum_{i=1}^{m_{s,u}} r(s,u) + \gamma V_{k-1}(s'_i)]\right.$$

$$
\begin{aligned}
& \left. - \max_a [\mathbb{E}[R(s,a)] + \gamma \sum_{s'} P^a_{s,s'} \hat{V}_{k-1}(s')] \right| \\
\leq & \max_{\rho \in \{-\hat{\varepsilon}, \hat{\varepsilon}\}} \left| \max_u [\mathbb{E}[R(s,u)] + \gamma \sum_{s'} P^u_{s,s'} V_{k-1}(s')] + \rho \right. \\
& \left. - \max_a [\mathbb{E}[R(s,a)] + \gamma \sum_{s'} P^a_{s,s'} \hat{V}_{k-1}(s')] \right| \\
\leq & \hat{\varepsilon} + \max_a \left| \gamma \sum_{s'} P^a_{s,s'} (V_{k-1}(s') - \hat{V}_{k-1}(s')) \right| \\
\leq & \hat{\varepsilon} + \gamma d_{k-1} \\
\leq & \hat{\varepsilon} + \gamma (\frac{\hat{\varepsilon}}{1-\gamma}) = \frac{\hat{\varepsilon}}{1-\gamma}.
\end{aligned}
$$

To conclude the proof note that for the value iteration we have that $\|\hat{V}_k - V^*\|_\infty \leq \gamma^k V_{max}$, where $\hat{V}_0 = 0$ (see, e.g., Bertsekas and Tsitsiklis, 1995). ■

Lemma 22 *When the MAB Phased Q-Learning algorithm terminates, the policy π it returns is ε-optimal with probability at least $1-\delta$.*

Proof By Lemma 21 we have that with probability at least $1-\delta$ the difference $\|V_k - V^*\|_\infty \leq \frac{\hat{\varepsilon}}{1-\gamma} + V_{max}\gamma^k$. Since $\hat{\varepsilon} = \varepsilon(1-\gamma)^2/2$, we have that $\|V_k - V^*\|_\infty \leq \varepsilon(1-\gamma)/2 + V_{max}\gamma^k$. The lemma follows from our choice of $n = \log_\gamma(\varepsilon(1-\gamma)/2V_{max})$. ■

We can now complete the proof Theorem 20.

Proof The correctness follows from Lemma 22. We bound the sample complexity as follows. By definition, the MAB Phased Q-Learning algorithm samples at each state and action during every phase $T_B(\hat{\varepsilon}, \hat{\delta})$. By definition of the algorithm, the number of phases is $n = O(\ln(V_{max}/\hat{\varepsilon})/(1-\gamma))$, and each phase is composed from $|S|$ MAB instances. This completes the bound on the sample complexity. ■

Applying the multi-armed bandit algorithms described in the previous sections we derive the following corollary. We show that by using the *median elimination* algorithm, the arm sample complexity can be reduced by a factor of $\log(|A|)$.

Corollary 23 *Let B be the median elimination algorithm. MAB Phased Q-Learning algorithm has sample complexity*

$$
T(\varepsilon, \delta) = O\left(\frac{|S|\ |A| V_{max}^2}{(1-\gamma)^5 \varepsilon^2} \ln(\frac{V_{max}}{(1-\gamma)\varepsilon}) \ln(\frac{|S| \ln(V_{max}/\varepsilon)}{\delta(1-\gamma)}) \right).
$$

Next we introduce an almost matching lower bound. Let us introduce some more notation before we proceed. Let T denote the time until an RL algorithm stops (this may be in general a random number). For a given RL algorithm L and a given MDP L we denote by $\mathbb{E}^{L,M}$ the expectation with respect to randomization in both the algorithm and the MDP.

Theorem 24 *Let L be a learning algorithm for MDPs under the parallel sampling model. There are constants $C_1, C_2, \varepsilon_0, \delta_0, \gamma_0$ such that for every $\varepsilon \in (0, \varepsilon_0)$, $\delta \in (0, \delta_0)$, and $\gamma \in (0, \gamma_0)$ if L returns an ε-optimal policy with probability of at least $1-\delta$ for every MDP with discount factor γ there exist an MDP M for which:*

$$\mathbb{E}^{L,M}[T] \geq C_1 \frac{|S|\,|A|}{(1-\gamma)^2\varepsilon^2} \log\left(\frac{C_2}{\delta}\right) = \Omega\left(\frac{|S|\,|A|}{(1-\gamma)^2\varepsilon^2} \log(\frac{1}{\delta})\right) .$$

Proof Consider the following construction which was used in Theorem 1 of Mannor and Tsitsiklis (2004) for the MAB problem. A MAB problem with $|A|$ arms is given. We enumerate the arms from 0 to $|A|-1$, and fix $\hat{\varepsilon} > 0$. In each of the states one of the following hypotheses is true:

$$H_0 : r(0) = 1/2 + \hat{\varepsilon}; \qquad r(i) = 1/2 \quad (i = 1, 2, \ldots, |A-1|),$$

and for $\ell = 1, 2, \ldots, |A|-1$:

$$H_\ell : r(0) = 1/2 + \hat{\varepsilon}; \qquad r(i) = 1/2 \quad (i = 1, 2, \ldots, |A-1|, i \neq \ell); \quad r(\ell) = 1/2 + 2\hat{\varepsilon}.$$

Let $\mathbb{E}_\ell$ be the expectation given than hypothesis H_ℓ is true, and $A(s)$ the event that the algorithm errs in state s. In Lemma 4 in Mannor and Tsitsiklis (2004) it was proved that there are constants c_1 and c_2 such that for $\hat{\varepsilon} < \hat{\varepsilon}_0$ every algorithm that can identify the true hypothesis with probability $1-\hat{\delta}$ for all the hypotheses must satisfy that:

$$\mathbb{E}_0[T] \geq c_1 \frac{|A|-1}{\hat{\varepsilon}^2} \log(\frac{c_2}{\hat{\delta}}). \tag{4}$$

We create the following set of MDPs. In each possible MDP there are $|S|$ states and $|A|$ actions in each state. All the states are absorbing and have one of the above A hypotheses per state. The reward in each state behaves according to H_0 or one of the H_ℓ. (There are $|A|^{|S|}$ possible MDPs.) We set $\hat{\varepsilon} = 2(1-\gamma)\varepsilon$. We run algorithm L until termination. When L terminates it returns a policy which is ε-optimal with probability of at least $1-\delta$. Since every state is absorbing, and by our choice of $\hat{\varepsilon}$, it implies that the right hypothesis was found in all states. Note that even if L returns a randomized policy, we will determine that the action with the highest reward is the best one (this is the reason for the factor of 2 in determining $\hat{\varepsilon}$). By taking the sum of Eq. (4) over all states we obtain that

$$\mathbb{E}^{L,M}[T] \geq c_1 \frac{|A|-1}{\hat{\varepsilon}^2} |S| \log(\frac{c_2}{\delta}).$$

The result follows by an appropriate choice of constants. ■

5. Experiments

In this section we show four types of MDPs in which the number of samples used by AE procedures is significantly smaller than the number of samples used by standard Q-learning and ε-greedy Q-learning. Both model free AE algorithm and standard Q-learning choose the action in each state uniformly at random. In our experiments we focused on the steady state norm (L_1 weighted by steady state probabilities) rather than the L_∞ norm to emphasize the average behavior. We note that we use the steady state rather than the discounted steady state. We run AE Q-learning algorithm from Section 4.2 with the same input (for actions that were not eliminated) as a standard Q-learning algorithm. The following experiments were conducted:

1. **A queueing system.** The MDP represents a queueing problem that appears in Differentiated Services (Aiello et al., 2000, Kesselman et al., 2004). The basic settings are that the arriving packets have different values and they are buffered in a FIFO queue before being sent. The major constraints are that we reject or accept a packet upon its arrival (no preemption) and that the buffer has limited capacity. We have analyzed a queue of size five and three different packets values, $1, 20, 150$. In each time unit we either receive a packet or send a packet according to some distribution. We modeled the queueing problem via a discounted MDP with discount factor $\gamma = 0.99$. The AE model-free algorithm[2] was compared with ε-greedy Q-learning with epsilon varying from 0.05 to 0.2. In Figure 1 we present the results for ε which was empirically best, $\varepsilon = 0.1$. In this experiment we used a fixed step size. We focused here on the fraction of times in which optimal actions were performed and on the value function criterion. The results are demonstrated in Figure 1, in which we see that not only AE has better results but the variance in the results is much smaller in both the fraction of times that almost optimal actions were performed and in the value function. Figure 2 demonstrates the elimination rate of the AE procedure.

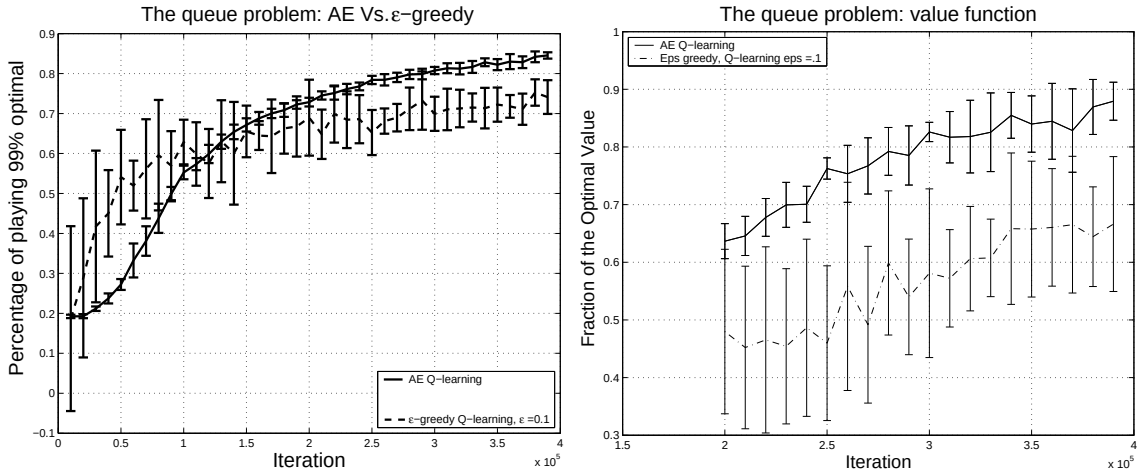


Figure 1: Example of a Queue of size 5 with three types of packets with values $1, 20, 150$. The discount factor is set to 0.99. We disregard the full queue state in which every action is optimal. We repeated each experiment 15 times and the error bars represent 1 standard deviation.

2. **Random MDPs.** Two types of random MDPs were randomly generated. In both types there were 20 states and 50 actions in each state. The first type is due to Puterman (1994) and is a sparse MDP, such that each action can reach only three states. The second type of random MDPs is dense, such that the next state distribution is randomly chosen for each state-action pair and might include all states. For both MDPs the immediate reward expectation is randomly chosen in the interval $[0, 10]$. Results of ten runs are presented by Figure 3 for the

2. Since we were interested in the short term results rather than the long term, we initialized the upper and lower values to similar values and allowed elimination only after an exploration period, we still used the β function for both the upper and lower estimates as stated in the theoretical part up to constants.

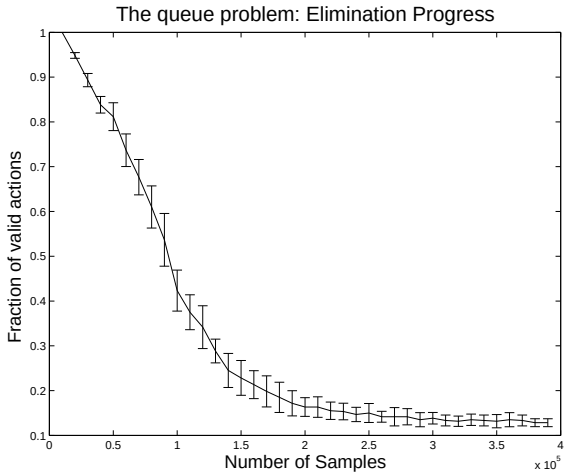


Figure 2: Example of a Queue of size 5 with three types of packets with values $1, 20, 150$. The discount factor is set to 0.99. This figure demonstrates the elimination rate. We repeated each experiment 15 times and the error bars represent 1 standard deviation.

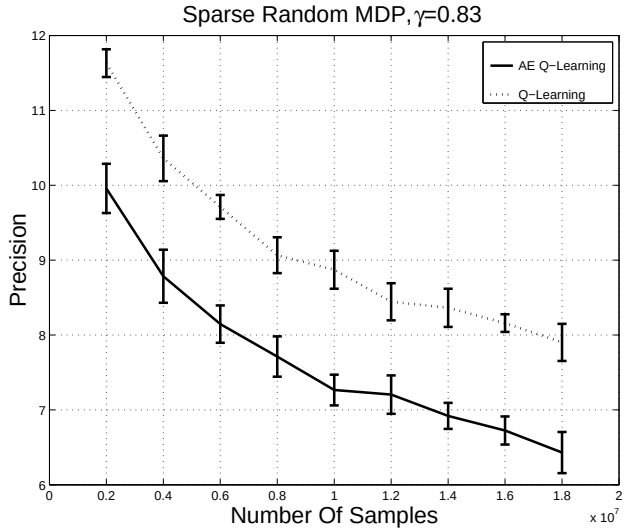


Figure 3: Example of a 20 state sparse randomly generated MDPs with 50 actions in each state, where $\gamma = 0.833$ (as in Puterman (1994).) The precision is the distance of the Q-function from the optimal Q-function. We repeated each experiment 10 times and the error bars represent 1 standard deviation.

sparse MDP, in this experiment the model free AE algorithm needs only about half the samples used by the Q-learning to achieve the same precision. The precision is measured as the distance of the Q-function from the optimal function in steady state norm. In Figure 4 for dense MDP, the results are similar. The AE algorithm required about 40% fewer samples.

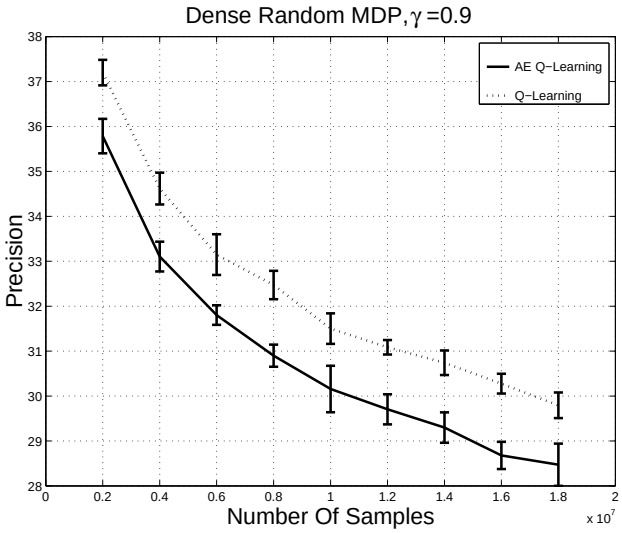


Figure 4: Example of a 20 state dense randomly generated MDPs with 50 actions in each state, $\gamma = 0.9$. The error bars represent 1 standard deviation.

3. **Howard's automobile replacement problem.** This MDP represents another realistic problem—Howard's automobile replacement problem Howard (1960). This problem contains 40 states, in each state there are 41 actions. See Howard (1960) for a detailed description. This problem was considered as a benchmark by several authors in the optimization community. We used the model free AE algorithm for this problem with discount factor $\gamma = 0.833$ against standard Q-learning and the results appear in Figure 5. A significant improvement is evident.

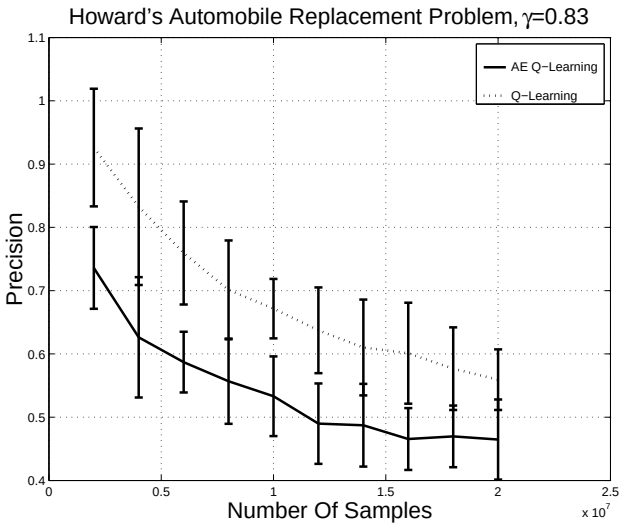


Figure 5: Example of Howard's Automobile Replacement Problem, where the discount factor, γ, is 0.833. The norm is the steady state norm. The error bars represent 1 standard deviation.

6. Future Directions

Extending the concept of action elimination to large state spaces is probably the most important direction. The extension to function approximation, which approximates the value function, requires some assumptions on the value (or Q) function approximation architecture. Following Kakade and Langford (2002) we can consider value functions that can be approximated under the infinity norm. For an example of such an algorithm see (Ormoneit and Sen (2002)). If convergence rate of the function approximation is provided, as in (Ormoneit and Sen (2002)), then an AE procedure can be derived as before.

Acknowledgements

E.E. and Y.M. was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, by a grants no. 525/00 and 1079/04 from the Israel Science Foundation and an IBM faculty award. The work was done while E.E was a graduate student at Tel Aviv University. S.M. was partially supported by the Natural Sciences and Engineering Research Council of Canada and by the Canada Research Chairs Program. We thank Alex Strehl and Csaba Szepesvari for helpful discussions. This publication only reflects the authors' views.

Appendix A. Proof of Proposition 17

In order to show the almost sure convergence of the upper and lower estimations, we follow the proof of Bertsekas and Tsitsiklis (1996). We consider a general type of *iterative stochastic algorithms*, which is performed as follows:

$$X_{t+1}(i) = (1-\alpha_t(i))X_t(i) + \alpha_t(i)((HX_t)(i) + w_t(i) + u_t(i)),$$

where w_t is a bounded random variable with zero expectation and each H is a pseudo contraction mapping (See Bertsekas and Tsitsiklis, 1996, for details).

Definition 25 *An iterative stochastic algorithm is well behaved if:*

1. *The step size $\alpha_t(i)$ satisfies (1) $\sum_{t=0}^{\infty}\alpha_t(i) = \infty$, (2) $\sum_{t=0}^{\infty}\alpha_t^2(i) < \infty$ and (3) $\alpha_t(i) \in (0,1)$.*
2. *There exists a constant A that bounds $w_t(i)$ for any history F_t, i.e., $\forall t,i:\ |w_t(i)| \le A$.*
3. *There exists $\gamma \in [0,1)$ and a vector X^* such that for any X we have $||HX - X^*|| \le \gamma||X - X^*||$, where $||\cdot||$ is any norm.*
4. *There exists a nonnegative random sequence θ_t, that converges to zero with probability 1, and is such that*
$$\forall i,t \quad |u_t(i)| \le \theta_t(||X_t|| + 1)$$

We first note that the Q-learning algorithm satisfies the first three criteria and the fourth criteria holds trivially since $u_t = 0$, thus its convergence follows if all state-action pairs are tried infinitely often (see Proposition 5.6 in Bertsekas and Tsitsiklis, 1996). The upper estimate has an additional noise term, u_t. If we show that it satisfies the fourth requirement, then the convergence will follow.

Lemma 26 *The upper estimation algorithm is well behaved.*

Proof In the convergence proof of Q-learning, it was shown that requirements 1–3 are satisfied, this implies that the upper estimates satisfies them as well. Now we let $u_t = \theta_t = c\sqrt{\frac{\ln(\#(s,a,t))}{\#(s,a,t)}}V_{max}$. It follows that θ_t converges to zero, thus

$$|u_t(i)| = \theta_t \leq \theta_t(||X_t|| + 1).$$

■

Similar result holds for the lower estimate as well.

References

W. A. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen. Competitive queue policies for differentiated services. In *INFOCOM*, 2000. (To appear in J. of Algorithms).

D. Angluin and L. G. Valiant. Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer and System Sciences*, 18:155–193, 1979.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, 1995.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM J. on Computing*, 32(1):48–77, 2002.

D. A. Berry and B. Fristedt. *Bandit Problems*. Chapman and Hall, 1985.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1995.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

V. S. Borkar and S.P Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.*, 38(2):447–469, 2000.

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25, 2003. (A preliminary version appeared in the Fourteenth Annual Conference on Computation Learning Theory (2001), 589-604.).

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

R. Howard. *Dynamic programming and Markov decision processes*. MIT press, 1960.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274. Morgan Kaufmann, 2002.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002. (A preliminary version appeared in ICML (1998), 260-268.).

M. Kearns and S. P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Neural Information Processing Systems 10*, pages 996–1002, 1998.

A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM J. on Computing*, 33(3):563–583, 2004. (A preliminary version appeared in ACM Symposium on Theory of Computing (2001), 520-529.).

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

J. MacQueen. A modified dynamic programming method for Markov decision problems. *J. Math. Anal. Appl.*, 14:38–43, 1966.

S. Mannor and J. N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004. (A preliminary version appeared in the Sixteenth Annual Conference on Computation Learning Theory (2003), 418-432.).

D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

M. Puterman. *Markov Decision Processes*. Wiley-Interscience, 1994.

H. Robbins. Some aspects of sequential design of experiments. *Bull. Amer. Math. Soc.*, 55:527–535, 1952.

S. P. Singh and R. C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

R. Sutton and A. Barto. *Reinforcement Learning*. 1998.

Cs. Szepesvri and R. Munos. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, page 881886, 2005.

C. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

Journal of Machine Learning Research 7 (2006) 1107–1133 Submitted 11/05; Published 06/06

Step Size Adaptation in Reproducing Kernel Hilbert Space

S.V. N. Vishwanathan SVN.VISHWANATHAN@NICTA.COM.AU
Nicol N. Schraudolph NIC.SCHRAUDOLPH@NICTA.COM.AU
Alex J. Smola ALEX.SMOLA@NICTA.COM.AU

Statistical Machine Learning Program
National ICT Australia
Locked Bag 8001
Canberra ACT 2601, Australia

Research School of Information Sciences and Engineering
Australian National University
Canberra ACT 0200, Australia

Editor: Thorsten Joachims

Abstract

This paper presents an online support vector machine (SVM) that uses the stochastic meta-descent (SMD) algorithm to adapt its step size automatically. We formulate the online learning problem as a stochastic gradient descent in reproducing kernel Hilbert space (RKHS) and translate SMD to the nonparametric setting, where its gradient trace parameter is no longer a coefficient vector but an element of the RKHS. We derive efficient updates that allow us to perform the step size adaptation in linear time. We apply the online SVM framework to a variety of loss functions, and in particular show how to handle structured output spaces and achieve efficient online multiclass classification. Experiments show that our algorithm outperforms more primitive methods for setting the gradient step size.

Keywords: online SVM, stochastic meta-descent, structured output spaces

1. Introduction

Stochastic ("online") gradient methods incrementally update their hypothesis by descending a stochastic approximation of the gradient computed from just the current observation. Although they require more iterations to converge than traditional deterministic ("batch") techniques, each iteration is faster as there is no need to go through the entire training set to measure the current gradient. For large, redundant data sets, or continuing (potentially non-stationary) streams of data, stochastic gradient thus outperforms classical optimization methods. Much work in this area centers on the key issue of choosing an appropriate time-dependent gradient step size η_t.

Though support vector machines (SVMs) were originally conceived as batch techniques with time complexity quadratic to cubic in the training set size, recent years have seen the development of online variants (Herbster, 2001; Kivinen et al., 2004; Crammer et al., 2004; Weston et al., 2005; Kim et al., 2005) which overcome this limitation. To date, online kernel methods based on stochastic gradient descent (Kivinen et al., 2004; Kim et al., 2005) have either held η_t constant, or let it decay according to some fixed schedule. Here we adopt the more sophisticated approach of *stochastic*

meta-descent (SMD): performing a simultaneous stochastic gradient descent on the step size itself. Translating this into the kernel framework yields a fast online optimization method for SVMs.

Outline. In Section 2 we review gradient-based step size adaptation algorithms so as to motivate our subsequent derivation of SMD. We briefly survey kernel-based online methods in Section 3, then present the online SVM algorithm with a systematic, unified view of various loss functions (including losses on structured label domains) in Section 4. Section 5 then introduces online SVMD, our novel application of SMD to the online SVM. Here we also derive linear-time incremental updates and standard SVM extensions for SVMD, and discuss issues of buffer management and time complexity. Experiments comparing SVMD to the online SVM are then presented in Section 6, followed by a discussion.

2. Stochastic Meta-Descent

The SMD algorithm (Schraudolph, 1999, 2002) for gradient step size adaptation can considerably accelerate the convergence of stochastic gradient descent; its applications to date include independent component analysis (Schraudolph and Giannakopoulos, 2000), nonlinear principal component analysis in computational fluid dynamics (Milano, 2002), visual tracking of articulated objects (Bray et al., 2005, 2006), policy gradient reinforcement learning (Schraudolph et al., 2006), and training of conditional random fields (Vishwanathan et al., 2006).

2.1 Gradient-Based Step Size Adaptation

Let V be a vector space, $\boldsymbol{\theta} \in V$ a parameter vector, and $J : V \to \mathbb{R}$ the objective function which we would like to optimize. We assume that J is twice differentiable almost[1] everywhere. Denote by $J_t : V \to \mathbb{R}$ the stochastic approximation of the objective function at time t. Our goal is to find $\boldsymbol{\theta}$ such that $\mathbb{E}_t[J_t(\boldsymbol{\theta})]$ is minimized. An adaptive version of stochastic gradient descent works by setting

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\eta}_t \cdot \boldsymbol{g}_t, \quad \text{where} \quad \boldsymbol{g}_t = \partial_{\boldsymbol{\theta}_t} J_t(\boldsymbol{\theta}_t), \tag{1}$$

using $\partial_{\boldsymbol{\theta}_t}$ as a shorthand for $\frac{\partial}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$. Unlike conventional gradient descent algorithms where η_t is scalar, here $\boldsymbol{\eta}_t \in \mathbb{R}^n_+$, and $\cdot$ denotes component-wise (Hadamard) multiplication. In other words, each coordinate of $\boldsymbol{\theta}$ has its own positive step size that serves as a diagonal conditioner. Since we need to choose suitable values we shall adapt $\boldsymbol{\eta}$ by a simultaneous meta-level gradient descent.

A straightforward implementation of this idea is the *delta-delta* algorithm (Sutton, 1981; Jacobs, 1988), which updates $\boldsymbol{\eta}$ via

$$\begin{aligned} \boldsymbol{\eta}_{t+1} &= \boldsymbol{\eta}_t - \mu \partial_{\boldsymbol{\eta}_t} J_{t+1}(\boldsymbol{\theta}_{t+1}) \\ &= \boldsymbol{\eta}_t - \mu \partial_{\boldsymbol{\theta}_{t+1}} J_{t+1}(\boldsymbol{\theta}_{t+1}) \cdot \partial_{\boldsymbol{\eta}_t} \boldsymbol{\theta}_{t+1} \\ &= \boldsymbol{\eta}_t + \mu \boldsymbol{g}_{t+1} \cdot \boldsymbol{g}_t, \end{aligned} \tag{2}$$

where $\mu \in \mathbb{R}$ is a scalar meta-step size. In a nutshell, step sizes are decreased where a negative autocorrelation of the gradient indicates oscillation about a local minimum, and increased otherwise. Unfortunately such a simplistic approach has several problems:

1. Since gradient descent implements a discrete approximation to an infinitesimal (differential) process in any case, we can in practice ignore non-differentiability of J on a set of measure zero, as long as our implementation of the gradient function returns a subgradient at those points.

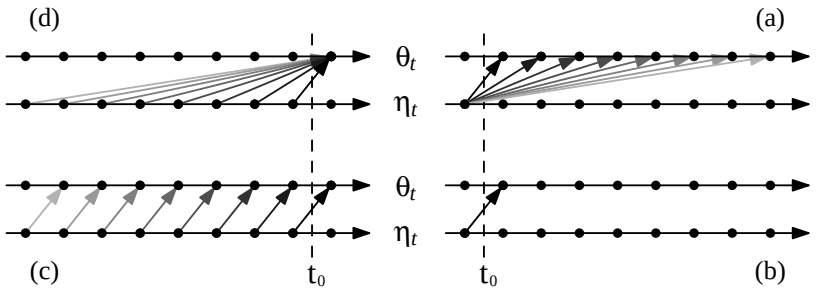


Figure 1: Dependence of a parameter θ on its step size η at time t_0. (a) Future parameter values depend on the current step size; the dependence diminishes over time due to the ongoing adaptation of η. (b) Standard step size adaptation methods capture only the immediate effect, even when (c) past gradients are exponentially smoothed. (d) SMD, by contrast, iteratively models the dependence of the current parameter on an exponentially weighted past history of step sizes, thereby capturing long-range effects. Figure adapted from Bray et al. (2005).

Firstly, (2) allows step sizes to become negative. This can be avoided by updating $\boldsymbol{\eta}$ multiplicatively, *e.g.* via *exponentiated gradient* descent (Kivinen and Warmuth, 1997).

Secondly, delta-delta's cure is worse than the disease: individual step sizes are meant to address ill-conditioning, but (2) actually squares the condition number. The auto-correlation of the gradient must therefore be normalized before it can be used. A popular (if extreme) form of normalization is to consider only the sign of the auto-correlation. Such sign-based methods (Jacobs, 1988; Tollenaere, 1990; Silva and Almeida, 1990; Riedmiller and Braun, 1993), however, do not cope well with stochastic approximation of the gradient since the non-linear sign function does not commute with the expectation operator (Almeida et al., 1999). More recent algorithms (Harmon and Baird, 1996; Almeida et al., 1999; Schraudolph, 1999, 2002) therefore use multiplicative (hence linear) normalization factors to condition the step size update.

Finally, (2) fails to take into account that changes in step size not only affect the current, but also future parameter updates (see Figure 1). In recognition of this shortcoming, $\boldsymbol{g}_t$ in (2) is usually replaced with an exponential running average of past gradients (Jacobs, 1988; Tollenaere, 1990; Silva and Almeida, 1990; Riedmiller and Braun, 1993; Almeida et al., 1999). Although such ad-hoc smoothing does improve performance, it does not properly capture long-term dependencies, the average still being one of immediate, single-step effects (Figure 1c).

By contrast, Sutton (1992) modeled the long-term effect of step sizes on future parameter values in a linear system by carrying the relevant partials forward in time, and found that the resulting step size adaptation can outperform a less than perfectly matched Kalman filter. Stochastic meta-descent (SMD) extends this approach to arbitrary twice-differentiable nonlinear systems, takes into account the full Hessian instead of just the diagonal, and applies an exponential decay to the partials being carried forward (Figure 1d).

2.2 SMD Algorithm

SMD employs two modifications to address the problems described above: it adjusts step sizes in log-space, and optimizes over an exponentially decaying trace of gradients. Thus $\log \boldsymbol{\eta}$ is updated

as follows:

$$\begin{aligned}\log \boldsymbol{\eta}_{t+1} &= \log \boldsymbol{\eta}_t - \mu \sum_{i=0}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} J(\boldsymbol{\theta}_{t+1}) \\ &= \log \boldsymbol{\eta}_t - \mu \partial_{\boldsymbol{\theta}_{t+1}} J(\boldsymbol{\theta}_{t+1}) \cdot \sum_{i=0}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} \boldsymbol{\theta}_{t+1} \\ &= \log \boldsymbol{\eta}_t - \mu \boldsymbol{g}_{t+1} \cdot \boldsymbol{v}_{t+1}, \end{aligned} \tag{3}$$

where the vector $\boldsymbol{v} \in V$ characterizes the long-term dependence of the system parameters on their past step sizes over a time scale governed by the decay factor $0 \leq \lambda \leq 1$.

Note that virtually the same derivation holds if — as will be the case in Section 5 — we wish to adapt only a single, scalar step size η_t for all system parameters; the only change necessary is to replace the Hadamard product in (3) with an inner product.

Element-wise exponentiation of (3) yields the desired multiplicative update

$$\begin{aligned}\boldsymbol{\eta}_{t+1} &= \boldsymbol{\eta}_t \cdot \exp(-\mu \boldsymbol{g}_{t+1} \cdot \boldsymbol{v}_{t+1}) \\ &\approx \boldsymbol{\eta}_t \cdot \max(\tfrac{1}{2}, 1 - \mu \boldsymbol{g}_{t+1} \cdot \boldsymbol{v}_{t+1}), \end{aligned} \tag{4}$$

where the approximation eliminates an expensive exponentiation operation for each step size update. The particular bi-linearization we use, $e^u \approx \max(\frac{1}{2}, 1+u)$,

- matches the exponential in value and first derivative at $u = 0$, and thus becomes accurate in the limit of small μ;
- ensures that all elements of $\boldsymbol{\eta}$ remain strictly positive; and
- improves robustness by reducing the effect of outliers: $u \gg 0$ leads to linear[2] rather than exponential growth in step sizes, while for $u \ll 0$ they are at most cut in half.

The choice of $\frac{1}{2}$ as the lower bound stems from the fact that a gradient descent converging on a minimum of a differentiable function can overshoot that minimum by at most a factor of two, since otherwise it would by definition be divergent. A reduction by at most $\frac{1}{2}$ in step size thus suffices to maintain stability from one iteration to the next.

To compute the gradient trace $\boldsymbol{v}$ efficiently, we expand $\boldsymbol{\theta}_{t+1}$ in terms of its recursive definition (1):

$$\begin{aligned}\boldsymbol{v}_{t+1} &:= \sum_{i=0}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} \boldsymbol{\theta}_{t+1} \\ &= \sum_{i=0}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} \boldsymbol{\theta}_t - \sum_{i=0}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} (\boldsymbol{\eta}_t \cdot \boldsymbol{g}_t) \\ &\approx \lambda \boldsymbol{v}_t - \boldsymbol{\eta}_t \cdot \boldsymbol{g}_t - \boldsymbol{\eta}_t \cdot \left[\partial_{\boldsymbol{\theta}_t} \boldsymbol{g}_t \sum_{i=0}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} \boldsymbol{\theta}_t \right] \end{aligned} \tag{5}$$

Here we have used $\partial_{\log \boldsymbol{\eta}_t} \boldsymbol{\theta}_t = 0$, and approximated

$$\sum_{i=1}^{t} \lambda^i \partial_{\log \boldsymbol{\eta}_{t-i}} \log \boldsymbol{\eta}_t \approx 0 \tag{6}$$

2. A quadratic approximation with similar properties would be $e^u \approx \begin{cases} \frac{1}{2}u^2 + u + 1 & \text{if } u > -1; \\ \frac{1}{2} & \text{otherwise.} \end{cases}$

which amounts to stating that the step size adaptation (in log space) must be in equilibrium at the time scale determined by λ. Noting that $\partial_{\boldsymbol{\theta}_t} \boldsymbol{g}_t$ is the Hessian $\boldsymbol{H}_t$ of $J_t(\boldsymbol{\theta}_t)$, we arrive at the simple iterative update

$$\boldsymbol{v}_{t+1} = \lambda \boldsymbol{v}_t - \boldsymbol{\eta}_t \cdot (\boldsymbol{g}_t + \lambda \boldsymbol{H}_t \boldsymbol{v}_t). \tag{7}$$

Since the initial parameters $\boldsymbol{\theta}_0$ do not depend on any step sizes, $\boldsymbol{v}_0 = \mathbf{0}$.

2.3 Efficiency and Conditioning

Although the Hessian $\boldsymbol{H}$ of a system with n parameters has $O(n^2)$ entries, efficient indirect methods from algorithmic differentiation are available to compute its product with an arbitrary vector within the same time as 2–3 gradient evaluations (Pearlmutter, 1994; Griewank, 2000). For non-convex systems (where positive semi-definiteness of the Hessian cannot be guaranteed) SMD uses an extended Gauss-Newton approximation of $\boldsymbol{H}$ for which a similar but even faster technique exists (Schraudolph, 2002). An iteration of SMD — comprising (1), (4), and (7) — thus consumes less than 3 times as many floating-point operations as simple gradient descent.

Iterating (7) while holding $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ constant would drive $\boldsymbol{v}$ towards the fixpoint

$$\boldsymbol{v} \to -[\lambda \boldsymbol{H} + (1-\lambda)\mathrm{diag}(1/\boldsymbol{\eta})]^{-1} \boldsymbol{g}, \tag{8}$$

which is a Levenberg-Marquardt gradient step with a trust region conditioned by $\boldsymbol{\eta}$ and scaled by $1/(1-\lambda)$. For $\lambda = 1$ this reduces to a Newton (resp. Gauss-Newton) step, which converges rapidly but may become unstable in regions of low curvature. In practice, we find that SMD performs best when λ is pushed as close to 1 as possible without losing stability.

Note that in this regime, the $\boldsymbol{g} \cdot \boldsymbol{v}$ term in (4) is approximately affine invariant, with the inverse curvature matrix in (8) compensating for the scale of the gradient auto-correlation. This means that the meta-step size μ is relatively problem-independent; in experiments we typically use values within an order of magnitude of $\mu = 0.1$. Likewise, well-adapted step sizes ($\boldsymbol{\eta} \cdot \boldsymbol{g} \approx \boldsymbol{H}^{-1} \boldsymbol{g}$) will condition the update not only of $\boldsymbol{\theta}$ (1) but also of $\boldsymbol{v}$ (7). Thus SMD maintains an adaptive conditioning of all its updates, provided it is given reasonable initial step sizes $\boldsymbol{\eta}_0$ to begin with.

3. Survey of Online Kernel Methods

The *perceptron* algorithm (Rosenblatt, 1958) is arguably one of the simplest online learning algorithms. Given a set of labeled instances $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots (\boldsymbol{x}_m, y_m)\} \subset \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subseteq \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ the algorithm starts with an initial weight vector $\boldsymbol{\theta} = \mathbf{0}$. It then predicts the label of a new instance $\boldsymbol{x}$ to be $\hat{y} = \mathrm{sign}(\langle \boldsymbol{\theta}, \boldsymbol{x} \rangle)$. If $\hat{y}$ differs from the true label y then the vector $\boldsymbol{\theta}$ is updated as $\boldsymbol{\theta} = \boldsymbol{\theta} + y\boldsymbol{x}$. This is repeated until all points are well classified. The following result bounds the number of mistakes made by the perceptron algorithm (Freund and Schapire, 1999, Theorem 1):

Theorem 1 *Let $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots (\boldsymbol{x}_m, y_m)\}$ be a sequence of labeled examples with $||\boldsymbol{x}_i|| \leq R$. Let $\boldsymbol{\theta}$ be any vector with $||\boldsymbol{\theta}|| = 1$ and let $\gamma > 0$. Define the deviation of each example as*

$$d_i = \max(0, \gamma - y_i \langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle), \tag{9}$$

and let $D = \sqrt{\sum_i d_i^2}$. Then the number of mistakes of the perceptron algorithm on this sequence is bounded by $(\frac{R+D}{\gamma})^2$.

This generalizes the original result (Block, 1962; Novikoff, 1962; Minsky and Papert, 1969) for the case when the points are strictly separable, *i.e.,* when there exists a $\boldsymbol{\theta}$ such that $||\boldsymbol{\theta}|| = 1$ and $y_i \langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle \geq \gamma$ for all $(\boldsymbol{x}_i, y_i)$.

The so-called *kernel trick* has recently gained popularity in machine learning (Schölkopf and Smola, 2002). As long as all operations of an algorithm can be expressed with inner products, the kernel trick can used to *lift* the algorithm to a higher-dimensional *feature space*: The inner product in the feature space produced by the mapping $\phi : \mathcal{X} \to \mathcal{H}$ is represented by a *kernel* $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$. We can now drop the condition $\mathcal{X} \subseteq \mathbb{R}^d$ but instead require that $\mathcal{H}$ be a *reproducing kernel Hilbert space* (RKHS).

To kernelize the perceptron algorithm, we first use ϕ to map the data into feature space, and observe that the weight vector can be expressed as $\boldsymbol{\theta} = \sum_{j \in \mathcal{J}} y_j \phi(\boldsymbol{x}_j)$, where $\mathcal{J}$ is the set of indices where mistakes occurred. We can now compute $\langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle = \sum_{j \in \mathcal{J}} y_j \left\langle \phi(\boldsymbol{x}_j), \phi(\boldsymbol{x}_i) \right\rangle = \sum_{j \in \mathcal{J}} y_j k(\boldsymbol{x}_j, \boldsymbol{x}_i)$, replacing explicit computation of ϕ with kernel evaluations.

The main drawback of the perceptron algorithm is that it does not maximize the margin of separation between the members of different classes. Frieß et al. (1998) address this issue with their closely related *kernel adatron* (KA). The KA algorithm uses a weight vector $\boldsymbol{\theta} = \sum_i \alpha_i y_i \phi(\boldsymbol{x}_i)$. Initially all α_i are set to 1. For a new instance $(\boldsymbol{x}, y)$ we compute $z = 1 - y \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})$, and update the corresponding α as $\alpha := \alpha + \eta z$ if $\alpha + \eta z > 0$; otherwise we set $\alpha = 0$.[3] Frieß et al. (1998) show that if the data is separable, this algorithm converges to the maximum margin solution in a finite number of iterations, and that the error rate decreases exponentially with the number of iterations.

To address the case where the data is not separable in feature space, Freund and Schapire (1999) work with a kernelized perceptron but use the online-to-batch conversion procedure of Helmbold and Warmuth (1995) to derive their *voted perceptron* algorithm. Essentially, every weight vector generated by the kernel perceptron is retained, and the decision rule is a majority vote amongst the predictions generated by these weight vectors. They prove the following mistake bound:

Theorem 2 *(Freund and Schapire, 1999, Corollary 1) Let* $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots (\boldsymbol{x}_m, y_m)\}$ *be a sequence of training samples and* $(\boldsymbol{x}_{m+1}, y_{m+1})$ *a test sample, all taken i.i.d. at random. Let* $R = \max_{1 \leq i \leq m+1} ||\boldsymbol{x}_i||$. *For* $||\boldsymbol{\theta}|| = 1$ *and* $\gamma > 0$, *let*

$$D_{\boldsymbol{\theta},\gamma} = \sqrt{\sum_{i=1}^{m+1} (\max(0, \gamma - y_i \langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle))^2}. \tag{10}$$

Then the probability (under resampling) that the voted perceptron algorithm does not predict y_{m+1} *on test sample* $\boldsymbol{x}_{m+1}$ *after one pass through the sequence of training samples is at most*

$$\frac{2}{m+1} \mathbb{E} \left[\inf_{||\boldsymbol{\theta}||=1; \gamma>0} \left(\frac{R + D_{\boldsymbol{\theta},\gamma}}{\gamma} \right)^2 \right], \tag{11}$$

where $\mathbb{E}$ *denotes the expectation under resampling.*

Another online algorithm which aims to maximize the margin of separation between classes is LASVM (Bordes et al., 2005). This follows a line of *budget* (kernel Perceptron) algorithms which sport a removal step (Crammer et al., 2004; Weston et al., 2005). Briefly, LASVM tries to solve the

3. In the interest of a clear exposition we ignore the *offset b* here.

SVM quadratic programming (QP) problem in an online manner. If the new instance violates the KKT conditions then it is added to the so-called *active set* during the PROCESS step. A REPROCESS step is run to identify points in the active set whose coefficients are constrained at either their upper or lower bound; such points are then discarded from the active set. Bordes et al. (2005) have shown that in the limit LASVM solves the SVM QP problem, although no rates of convergence or mistake bounds have been proven.

The *ballseptron* is another variant of the perceptron algorithm which takes the margin of separation between classes into account (Shalev-Shwartz and Singer, 2005). In contrast to the classic perceptron, the ballseptron updates its weight vector even for well-classified instances if they are close to the decision boundary. More precisely, if a ball $B(\boldsymbol{x},r)$ of radius r around the instance $\boldsymbol{x}$ intersects the decision boundary, the worst-violating point in B is used as a pseudo-instance for updating the weight vector. Shalev-Shwartz and Singer (2005) show that appropriate choice of r yields essentially the same bound as Theorem 1 above; this bound can be tightened further when the number of margin errors is strictly positive.

Another notable effort to derive a margin-based online learning algorithm is ALMA$_p$, the *approximate large margin algorithm* w.r.t. norm p (Gentile, 2001). Following Gentile and Littlestone (1999), the notion of a margin is extended to p-norms: Let $\boldsymbol{x}' = \boldsymbol{x}/||\boldsymbol{x}||_p$, and $||\boldsymbol{\theta}||_q \leq 1$, where $\frac{1}{p}+\frac{1}{q}=1$. Then the p-margin of $(\boldsymbol{x},y)$ w.r.t. $\boldsymbol{\theta}$ is defined as $y_i \langle \boldsymbol{\theta}, \boldsymbol{x}' \rangle$. Like other perceptron-inspired algorithms, ALMA$_p$ does not perform an update if the current weight vector classifies the current instance with a large p-margin. If a margin violation occurs, however, the algorithm performs a p-norm perceptron update, then projects the obtained $\boldsymbol{\theta}$ to the q-norm unit ball to maintain the constraint $||\boldsymbol{\theta}||_q \leq 1$. ALMA$_p$ is one of the few percpetron-derived online algorithms we know of which modify their learning rate: Its p-norm perceptron update step scales with the number of corrections which have occurred so far. ALMA$_p$ can be kernelized only for $p=2$.

Many large-margin algorithms (Li and Long, 2002; Crammer and Singer, 2003; Herbster, 2001) are based on the same general principle: They explicitly maximize the margin and update their weights only when a margin violation occurs. These violating instances are inserted into the kernel expansion with a suitable coefficient. To avoid potential over-fitting and reduce computational complexity, these algorithms either implement a removal step or work with a fixed-size buffer. The online SVM (Kivinen et al., 2004) is one such algorithm.

4. Online SVM

We now present the online SVM (*aka* NORMA) algorithm (Kivinen et al., 2004) from a loss function and regularization point of view, with additions and modifications for logistic regression, novelty detection, multiclass classification, and graph-structured label domains. This sets the scene for our application of SMD to the online SVM in Section 5. While many of the loss functions discussed below have been proposed before, we present them here in a common, unifying framework that cleanly separates the roles of loss function and optimization algorithm.

4.1 Optimization Problem

Let $\mathcal{X}$ be the space of observations, and $\mathcal{Y}$ the space of labels. We use $|\mathcal{Y}|$ to denote the size of $\mathcal{Y}$. Given a sequence $\{(\boldsymbol{x}_i,y_i)|\boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ of examples and a loss function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}$, our

goal is to minimize the regularized risk

$$J(f) = \frac{1}{m}\sum_{i=1}^{m} l(\boldsymbol{x}_i, y_i, f) + \frac{c}{2}\|f\|^2_{\mathcal{H}}, \tag{12}$$

where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) of functions on $\mathcal{X} \times \mathcal{Y}$. Its defining kernel is denoted by $k : (\mathcal{X} \times \mathcal{Y})^2 \to \mathbb{R}$, which satisfies $\langle f, k((\boldsymbol{x},y),\cdot)\rangle_{\mathcal{H}} = f(\boldsymbol{x},y)$ for all $f \in \mathcal{H}$. In a departure from tradition, but keeping in line with Altun et al. (2004); Tsochantaridis et al. (2004); Cai and Hofmann (2004), we let our kernel depend on the labels as well as the observations. Finally, we make the assumption that l only depends on f via its evaluations at $f(\boldsymbol{x}_i, y_i)$ and that l is piecewise differentiable.

By the reproducing property of $\mathcal{H}$ we can compute derivatives of the evaluation functional. That is,

$$\partial_f f(\boldsymbol{x},y) = \partial_f \langle f, k((\boldsymbol{x},y),\cdot)\rangle_{\mathcal{H}} = k((\boldsymbol{x},y),\cdot). \tag{13}$$

Since l depends on f only via its evaluations we can see that $\partial_f l(\boldsymbol{x}, y, f) \in \mathcal{H}$, and more specifically

$$\partial_f l(\boldsymbol{x}, y, f) \in \text{span}\{k((\boldsymbol{x},\tilde{y}),\cdot) \text{ where } \tilde{y} \in \mathcal{Y}\}. \tag{14}$$

Let $(\boldsymbol{x}_t, y_t)$ denote the example presented to the online algorithm at time instance t. Using the stochastic approximation of $J(f)$ at time t:

$$J_t(f) := l(\boldsymbol{x}_t, y_t, f) + \frac{c}{2}\|f\|^2_{\mathcal{H}} \tag{15}$$

and setting

$$g_t := \partial_f J_t(f_t) = \partial_f l(\boldsymbol{x}_t, y_t, f_t) + c f_t, \tag{16}$$

we obtain the following online learning algorithm:

Algorithm 1 Online learning (adaptive step size)

1. Initialize $f_0 = 0$
2. **Repeat**
   (a) Draw data sample $(\boldsymbol{x}_t, y_t)$
   (b) Adapt step size η_t
   (c) Update $f_{t+1} \leftarrow f_t - \eta_t g_t$

Practical considerations are how to implement steps 2(b) and 2(c) efficiently. We will discuss 2(c) below. Step 2(b), which primarily distinguishes the present paper from the previous work of Kivinen et al. (2004), is discussed in Section 5.

Observe that, so far, our discussion of the online update algorithm is independent of the particular loss function used. In other words, to apply our method to a new setting we simply need to compute the corresponding loss function and its gradient. We discuss particular examples of loss functions and their gradients in the next section.

4.2 Loss Functions

A multitude of loss functions are commonly used to derive seemingly different kernel methods. This often blurs the similarities as well as subtle differences between these methods. In this section, we discuss some commonly used loss functions and put them in perspective. We begin with loss functions on unstructured output domains, then proceed to to cases where the label space $\mathcal{Y}$ is structured. Since our online update depends on it, we will state the gradient of all loss functions we present below, and give its kernel expansion coefficients. For piecewise linear loss functions, we employ one-sided derivatives at the points where they are not differentiable — *cf.* Footnote 1.

4.2.1 LOSS FUNCTIONS ON UNSTRUCTURED OUTPUT DOMAINS

Binary Classification uses the hinge or soft margin loss (Bennett and Mangasarian, 1992; Cortes and Vapnik, 1995)

$$l(\boldsymbol{x}, y, f) = \max(0, 1 - y f(\boldsymbol{x})) \tag{17}$$

where $\mathcal{H}$ is defined on $\mathcal{X}$ alone. We have

$$\partial_f l(\boldsymbol{x}, y, f) = \begin{cases} 0 \text{ if } y f(\boldsymbol{x}) \geq 1 \\ -y k(\boldsymbol{x}, \cdot) \text{ otherwise} \end{cases} \tag{18}$$

Multiclass Classification employs a definition of the margin arising from log-likelihood ratios (Crammer and Singer, 2000). This leads to

$$l(\boldsymbol{x}, y, f) = \max(0, 1 + \max_{\tilde{y} \neq y} f(\boldsymbol{x}, \tilde{y}) - f(\boldsymbol{x}, y)) \tag{19}$$

$$\partial_f l(\boldsymbol{x}, y, f) = \begin{cases} 0 \text{ if } f(\boldsymbol{x}, y) \geq 1 + f(\boldsymbol{x}, y^*) \\ k((\boldsymbol{x}, y^*), \cdot) - k((\boldsymbol{x}, y), \cdot) \text{ otherwise} \end{cases} \tag{20}$$

Here we defined y^* to be the maximizer of the $\max_{\tilde{y} \neq y}$ operation. If several y^* exist we pick one of them arbitrarily, *e.g.* by dictionary order.

Logistic Regression works by minimizing the negative log-likelihood. This loss function is used in Gaussian process classification (MacKay, 1998). For binary classification this yields

$$l(\boldsymbol{x}, y, f) = \log(1 + \exp(-y f(\boldsymbol{x}))) \tag{21}$$

$$\partial_f l(\boldsymbol{x}, y, f) = -y k(\boldsymbol{x}, \cdot) \frac{1}{1 + \exp(y f(\boldsymbol{x}))} \tag{22}$$

Again the RKHS $\mathcal{H}$ is defined on $\mathcal{X}$ only.

Multiclass Logistic Regression works similarly to the example above. The only difference is that the log-likelihood arises from a conditionally multinomial model (MacKay, 1998). This means that

$$l(\boldsymbol{x}, y, f) = -f(\boldsymbol{x}, y) + \log \sum_{\tilde{y} \in \mathcal{Y}} \exp f(\boldsymbol{x}, \tilde{y}) \tag{23}$$

$$\partial_f l(\boldsymbol{x}, y, f) = \sum_{\tilde{y} \in \mathcal{Y}} k((\boldsymbol{x}, \tilde{y}), \cdot)[p(\tilde{y}|\boldsymbol{x}, f) - \delta_{y,\tilde{y}}], \tag{24}$$

$$\text{where we used} \quad p(y|\boldsymbol{x}, f) = \frac{e^{f(\boldsymbol{x}, y)}}{\sum_{\tilde{y} \in \mathcal{Y}} e^{f(\boldsymbol{x}, \tilde{y})}}. \tag{25}$$

Novelty Detection uses a trimmed version of the log-likelihood as a loss function. In practice this means that labels are ignored and the one-class margin needs to exceed 1 (Schölkopf et al., 2001). This leads to

$$l(\boldsymbol{x}, y, f) = \max(0, 1 - f(\boldsymbol{x})) \tag{26}$$

$$\partial_f l(\boldsymbol{x}, y, f) = \begin{cases} 0 \text{ if } f(\boldsymbol{x}) \geq 1 \\ -k(\boldsymbol{x}, \cdot) \text{ otherwise} \end{cases} \tag{27}$$

4.2.2 LOSS FUNCTIONS ON STRUCTURED LABEL DOMAINS

In many applications the output domain has an inherent structure. For example, document categorization deals with the problem of assigning a set of documents to a set of pre-defined topic hierarchies or taxonomies. Consider a typical taxonomy shown in Figure 2 which is based on a subset of the open directory project (`http://www.dmoz.org/`). If a document describing CDROMs is classified under hard disk drives ('HDD'), intuitively the loss should be smaller than when the same document is classified under 'Cables'. Roughly speaking, the value of the loss function should depend on the length of the shortest path connecting the actual label to the predicted label *i.e.*, the loss function should respect the structure of the output space (Tsochantaridis et al., 2004).

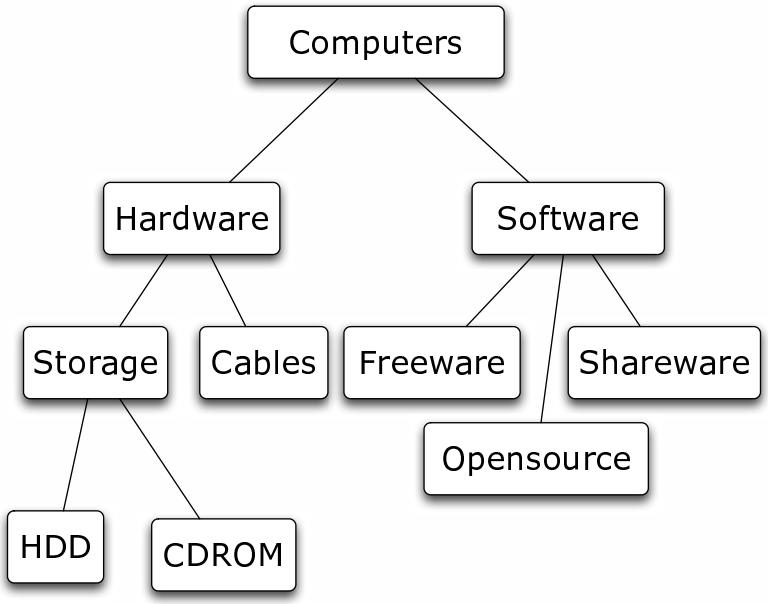


Figure 2: A taxonomy based on the open directory project.

To formalize our intuition, we need to introduce some notation. A weighted graph $G = (V, E)$ is defined by a set of nodes V and edges $E \subseteq V \times V$, such that, each edge $(v_i, v_j) \in E$ is assigned a non-negative weight $w(v_i, v_j) \in \mathbb{R}^+$. A path from $v_1 \in V$ to $v_n \in V$ is a sequence of nodes $v_1 v_2 \ldots v_n$ such that $(v_i, v_{i+1}) \in E$. The weight of a path is the sum of the weights on the edges. For an undirected graph, $(v_i, v_j) \in E \implies (v_j, v_i) \in E \wedge w(v_i, v_j) = w(v_j, v_i)$.

A graph is said to be connected if every pair of nodes in the graph are connected by a path. In the sequel we will deal exclusively with connected graphs, and let $\Delta_G(v_i, v_j)$ denote the weight of the shortest (*i.e.*, minimum weight) path from v_i to v_j. If the output labels are nodes in a graph G, the following loss function takes the structure of G into account:

$$l(\boldsymbol{x}, y, f) = \max\{0, \max_{\tilde{y} \neq y} [\Delta_G(\tilde{y}, y) + f(\boldsymbol{x}, \tilde{y})] - f(\boldsymbol{x}, y)\}. \tag{28}$$

This loss requires that the output labels $\tilde{y}$ which are "far away" from the actual label y (on the graph) must be classified with a larger margin while nearby labels are allowed to be classified with a smaller margin. More general notions of distance, including kernels on the nodes of the graph, can also be used here instead of the shortest path $\Delta_G(\tilde{y}, y)$.

Analogous to (24), by defining y^* to be the maximizer of the $\max_{\tilde{y} \neq y}$ operation we can write the gradient of the loss as:

$$\partial_f l(\boldsymbol{x}, y, f) = \begin{cases} 0 \text{ if } f(\boldsymbol{x}, y) \geq \Delta(y, y^*) + f(\boldsymbol{x}, y^*) \\ k((\boldsymbol{x}, y^*), \cdot) - k((\boldsymbol{x}, y), \cdot) \text{ otherwise} \end{cases} \tag{29}$$

The multiclass loss (19) is a special case of graph-based loss (28): consider a simple two-level tree in which each label is a child of the root node, and every edge has a weight of $\frac{1}{2}$. In this graph, any two labels $y \neq \tilde{y}$ will have $\Delta(y, \tilde{y}) = 1$, and thus (28) reduces to (19). We will employ a similar but multi-level tree-structured loss in our experiments on hierarchical document categorization (Section 6.4).

4.2.3 LOSS FUNCTION SUMMARY AND EXPANSION COEFFICIENTS

Note that the gradient always has the form

$$\partial_f l(\boldsymbol{x}_t, y_t, f_t) =: \langle \boldsymbol{\xi}_t, k((\boldsymbol{x}_t, \cdot), \cdot) \rangle \tag{30}$$

where $\boldsymbol{\xi}$ denotes the *expansion coefficient(s)* — more than one in the multiclass and structured label domain cases — arising from the derivative of the loss at time t.

Table 1 summarizes the tasks, loss functions, and expansion coefficients we have considered above. Similar derivations can be found for ε-insensitive regression, Huber's robust regression, or LMS problems.

4.3 Coefficient Updates

Since the online update in step 2(c) of Algorithm 1 is not particularly useful in Hilbert space, we now rephrase it in terms of kernel function expansions. This extends and complements the reasoning of Kivinen et al. (2004) as applied to the various loss functions of the previous section. From (15) it follows that $g_t = \partial_f l(\boldsymbol{x}_t, y_t, f_t) + c f_t$ and consequently

$$\begin{aligned} f_{t+1} &= f_t - \eta_t [\partial_f l(\boldsymbol{x}_t, y_t, f_t) + c f_t] \\ &= (1 - \eta_t c) f_t - \eta_t \partial_f l(\boldsymbol{x}_t, y_t, f_t). \end{aligned} \tag{31}$$

Using the initialization $f_1 = 0$ this implies that

$$f_{t+1}(\cdot) = \sum_{i=1}^{t} \sum_{y} \alpha_{iy} k((\boldsymbol{x}_i, y), \cdot). \tag{32}$$

task	loss function $l(\boldsymbol{x}_t, y_t, f_t)$	expansion coefficient(s) $\boldsymbol{\xi}_t$
Novelty Detection	$\max(0, 1 - f_t(\boldsymbol{x}_t))$	$\xi_t = \begin{cases} 0 & \text{if } f_t(\boldsymbol{x}_t) \geq 1 \\ -1 & \text{otherwise} \end{cases}$
Binary Classification	$\max(0, 1 - y_t f_t(\boldsymbol{x}_t))$	$\xi_t = \begin{cases} 0 & \text{if } y_t f_t(\boldsymbol{x}_t) \geq 1 \\ -y_t & \text{otherwise} \end{cases}$
Multiclass Classification	$\max[0, 1 - f_t(\boldsymbol{x}_t, y_t) + \max_{\tilde{y} \neq y_t} f_t(\boldsymbol{x}_t, \tilde{y})]$	$\boldsymbol{\xi}_t = \mathbf{0}$ if $f_t(\boldsymbol{x}_t, y_t) \geq 1 + f_t(\boldsymbol{x}_t, y^*)$ $\xi_{t,y_t} = -1, \xi_{t,y^*} = 1$ otherwise
Graph-Struct. Label Domains	$\max\{0, -f(\boldsymbol{x}_t, y_t) + \max_{\tilde{y} \neq y_t}[\Delta(y_t, \tilde{y}) + f(\boldsymbol{x}_t, \tilde{y})]\}$	$\boldsymbol{\xi}_t = \mathbf{0}$ if $f_t(\boldsymbol{x}_t, y_t) \geq \Delta(y_t, y^*) + f_t(\boldsymbol{x}_t, y^*)$ $\xi_{t,y_t} = -1, \xi_{t,y^*} = 1$ otherwise
Binary Logistic Regression	$\log\left(1 + e^{-y_t f_t(\boldsymbol{x}_t)}\right)$	$\xi_t = \frac{-y_t}{1 + e^{y_t f_t(\boldsymbol{x}_t)}}$
Multiclass Logistic Regression	$\log \sum_{\tilde{y} \in \mathcal{Y}} e^{f_t(\boldsymbol{x}_t, \tilde{y})} - f_t(\boldsymbol{x}_t, y_t)$	$\xi_{t,y} = p(y \vert \boldsymbol{x}_t, f_t) - \delta_{y,y_t}$

Table 1: Loss functions and gradient expansion coefficients.

With some abuse of notation we will use the same expression for the cases where $\mathcal{H}$ is defined on $\mathcal{X}$ rather than $\mathcal{X} \times \mathcal{Y}$. In this setting we replace (32) by the sum over i only (with corresponding coefficients α_{ti}). Whenever necessary we will use $\boldsymbol{\alpha}_t$ to refer to the entire coefficient vector (or matrix) and α_{ti} (or α_{tiy}) will refer to the specific coefficients. Observe that we can write

$$g_t(\cdot) = \sum_{i=1}^{t} \sum_{y} \gamma_{tiy} k((\boldsymbol{x}_i, y), \cdot), \tag{33}$$

$$\text{where} \quad \boldsymbol{\gamma}_t := \begin{bmatrix} c\,\boldsymbol{\alpha}_{t-1} \\ \boldsymbol{\xi}_t^\top \end{bmatrix}. \tag{34}$$

We can now rewrite the update equation (31) using only the expansion coefficients as

$$\boldsymbol{\alpha}_t = \begin{bmatrix} (1 - \eta_t c)\boldsymbol{\alpha}_{t-1} \\ -\eta_t \boldsymbol{\xi}_t^\top \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}_{t-1} \\ 0 \end{bmatrix} - \eta_t \boldsymbol{\gamma}_t. \tag{35}$$

Note that conceptually $\boldsymbol{\alpha}$ grows indefinitely as it acquires an additional row with each new data sample. Practical implementations will of course retain only a buffer of past examples with nonzero coefficients (see Section 5.5).

5. Online SVMD

We now show how the SMD framework described in Section 2 can be used to adapt the step size for online SVMs. The updates given in Section 4 remain as before, the only difference being that the step size η_t is adapted before its value is used to update $\boldsymbol{\alpha}$.

5.1 Scalar Representation

Since we are dealing with an optimization in a RKHS only scalar variants are possible.[4] The scalar equivalent of (4) is

$$\eta_{t+1} = \eta_t \max(\tfrac{1}{2}, 1 - \mu \langle g_{t+1}, v_{t+1} \rangle), \tag{36}$$

where μ is the meta-step size described in Section 2. The update for v is now given by

$$v_{t+1} = \lambda v_t - \eta_t (g_t + \lambda \boldsymbol{H}_t v_t), \tag{37}$$

where $\boldsymbol{H}_t$ is the Hessian of the objective function. Note that now $\boldsymbol{H}_t$ is an operator in Hilbert space. For $J_t(f)$ as defined in (15), this operator has a form that permits efficient computation of $\boldsymbol{H}_t v_t$:

For piecewise linear loss functions, such as (18), (20), and (27), we have $\boldsymbol{H}_t = c\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity operator, and obtain the simple update

$$v_{t+1} = (1 - \eta_t c)\lambda v_t - \eta_t g_t. \tag{38}$$

For other losses, note that l only depends on f via its evaluations at (x, y). This means that $\boldsymbol{H}_t$ differs from $c\boldsymbol{I}$ only by a low-rank object. In particular, for logistic regression (22) we have

$$\boldsymbol{H}_t - c\boldsymbol{I} = \rho(\boldsymbol{x}_t)\, k(\boldsymbol{x}_t, \cdot) \otimes k(\boldsymbol{x}_t, \cdot), \tag{39}$$

where $\rho(\boldsymbol{x}_t) := e^{y_t f_t(\boldsymbol{x}_t)} / (1 + e^{y_t f_t(\boldsymbol{x}_t)})^2$, and $\otimes$ denotes the outer product between functions in $\mathcal{H}$, obeying $(u \otimes v)w = u \langle v, w \rangle$ for $u, v, w \in \mathcal{H}$. Likewise, for multiclass logistic regression (24) we have

$$\boldsymbol{H}_t - c\boldsymbol{I} = \sum_{y, \tilde{y} \in \mathcal{Y}} \rho(\boldsymbol{x}_t, y, \tilde{y})\, k((\boldsymbol{x}_t, y), \cdot) \otimes k((\boldsymbol{x}_t, \tilde{y}), \cdot), \tag{40}$$

$$\text{where} \quad \rho(\boldsymbol{x}_t, y, \tilde{y}) := \delta_{y,\tilde{y}}\, p(\tilde{y} | \boldsymbol{x}_t, f_t) - p(\tilde{y} | \boldsymbol{x}_t, f_t)\, p(y | \boldsymbol{x}_t, f_t). \tag{41}$$

5.2 Expansion in Hilbert Space

The above discussion implies that v can be expressed as a linear combination of kernel functions, and consequently is also a member of the RKHS defined by $k(\cdot, \cdot)$. Thus v cannot be updated explicitly, as is done in the normal SMD algorithm (Section 2). Instead we write

$$v_{t+1}(\cdot) = \sum_{i=1}^{t} \sum_{y} \beta_{tiy} k((\boldsymbol{x}_i, y), \cdot) \tag{42}$$

and update the coefficients $\boldsymbol{\beta}$. This is sufficient for our purpose because we only need to be able to compute the inner products $\langle g, v \rangle_{\mathcal{H}}$ in order to update η. Below, we first discuss the case when $H = c\boldsymbol{I}$ and then extend the discussion to handle the off diagonal entries.

Diagonal Hessian. Analogous to the update on $\boldsymbol{\alpha}$ we can determine the updates on $\boldsymbol{\beta}$ via

$$\boldsymbol{\beta}_t = \begin{bmatrix} (1 - \eta_t c)\lambda \boldsymbol{\beta}_{t-1} \\ 0 \end{bmatrix} - \eta_t \gamma_t. \tag{43}$$

Although (43) suffices in principle to implement the overall algorithm, a naive implementation of the inner product $\langle g_t, v_t \rangle$ in (36) takes $O(t^2)$ time, rendering it impractical. We show in Section 5.3 how we can exploit the incremental nature of the updates to compute this inner product in linear time.

4. The situation is different for reduced-rank expansions which are parametrized by the reduced set vectors.

Non-diagonal Hessian. The only modification to (43) that we need to take into account is the contribution of the off-diagonal entries of $\boldsymbol{H}_t$ to $\boldsymbol{\beta}_t$. A close look at (39) and (40) reveals that the low-rank modification to $c\boldsymbol{I}$ only happens in the subspace spanned by $k((\boldsymbol{x}_t, y), \cdot)$ for $y \in \mathcal{Y}$. This means that we can express

$$(\boldsymbol{H}_t - c\boldsymbol{I})v_t = \sum_{\tilde{y} \in \mathcal{Y}} \chi_{ty} k((\boldsymbol{x}_t, \tilde{y}), \cdot), \tag{44}$$

allowing us to derive the analog of (43):

$$\boldsymbol{\beta}_t = \begin{bmatrix} (1 - \eta_t c)\lambda \boldsymbol{\beta}_{t-1} \\ 0 \end{bmatrix} - \eta_t(\gamma_t + \lambda \boldsymbol{\chi}_t). \tag{45}$$

In the case of binary logistic regression we have

$$(\boldsymbol{H}_t - c\boldsymbol{I})v_t = \rho(\boldsymbol{x}_t) v_t(\boldsymbol{x}_t) k(\boldsymbol{x}_t, \cdot), \tag{46}$$

and for multiclass logistic regression

$$(\boldsymbol{H}_t - c\boldsymbol{I})v_t = \sum_{y, \tilde{y} \in \mathcal{Y}} \rho(\boldsymbol{x}_t, y, \tilde{y}) v_t(\boldsymbol{x}_t, y) k((\boldsymbol{x}_t, \tilde{y}), \cdot). \tag{47}$$

5.3 Linear-Time Incremental Updates

We now turn to computing $\langle g_{t+1}, v_{t+1} \rangle$ in linear time by bringing it into an incremental form. For ease of exposition we will consider the case where $\boldsymbol{H}_t = c\boldsymbol{I}$; an extension to non-diagonal Hessians is straightforward but tedious. We use the notation $f(\boldsymbol{x}_t, \cdot)$ to denote the vector of $f(\boldsymbol{x}_t, \tilde{y})$ for $\tilde{y} \in \mathcal{Y}$. Expanding g_{t+1} into $c f_{t+1} + \boldsymbol{\xi}_{t+1}$ we can write

$$\langle g_{t+1}, v_{t+1} \rangle = c\pi_{t+1} + \boldsymbol{\xi}_{t+1}^\top v_{t+1}(\boldsymbol{x}_{t+1}, \cdot), \tag{48}$$

where $\pi_t := \langle f_t, v_t \rangle$. The function update (31) yields

$$\pi_{t+1} = (1 - \eta_t c) \langle f_t, v_{t+1} \rangle - \eta_t \boldsymbol{\xi}_t^\top v_{t+1}(\boldsymbol{x}_t, \cdot). \tag{49}$$

The v update (37) then gives us

$$\langle f_t, v_{t+1} \rangle = (1 - \eta_t c)\lambda \pi_t - \eta_t \langle f_t, g_t \rangle, \tag{50}$$

and using $g_t = c f_t + \boldsymbol{\xi}_t$ again we have

$$\langle f_t, g_t \rangle = c \| f_t \|^2 + \boldsymbol{\xi}_t^\top f_t(\boldsymbol{x}_t, \cdot). \tag{51}$$

Finally, the squared norm of f can be maintained via:

$$\| f_{t+1} \|^2 = (1 - \eta_t c)^2 \| f_t \|^2 - 2\eta_t (1 - \eta_t c) \boldsymbol{\xi}_t^\top f_t(\boldsymbol{x}_t, \cdot) + \eta_t^2 \boldsymbol{\xi}_t^\top k((\boldsymbol{x}_t, \cdot), (\boldsymbol{x}_t, \cdot)) \boldsymbol{\xi}_t. \tag{52}$$

The above sequence (48)–(52) of equations, including the evaluation of the associated functionals, can be performed in $O(t)$ time, and thus allows us to efficiently compute $\langle g_{t+1}, v_{t+1} \rangle$.

5.4 Extensions

We will now implement in the context of SVMD two important standard extensions to the SVM framework: offsets, and the specification of the fraction of points which violate the margin via the so-called ν-trick. Both of these extensions create new parameters, which we will also tune by stochastic gradient descent, again using SMD for step size adaptation.

5.4.1 HANDLING OFFSETS

In many situations, for instance in binary classification, it is advantageous to add an offset $\mathbf{b} \in \mathbb{R}^{|\mathcal{Y}|}$ to the prediction $f \in \mathcal{H}$. While the update equations described above remain unchanged, the offset parameter $\mathbf{b}$ is now adapted as well:

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \boldsymbol{\eta}_{\mathbf{b},t} \cdot \partial_{\mathbf{b}} J_t(f_t + \mathbf{b}_t) = \mathbf{b}_t - \boldsymbol{\eta}_{\mathbf{b},t} \cdot \boldsymbol{\xi}_t. \tag{53}$$

Applying the standard SMD equations (4) and (7) to the case at hand, we update the offset step sizes $\boldsymbol{\eta}_{\mathbf{b}}$ via

$$\boldsymbol{\eta}_{\mathbf{b},t+1} = \boldsymbol{\eta}_{\mathbf{b},t} \cdot \max(\tfrac{1}{2}, 1 - \mu_{\mathbf{b}} \boldsymbol{\xi}_{t+1} \cdot \boldsymbol{v}_{\mathbf{b},t+1}), \tag{54}$$

where $\mu_{\mathbf{b}}$ is the meta-step size for adjusting $\boldsymbol{\eta}_{\mathbf{b}}$, and $\boldsymbol{v}_{\mathbf{b}}$ is adapted as

$$\boldsymbol{v}_{\mathbf{b},t+1} = \lambda_{\mathbf{b}} \boldsymbol{v}_{\mathbf{b},t} - \eta_{\mathbf{b},t} \cdot \boldsymbol{\xi}_t. \tag{55}$$

Note that we can adjust the offset for each class in $\mathcal{Y}$ individually.

5.4.2 THE ν-TRICK

The so called ν-trick allows one to pre-specify the fraction, $0 < \nu < 1$, of points which violate the margin. For instance, when performing novelty detection using the ν-trick, the loss function that is commonly used is

$$l(\boldsymbol{x}, y, f) = \max(0, \varepsilon - f(\boldsymbol{x})) - \nu\varepsilon. \tag{56}$$

Here, the regularization parameter is fixed at $c = 1$, but the margin is adapted with the additional constraint $\varepsilon > 0$. This can easily be taken into account by adapting ε in log-space. Observe that

$$\partial_{\log\varepsilon} J_t(f_t) = \varepsilon \, \partial_{\varepsilon} J_t(f_t) = -\varepsilon(\xi_t + \nu), \tag{57}$$

and therefore the updates for ε can now be written as

$$\varepsilon_{t+1} = \varepsilon_t \exp(-\eta_{\varepsilon,t} \, \partial_{\log\varepsilon} J_t(f_t)) \tag{58}$$

$$= \varepsilon_t \exp(\eta_{\varepsilon,t} \varepsilon_t (\xi_t + \nu)). \tag{59}$$

We now use SMD to adapt the margin step size $\eta_{\varepsilon,t}$:

$$\eta_{\varepsilon,t+1} = \eta_{\varepsilon,t} \max(\tfrac{1}{2}, 1 + \mu_{\varepsilon} v_{\varepsilon,t} \varepsilon_t (\xi_t + \nu)), \tag{60}$$

where $v_{\varepsilon,t}$ is updated as

$$v_{\varepsilon,t+1} = \lambda_{\varepsilon} v_{\varepsilon,t} + \eta_{\varepsilon,t} \varepsilon_t (\xi_t + \nu)(1 + \lambda_{\varepsilon} v_{\varepsilon,t}). \tag{61}$$

This is a straightforward application of the SMD update equations (4) and (7), taking into account that ε is adapted in log-space.

This completes our description of the online SVMD algorithm. Since it comprises a rather large number of update equations, it is non-trivial to arrange them in an appropriate order. Algorithm 2 shows the ordering which we have found most advantageous.

Algorithm 2 Online [ν-]SVMD

1. Initialize
2. **Repeat**
   (a) data, prediction and loss:
      i. draw data sample $(\boldsymbol{x}_t, y_t)$
      ii. calculate prediction $f_t(\boldsymbol{x}_t)$
      iii. calculate loss $l(\boldsymbol{x}_t, y_t, f_t)$
   (b) obtain gradients:
      i. calculate $\xi_t = \partial_f l(\boldsymbol{x}_t, y_t, f_t)$
      ii. (42) calculate $v_t(\boldsymbol{x}_t)$
      iii. (48) calculate $\langle g_t, v_t \rangle$
      iv. (34) calculate *g resp.* γ_t
   (c) perform SMD:
      i. (36) update step size(s) $[\eta_\varepsilon, \eta_{\boldsymbol{b}},]\ \eta$
      ii. (46)/(47) if $\boldsymbol{H}$ non-diag.: compute $\boldsymbol{\chi}$
      iii. (43)/(45) update $[\varepsilon, v_\varepsilon, \boldsymbol{v}_{\boldsymbol{b}},]$ *v resp.* $\boldsymbol{\beta}$
   (d) maintain incrementals:
      i. (51) calculate $\langle f_t, g_t \rangle$
      ii. (50) calculate $\langle f_t, v_{t+1} \rangle$
      iii. (42) calculate $v_{t+1}(\boldsymbol{x}_t)$
      iv. (49) update π
      v. (52) update $\|f\|^2$
   (e) (35) update function f *resp.* $\boldsymbol{\alpha}$

5.5 Time Complexity and Buffer Management

The time complexity of online SVMD is dominated by the cost of the kernel expansions in steps 2(a)ii, 2(b)ii, and 2(d)iii of Algorithm 2, which grows linearly with the size of the expansion. Since unlimited growth would be undesirable, we maintain a *least recently used* (LRU) circular buffer which stores only the last ω non-zero expansion coefficients; each kernel expansion then takes $O(\omega|\mathcal{Y}|)$ time.

The online SVM (NORMA) algorithm does not require steps 2(b)ii or 2(d)iii, but still has to employ step 2(a)ii to make a prediction, so its asymptotic time complexity is $O(\omega|\mathcal{Y}|)$ as well. The two algorithms thus differ in time complexity only by a constant factor; in practice we observe online SVMD to be 3–4 times slower per iteration than NORMA.

Limiting the kernel expansion to the most recent ω non-zero terms makes sense because at each iteration t the coefficients α_i with $i < t$ are multiplied by a factor $(1 - \eta_t c) < 1$. After sufficiently many iterations α_i will thus have shrunk to a small value whose contribution to $f(\boldsymbol{x}_t)$ becomes negligible — and likewise for β_i's contribution to $v(\boldsymbol{x}_t)$. If the loss function has a bounded gradient (as in all cases of Table 1), then it can be shown that the truncation error thus introduced decreases

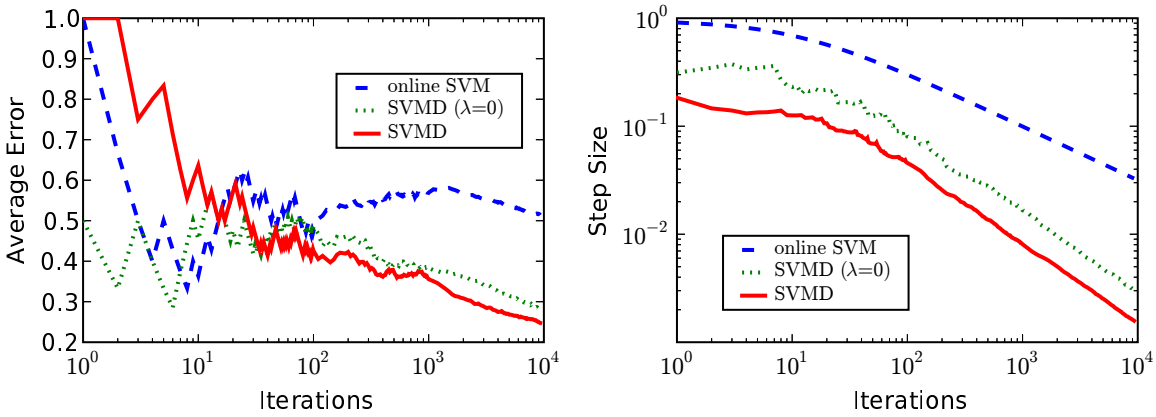


Figure 3: ν-SVM binary classification over a single run through the USPS data set. Current average error (left) and step size (right) for SVMD with $\lambda = 0.95$ (solid), $\lambda = 0$ (dotted), and online SVM with step size decay (62), using $\tau = 10$ (dashed).

exponentially with the number of terms retained (Kivinen et al., 2004, Proposition 1), so good solutions can be obtained with limited buffer size ω.

A good buffer management scheme has to deal with two conflicting demands: To the extent that the data set is non-stationary, it is desirable to remove old items from the buffer in order to reduce the effect of obsolete data. The truncation error, on the other hand, is reduced by using as large a buffer as possible. Although we employ a simple LRU circular buffer to good effect here, smarter buffer management strategies which explicitly remove the least important point based on some well-defined criterion (Crammer et al., 2004; Weston et al., 2005; Dekel et al., 2006) could also be adapted to work with our algorithm.

6. Experiments

We now evaluate the performance of SMD's step size adaptation in RKHS by comparing the online SVMD algorithm described in Section 5 above to step size adaptation based only on immediate, single-step effects —obtained by setting $\lambda = 0$ in SVMD— and to the conventional online SVM (*aka* NORMA) algorithm (Kivinen et al., 2004) with a scheduled step size decay of

$$\eta_t = \sqrt{\tau/(\tau+t)}\,, \tag{62}$$

where τ is hand-tuned to obtain good performance. We do not use offsets here; where we employ the ν-trick (*cf.* Section 5.4.2), we always set $\eta_{\varepsilon,0} = 1$, $\mu_\varepsilon = \mu$, and $\lambda_\varepsilon = \lambda$.

6.1 USPS Data Set

For our first set of experiments we use the well-known USPS data set (LeCun et al., 1989) with the RBF kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}\right), \tag{63}$$

setting $\sigma = 8$ via cross-validation. We extend this kernel to the multiclass case via the delta kernel:

$$k((\boldsymbol{x},y),(\boldsymbol{x}',y')) := k(\boldsymbol{x},\boldsymbol{x}')\,\delta_{yy'}. \tag{64}$$

In the spirit of online learning, we train for just a single run through the data, so that no digit is seen more than once by any algorithm. For a fair comparison, all algorithms started with the same initial step size η_0, and had the data presented in the same order. For implementation efficiency, we only store the last 512 support vectors in a circular buffer.

6.1.1 BINARY CLASSIFICATION

Figure 3 shows our results for binary classification. Here, the data was split into two classes comprising the digits 0–4 and 5–9, respectively. We use ν-SVM with $\nu = 0.05$, $\eta_0 = 1$, and $\mu = 1$ and plot current average error rate — that is, the total number of errors on the training samples seen so far divided by the iteration number — and step size. Observe that online SVMD (solid) is initially slower to learn, but after about 20 iterations it overtakes the online SVM (dashed), and overall makes only about half as many classification errors. The single-step version of SVMD with $\lambda = 0$ (dotted) has the fastest early convergence but is asymptotically inferior to SVMD proper, though still far better than the online SVM with scheduled step size decay.

6.1.2 MULTICLASS CLASSIFICATION

Figure 4 shows our results for 10-way multiclass classification using soft margin loss with $\eta_0 = 0.1$, $\mu = 0.1$, and $c = 1/(500n)$, where n is the number of training samples. Again online SVMD (solid) makes only about half as many classification errors overall as the online SVM (dashed), with the single-step ($\lambda = 0$) variant of SVMD (dotted) falling in between.

We found (by cross-validation) the online SVM with fixed decay schedule to perform best here for $\eta_0 = 0.1$. SVMD, on the other hand, is less dependent on a particular value of η_0 since it can adaptively adjust its step size. In this experiment, for instance, SVMD raised η significantly above its initial value of 0.1 — something that a predetermined decay schedule cannot do. We generally find the performance of online SVMD to be fairly independent of the initial step size.

6.2 Non-stationary USPS Counting Sequence

For our next set of experiments we rearrange the USPS data to create a highly non-stationary problem: we take 600 samples of each digit, then present them in the order corresponding to a 3-digit decimal counter running twice from 000 through 999 (Figure 5). This creates pronounced non-stationarities in the distribution of labels: '0' for instance is very frequent early in the sequence but rare towards the end.

6.2.1 MULTICLASS CLASSIFICATION

Here we perform 10-way multiclass classification on the USPS counting sequence, using ν-SVMD with soft margin loss, $\nu = 0.05$, $\eta_0 = 1$, and $\mu = 1$. As Figure 6 shows, ν-SVMD (solid) makes significantly fewer classification errors than the controls: The *average* error rate for ν-SVMD over its entire first (and only) pass through this sequence is less than 19% here. Online ν-SVM with scheduled step size decay, on the other hand, has serious difficulty with the non-stationary nature of our data and performs barely above change level (90% error rate); even the simple step size

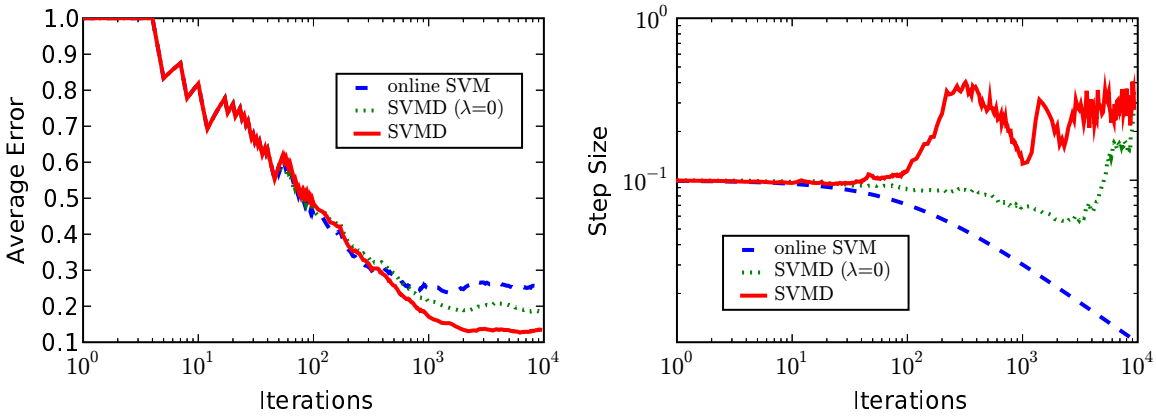


Figure 4: Online 10-way multiclass classification over a single run through the USPS data set. Current average error (left) and step size (right) for SVMD with $\lambda = 0.99$ (solid), $\lambda = 0$ (dotted), and online SVM with step size decay (62), using $\tau = 100$ (dashed).

adaptation obtained for $\lambda = 0$ clearly outperforms it. This is not surprising since a step size decay schedule typically assumes stationarity. By contrast, the decay factor of SVMD can be adjusted to match the time scale of non-stationarity; here $\lambda = 0.95$ yielded the best performance. In other experiments, we found (as one would expect) $\lambda = 1$ to work well for stationary data.

6.2.2 NOVELTY DETECTION

We also perform novelty detection with SVMD ($\mu = \lambda = 1$) on the USPS counting sequence. Figure 7 (left) shows that though SVMD markedly reduces the initial step size, it does not anneal it down to zero. Its ongoing reactivity is evidenced by the occurrence of spikes in η_t that correspond to identifiable events in our counting sequence. Specifically, major increases in η_t can be observed after seeing the first non-zero digit at $t = 6$, as the counter wraps from 19 to 20 at $t = 60$ (and likewise at $t = 120, 150, 180$), then at $t = 300, 1200, 1500$ as the hundreds wrap from 099 to 100, 399 to 400, and 499 to 500, respectively, and finally at $t = 3000$ as the entire sequence wraps around from 999 to 000. Many more minor peaks and troughs occur in between, closely tracking the fractal structure of our counting sequence.

Figure 5: To create a highly non-stationary problem, we rearranged 6000 USPS digits into a 3-digit counting sequence, running twice from 000 to 999.

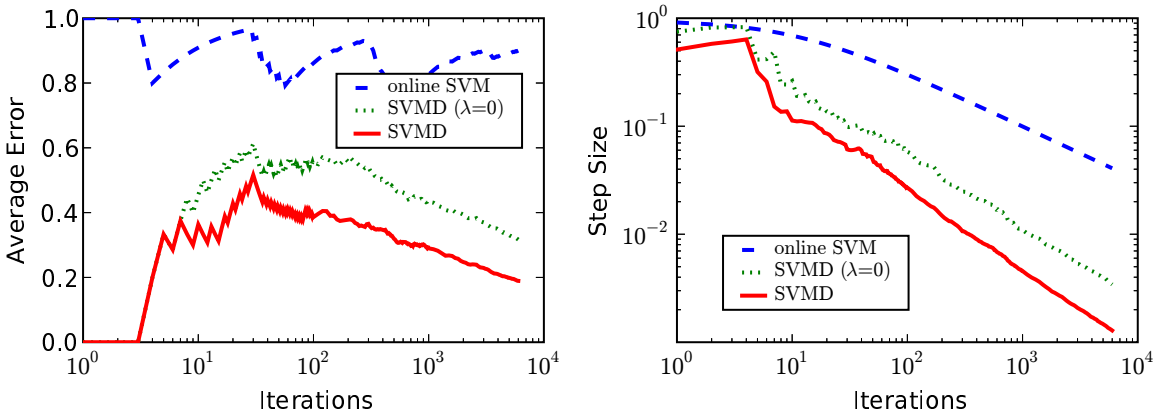


Figure 6: Online 10-way multiclass classification over a single run through our USPS counting sequence (Figure 5). Current average error (left) and step size (right) for ν-SVMD with $\lambda = 0.95$ (solid), $\lambda = 0$ (dotted), and online ν-SVM with step size decay (62), using $\tau = 100$ (dashed).

6.3 MNIST Data Set: Effect of Buffer Size

We now scale up our 10-way multiclass classification experiments to a much larger data set: 60000 digits from MNIST. We illustrate the effect of the circular buffer size on classification accuracy, using $\lambda = 1$, $\mu = 0.01$, and a polynomial kernel of degree 9. On a single run through the data, a buffer size of 256 yields around 20% average error (Figure 7, right). This reduces to 3.9% average error when the buffer size is increased to 4096; averaged over the last 4500 digits, the error rate is as low as 2.9%. For comparison, batch SVM algorithms achieve (at far greater computational cost) a generalization error rate of 1.4% on this data (Burges and Schölkopf, 1997; Schölkopf and Smola, 2002, p. 341).

6.4 WIPO-alpha Data Set: Tree-Structured Loss

For our experiments on document categorization we use the WIPO-alpha data set published in 2002 by the World Intellectual Property Organization (WIPO).[5] The data set consists of 75250 patents which have been classified into 4656 categories according to a standard taxonomy known as *international patent classification* (IPC, `http://www.wipo.int/classifications/en/`). Each document is assigned labels from a four-level hierarchy comprising sections, classes, sub-classes and groups. A typical label might be 'D05C 1/00' which would be read as Section D (Textiles; Paper), class 05 (Sewing; Embroidering; Tufting), sub-class C (Embroidering; Tufting) and group 1/00 (apparatus, devices, or tools for hand embroidering).

The IPC defines an undirected taxonomy tree. A tree is a graph with no cycles — *i.e.*, no paths whose start and end vertices coincide — and one node designated as the root. We use $\tilde{y} \preceq y$ to denote that $\tilde{y}$ is an ancestor of y, *i.e.*, the path from y to the root contains $\tilde{y}$.[6]

5. This data is now available on request from WIPO (`http://www.wipo.int/`).
6. Note that according to this definition, every node is an ancestor of itself; this is deliberate.

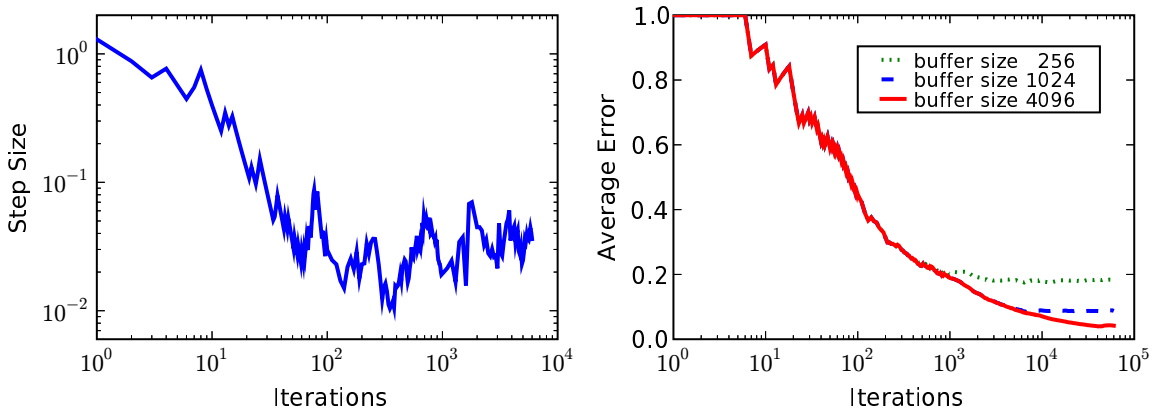


Figure 7: Left: The step size for novelty detection with SVMD on the USPS counting sequence (Figure 5) closely follows the non-stationarity of the data. Right: Average error of SVMD on the MNIST data set, for three different circular buffer sizes.

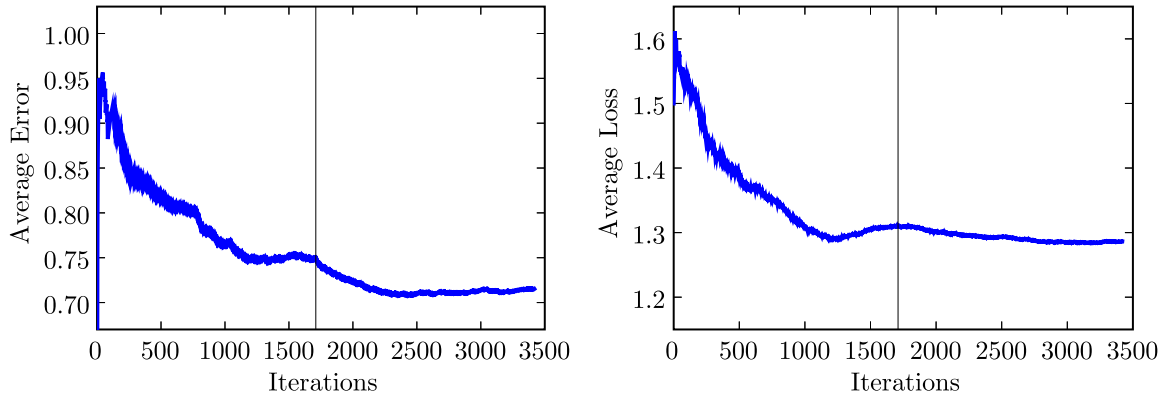


Figure 8: Average error (left) and loss (right) for SVMD over two passes (separated by vertical line) through section D of the WIPO-alpha data set.

Following Cai and Hofmann (2004), we perform document categorization experiments on the WIPO-alpha data set, using a loss function that is a special case of our graph-structured loss (28). Here the graph G is the taxonomy tree with a weight of $\frac{1}{2}$ on every edge, and the weighted distance between nodes is defined as (Cai and Hofmann, 2004):

$$\Delta_G(y,\tilde{y}) := \left(\sum_{\substack{z: z \prec y \\ \wedge z \not\preceq \tilde{y}}} \frac{1}{2} \right) + \left(\sum_{\substack{z: z \prec \tilde{y} \\ \wedge z \not\preceq y}} \frac{1}{2} \right). \tag{65}$$

A patent could have potentially many categories, but it has exactly one primary category. Following Cai and Hofmann (2004) we concentrate on predicting the primary category using the title

and claim contents. Furthermore, for the sake of reporting results, we restrict ourselves to Section D from the data set. This section contains 1710 documents categorized into 160 categories. Preprocessing of the data consisted of document parsing, tokenization, and term normalization in order to produce a bag-of-words representation. The bag-of-words vector is normalized to unit length. We use a product kernel which is defined as

$$k((\boldsymbol{x},y),(\boldsymbol{x}',y')) := k(\boldsymbol{x},\boldsymbol{x}')\,\kappa(y,y'). \tag{66}$$

For the document vectors we use a linear dot-product kernel

$$k(\boldsymbol{x},\boldsymbol{x}') := \boldsymbol{x}^\top \boldsymbol{x}', \tag{67}$$

while for the labels we use

$$\kappa(y,y') := \sum_{\substack{z:z\preceq y\\ \wedge z\preceq y'}} 1, \tag{68}$$

which counts the number of common ancestors of y and y'. Like Cai and Hofmann (2004), we set the regularizer c to the reciprocal of the number of data points. We use a buffer of size 1024, initial step size $\eta_0 = 1$, meta-step size $\mu = 0.1$, and decay parameter $\lambda = 0.999$. In Figure 8 we plot the current average error rate and graph-structured loss (28) over two passes through the data.

The WIPO-alpha data set is known to be difficult to learn (Cai and Hofmann, 2004; Tsochantaridis et al., 2004). Only 94 out of the 160 possible categories contain four or more examples while as many as 34 categories have exactly one sample, which makes it extremely hard for an online learning algorithm to predict well. Even the best offline algorithms have a reported error rate of 57.2% (Cai and Hofmann, 2004). SVMD performs competitively on this challenging data set, achieving an average error rate of around 75% over its first pass through the data, and 68% over its second pass. It reduces the average loss to around 1.32 over the first pass, and 1.28 over both passes (Figure 8). We found that further runs through the data set did not yield a significant increase in accuracy or reduction of the loss.

7. Discussion

We presented online SVMD, an extension of the SMD step size adaptation method to the kernel framework. Using an incremental approach to reduce a naive $O(t^2)$ computation to $O(t)$, we showed how the SMD parameters can be updated efficiently even though they now reside in an RKHS. We addressed the difficult cases of multiclass classification and logistic regression, where the Hessian operator in RKHS includes non-diagonal terms. We also showed how SVMD can be adapted to deal with structured output spaces. In experiments online SVMD outperformed the conventional online SVM (*aka* NORMA) algorithm with scheduled step size decay for binary and multiclass classification, especially on a non-stationary problem. In particular, it accelerated convergence to a good solution, which is one of the main aims of performing step size adaptation. In novelty detection experiments we observe that SVMD is able to closely track the non-stationarity in the data and adapt the step sizes correspondingly. With a reasonable buffer size SVMD attains competitive performance in a single pass through the MNIST data set. On a difficult document categorization task using the WIPO-alpha data set, SVMD performed well compared to the best offline algorithm.

Empirically we observe that in all our experiments the SVMD algorithm significantly speeds up the convergence of the conventional online SVM algorithm. It would be interesting to obtain worst case loss bounds for SVMD akin to those of Kivinen et al. (2004). The main technical challenge here is that the SMD update consists of three interleaved updates, and applying a straightforward analysis using Bregman divergences (Gentile and Littlestone, 1999; Azoury and Warmuth, 2001) is infeasible. Established methods for proving worst case loss bounds rely on the cancellation of telescoped terms, which works only when the step size η is held constant. In the case of SVMD, however, step sizes change from iteration to iteration. Even worse, their update is governed by two other feedback equations. A more sophisticated analysis, possibly involving second-order information, will have to be developed to establish similar loss bounds for SVMD.

SMD is a generic method to hasten the convergence of stochastic gradient descent methods. In combination with the kernel trick this provides a powerful learning tool. Other kernel algorithms which rely on stochastic gradient descent — *e.g.*, that of Kim et al. (2005) — could also be accelerated with SMD; this is a focus of our ongoing work in this area.

Acknowledgments

We would like to thank Alexandros Karatzoglou and Chang Chui for their help with early implementations, Lijuan Cai and Thomas Hofmann for making a pre-preprocessed version of the WIPO-Alpha data set available to us, and the anonymous reviewers for ICML, NIPS, and JMLR for their many helpful comments. National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program. This work was supported by the IST Program of the European Community, under the Pascal Network of Excellence, IST-2002-506778.

References

L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. Parameter adaptation in stochastic optimization. In David Saad, editor, *On-Line Learning in Neural Networks*, Publications of the Newton Institute, chapter 6, pages 111–134. Cambridge University Press, 1999.

Y. Altun, A. J. Smola, and T. Hofmann. Exponential families for conditional random fields. In *Uncertainty in Artificial Intelligence (UAI)*, pages 2–9, 2004.

K. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001. Special issue on Theoretical Advances in On-line Learning, Game Theory and Boosting.

A. G. Barto and R. S. Sutton. Goal seeking components for adaptive intelligence: An initial assessment. Technical Report AFWAL-TR-81-1070, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, Ohio 45433, USA, 1981.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

H. D. Block. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34: 123–135, 1962. Reprinted in *Neurocomputing* by Anderson and Rosenfeld.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, September 2005.

M. Bray, E. Koller-Meier, P. Müller, N. N. Schraudolph, and L. Van Gool. Stochastic optimization for high-dimensional tracking in dense range maps. *IEE Proceedings Vision, Image & Signal Processing*, 152(4):501–512, 2005.

M. Bray, E. Koller-Meier, N. N. Schraudolph, and L. Van Gool. Fast stochastic optimization for articulated structure tracking. *Image and Vision Computing*, 24, in press 2006.

C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 78–87, New York, NY, USA, 2004. ACM Press.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In N. Cesa-Bianchi and S. Goldman, editors, *Proc. Annual Conf. Computational Learning Theory*, pages 35–46, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, January 2003.

K. Crammer, J. Kandola, and Y. Singer. Online classification on a budget. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 225–232, Cambridge, MA, 2004. MIT Press.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a fixed budget. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

T.-T. Frieß, N. Cristianini, and C. Campbell. The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. In J. Shavlik, editor, *Proc. Intl. Conf. Machine Learning*, pages 188–196. Morgan Kaufmann Publishers, 1998.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, December 2001.

C. Gentile and N. Littlestone. The robustness of the p-norm algorithms. In *Proc. Annual Conf. Computational Learning Theory*, pages 1–11, Santa Cruz, California, United States, 1999. ACM Press, New York, NY.

A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

M. E. Harmon and L. C. Baird, III. Multi-player residual advantage learning with general function approximation. Technical Report WL-TR-1065, Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH 45433-7308, 1996. `http://www.leemon.com/papers/sim_tech/sim_tech.pdf`.

D. Helmbold and M. K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50(3):551–573, June 1995.

M. Herbster. Learning additive models online with fast evaluating kernels. In D. P. Helmbold and R. C. Williamson, editors, *Proc. Annual Conf. Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 444–460. Springer, 2001.

R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1: 295–307, 1988.

K. I. Kim, M. O. Franz, and B. Schölkopf. Iterative kernel principal component analysis for image modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1351–1366, 2005.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8), Aug 2004.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1–3): 361–387, 2002.

D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, pages 133–165. Springer, Berlin, 1998.

M. Milano. *Machine Learning Techniques for Flow Modeling and Control*. PhD thesis, Eidgenössische Technische Hochschule (ETH), Zürich, Switzerland, 2002.

M. Minsky and S. Papert. *Perceptrons: An Introduction To Computational Geometry*. MIT Press, Cambridge, MA, 1969.

A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

B. A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. International Conference on Neural Networks*, pages 586–591, San Francisco, CA, 1993. IEEE, New York.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *Proc. Intl. Conf. Artificial Neural Networks*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London.

N. N. Schraudolph and X. Giannakopoulos. Online independent component analysis with local learning rate adaptation. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Neural Information Processing Systems*, volume 12, pages 789–795, Vancouver, Canada, 2000. MIT Press.

N. N. Schraudolph, J. Yu, and D. Aberdeen. Fast online policy gradient learning with SMD gain vector adaptation. In Yair Weiss, Bernhard Schölkopf, and John Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

S. Shalev-Shwartz and Y. Singer. A new perspective on an old perceptron algorithm. In P. Auer and R. Meir, editors, *Proc. Annual Conf. Computational Learning Theory*, number 3559 in Lecture Notes in Artificial Intelligence, pages 264 – 279, Bertinoro, Italy, June 2005. Springer-Verlag.

F. M. Silva and L. B. Almeida. Acceleration techniques for the backpropagation algorithm. In Luís B. Almeida and C. J. Wellekens, editors, *Neural Networks: Proc. EURASIP Workshop*, volume 412 of *Lecture Notes in Computer Science*, pages 110–119. Springer Verlag, 1990.

R. S. Sutton. Adaptation of learning rate parameters, 1981. URL `http://www.cs.ualberta.ca/~sutton/papers/sutton-81.pdf`. Appendix C of (Barto and Sutton, 1981).

R. S. Sutton. Gain adaptation beats least squares? In *Proceedings of the 7th Yale Workshop on Adaptive and Learning Systems*, pages 161–166, 1992. URL `http://www.cs.ualberta.ca/~sutton/papers/sutton-92b.pdf`.

T. Tollenaere. SuperSAB: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3:561–573, 1990.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. Intl. Conf. Machine Learning*, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark Schmidt, and Kevin Murphy. Training conditional random fields with stochastic gradient methods. In *Proc. Intl. Conf. Machine Learning*, to appear 2006.

J. Weston, A. Bordes, and L. Bottou. Online (and offline) on an even tighter budget. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2005.

Journal of Machine Learning Research 7 (2006) 1135–1158 Submitted 2/05; Published 6/06

New Algorithms for Efficient High-Dimensional Nonparametric Classification

Ting Liu TINGLIU@CS.CMU.EDU
Andrew W. Moore AWM@CS.CMU.EDU
Alexander Gray AGRAY@CS.CMU.EDU
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Claire Cardie

Abstract

This paper is about non-approximate acceleration of high-dimensional nonparametric operations such as k nearest neighbor classifiers. We attempt to exploit the fact that even if we want exact answers to nonparametric queries, we usually do not need to explicitly find the data points close to the query, but merely need to answer questions about the properties of that set of data points. This offers a small amount of computational leeway, and we investigate how much that leeway can be exploited. This is applicable to many algorithms in nonparametric statistics, memory-based learning and kernel-based learning. But for clarity, this paper concentrates on pure k-NN classification. We introduce new ball-tree algorithms that on real-world data sets give accelerations from 2-fold to 100-fold compared against highly optimized traditional ball-tree-based k-NN . These results include data sets with up to 10^6 dimensions and 10^5 records, and demonstrate non-trivial speed-ups while giving exact answers.

keywords: ball-tree, k-NN classification

1. Introduction

Nonparametric models have become increasingly popular in the statistics and probabilistic AI communities. These models are extremely useful when the underlying distribution of the problem is unknown except that which can be inferred from samples. One simple well-known nonparametric classification method is called the k-nearest-neighbors or k-NN rule. Given a data set $V \subset R^D$ containing n points, it finds the k closest points to a query point $\mathrm{q} \in R^D$, typically under the Euclidean distance, and chooses the label corresponding to the majority. Despite the simplicity of this idea, it was famously shown by Cover and Hart (Cover and Hart, 1967) that asymptotically its error is within a factor of 2 of the optimal. Its simplicity allows it to be easily and flexibly applied to a variety of complex problems. It has applications in a wide range of real-world settings, in particular pattern recognition (Duda and Hart, 1973; Draper and Smith, 1981); text categorization (Uchimura and Tomita, 1997); database and data mining (Guttman, 1984; Hastie and Tibshirani, 1996); information retrieval (Deerwester et al., 1990; Faloutsos and Oard, 1995; Salton and McGill, 1983); image and multimedia search (Faloutsos et al., 1994; Pentland et al., 1994; Flickner et al., 1995; Smeulders and Jain, 1996); machine learning (Cost and Salzberg, 1993); statistics and data analysis (Devroye and Wagner, 1982; Koivune and Kassam, 1995) and also combination with other

methods (Woods et al., 1997). However, these methods all remain hampered by their computational complexity.

Several effective solutions exist for this problem when the dimension D is small, including Voronoi diagrams (Preparata and Shamos, 1985), which work well for two dimensional data. Other methods are designed to work for problems with moderate dimension (i.e. tens of dimensions), such as k-D tree (Friedman et al., 1977; Preparata and Shamos, 1985), R-tree (Guttman, 1984), and ball-tree (Fukunaga and Narendra, 1975; Omohundro, 1991; Uhlmann, 1991; Ciaccia et al., 1997). Among these tree structures, balltree, or metric-tree (Omohundro, 1991), represent the practical state of the art for achieving efficiency in the largest dimension possible (Moore, 2000; Clarkson, 2002) without resorting to approximate answers. They have been used in many different ways, in a variety of tree search algorithms and with a variety of "cached sufficient statistics" decorating the internal leaves, for example in Omohundro (1987); Deng and Moore (1995); Zhang et al. (1996); Pelleg and Moore (1999); Gray and Moore (2001). However, many real-world problems are posed with very large dimensions that are beyond the capability of such search structures to achieve sublinear efficiency, for example in computer vision, in which each pixel of an image represents a dimension. Thus, the high-dimensional case is the long-standing frontier of the nearest-neighbor problem.

With one exception, the proposals involving tree-based or other data structures have considered the generic nearest-neighbor problem, not that of nearest-neighbor *classification* specifically. Many proposals designed specifically for nearest-neighbor classification have been proposed, virtually all of them pursuing the idea of reducing the number of training points. In most of these approaches, such as Hart (1968), although the runtime is reduced, so is the classification accuracy. Several similar training set reduction schemes yielding only approximate classifications have been proposed (Fisher and Patrick, 1970; Gates, 1972; Chang, 1974; Ritter et al., 1975; Sethi, 1981; Palau and Snapp, 1998). Our method achieves the exact classification that would be achieved by exhaustive search for the nearest neighbors. A few training set reduction methods have the capability of yielding exact classifications. Djouadi and Bouktache (1997) described both approximate and exact methods, however a speedup of only about a factor of two over exhaustive search was reported for the exact case, for simulated, low-dimensional data. Lee and Chae (1998) also achieves exact classifications, but only obtained a speedup over exhaustive search of about 1.7. It is in fact common among the results reported for training set reduction methods that only 40-60% of the training points can be discarded, *i.e.* no important speedups are possible with this approach when the Bayes risk is not insignificant. Zhang and Srihari (2004) pursued a combination of training set reduction and a tree data structure, but is an approximate method.

In this paper, we propose two new ball-tree based algorithms, which we'll call KNS2 and KNS3. They are both designed for binary k-NN classification. We only focus on binary case, since there are many binary classification problems, such as anomaly detection (Kruegel and Vigna, 2003), drug activity detection (Komarek and Moore, 2003); and video segmentation (Qi et al., 2003). Liu et al. (2004b) applied similar ideas to many-class classification and proposed a variation of the k-NN algorithm. KNS2 and KNS3 share the same insight that the task of k-nearest-neighbor classification of a query q *need not require us to explicitly find those k nearest neighbors*. To be more specific, there are three similar but in fact different questions: (a) *"What are the k nearest neigh-*

bors of q*?"* (b) *"How many of the k nearest neighbors of* q *are from the positive class?"* and (c) *"Are at least t of the k nearest neighbors from the positive class?"* Many researches have focused on the first question (a), but uses of proximity queries in statistics far more frequently require (b) and (c) types of computations. In fact, for the *k*-NN classification problem, when the threshold *t* is set, it is sufficient to just answer the much simpler question (c). The triangle inequality underlying a ball-tree has the advantage of bounding the distances between data points, and can thus help us estimate the nearest neighbors without explicitly finding them. In our paper, we test our algorithms on 17 synthetic and real-world data sets, with dimensions ranging from 2 to 1.1×10^6 and number of data points ranging from 10^4 to 4.9×10^5. We observe up to 100-fold speedup compared against highly optimized traditional ball-tree-based *k*-NN , in which the neighbors are found explicitly.

Omachi and Aso (2000) proposed a fast *k*-NN classifier based on a branch and bound method, and the algorithm shares some ideas of KNS2, but it did not fully explore the idea of doing *k*-NN classification without explicitly finding the *k* nearest neighbor set, and the speed-up the algorithm achieved is limited. In section 4, we address Omachi and Aso's method in more detail.

We will first describe ball-trees and this traditional way of using them (which we call KNS1), which computes problem (a). Then we will describe a new method (KNS2) for problem (b), designed for the common setting of skewed-class data. We'll then describe a new method (KNS3) for problem (c), which removes the skewed-class assumption, applying to arbitrary classification problems. At the end of Section 5 we will say a bit about the relative value of KNS2 versus KNS3.

2. Ball-Tree

A *ball-tree* (Fukunaga and Narendra, 1975; Omohundro, 1991; Uhlmann, 1991; Ciaccia et al., 1997; Moore, 2000) is a binary tree where each node represents a set of points, called Points(Node). Given a data set, the *root node* of a ball-tree represents the full set of points in the data set. A node can be either a *leaf node* or a *non-leaf node*. A leaf node explicitly contains a list of the points represented by the node. A non-leaf node has two children nodes: *Node.child1* and *Node.child2*, where

$$Points(Node.child1) \cap Points(Node.child2) = \phi$$
$$Points(Node.child1) \cup Points(Node.child2) = Points(Node)$$

Points are organized spatially. Each node has a distinguished point called a *Pivot*. Depending on the implementation, the *Pivot* may be one of the data points, or it may be the centroid of *Points(Node)*. Each node records the maximum distance of the points it owns to its pivot. Call this the radius of the node

$$Node.Radius = \max_{\mathbf{x} \in Points(Node)} | Node.Pivot - \mathbf{x} |$$

Nodes lower down the tree have smaller radius. This is achieved by insisting, at tree construction time, that

$$\mathbf{x} \in Points(Node.child1) \quad \Rightarrow \quad | \mathbf{x} - Node.child1.Pivot | \ \leq \ | \mathbf{x} - Node.child2.Pivot |$$
$$\mathbf{x} \in Points(Node.child2) \quad \Rightarrow \quad | \mathbf{x} - Node.child2.Pivot | \ \leq \ | \mathbf{x} - Node.child1.Pivot |$$

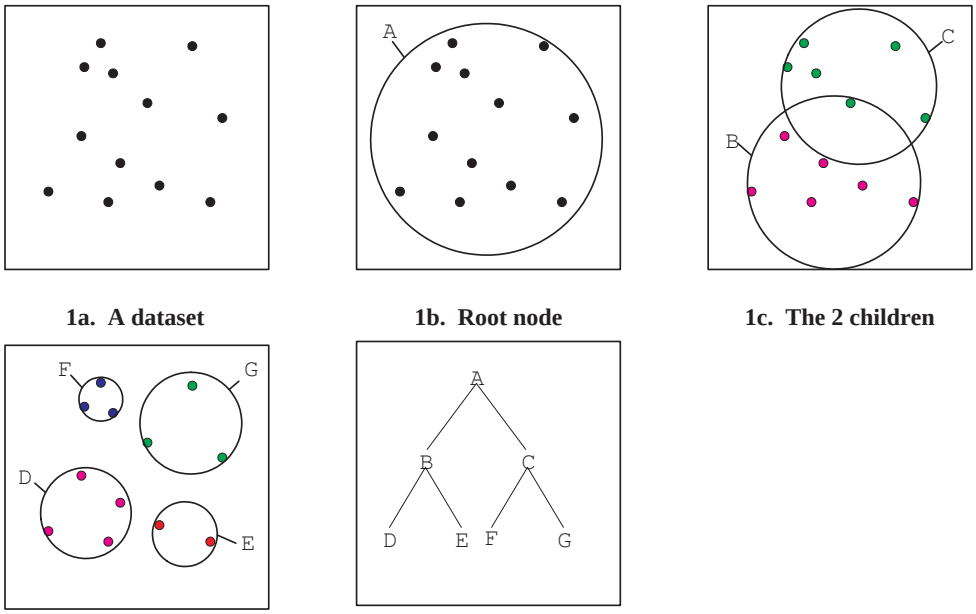
Provided that our distance function satisfies the triangle inequality, we can bound the distance from a query point $\mathbf{q}$ to any point in any ball-tree node. If $\mathbf{x} \in$ *Points(Node)* then we know that:

$$|\mathbf{x}-\mathbf{q}| \geq |\mathbf{q}-Node.Pivot| - Node.Radius \quad (1)$$

$$|\mathbf{x}-\mathbf{q}| \leq |\mathbf{q}-Node.Pivot| + Node.Radius \quad (2)$$

Here is an easy proof of the inequality. According to triangle inequality, we have $|\mathbf{x}-\mathbf{q}| \geq |\mathbf{q}-Node.Pivot| - |x-Node.Pivot|$. Given $\mathbf{x} \in$ *Points(Node)* and *Node.Radius* is the maximum distance of the points it owns to its pivot, $|\mathbf{x}-Node.Pivot| \leq Node.Radius$, so $|\mathbf{x}-\mathbf{q}| \geq |\mathbf{q}-Node.Pivot| - Node.Radius$. Similarly, we can prove Equation (2). ■

A ball-tree is constructed top-down. There are several ways to construct them, and practical algorithms trade off the cost of construction (it can be inefficient to be $O(R^2)$ given a data set with R points, for example) against the tightness of the radius of the balls. Moore (2000) describes a fast way for constructing a ball-tree appropriate for computational statistics. If a ball-tree is balanced, then the construction time is $O(CR\log R)$, where C is the cost of a point-point distance computation (which is $O(m)$ if there are m dense attributes, and $O(fm)$ if the records are sparse with only fraction f of attributes taking non-zero value). Figure 1 shows a 2-dimensional data set and the first few levels of a ball-tree.



1a. A dataset **1b. Root node** **1c. The 2 children**

1d. The 4 grandchildren **1e. The internal tree structure**

Figure 1: An example of ball-tree structure

3. KNS1: Conventional k Nearest Neighbor Search with Ball-Tree

In this paper, we call conventional ball-tree-based search (Uhlmann, 1991) *KNS1*. Let *PS* be a set of data points, and $PS \subseteq V$, where V is the training set. We begin with the following definition:

Say that *PS consists of the k-NN of* q *in V* if and only if

$$\begin{array}{lll} |V| \geq k & \textit{and} & PS \text{ are the } k\text{-NN of } \mathsf{q} \text{ in V} \\ & \textit{or} & \\ |V| < k & \textit{and} & PS == V \end{array} \tag{3}$$

We now define a recursive procedure called *BallKNN* with the following inputs and output.

$$PS^{out} = BallKNN(PS^{in}, Node)$$

Let V be the set of points searched so far, on entry. Assume that PS^{in} consists of the k-NN of q in V. This function efficiently ensures that on exit, PS^{out} consists of the k-NN of q in $V \cup Points(Node)$. We define

$$D_{\text{sofar}} = \begin{cases} \infty & if \ \ |PS^{in}| < k \\ \max_{\mathbf{x} \in PS^{in}} |\mathbf{x} - \mathsf{q}| & if \ \ |PS^{in}| == k \end{cases} \tag{4}$$

D_{sofar} is the minimum distance within which points become interesting to us.

$$\text{Let } D^{\text{Node}}_{\text{minp}} = \begin{cases} \max(|\mathsf{q} - Node.Pivot| - Node.Radius, D^{Node.parent}_{minp}) & if \ Node \neq Root \\ \max(|\mathsf{q} - Node.Pivot| - Node.Radius, 0) & if \ Node == Root \end{cases} \tag{5}$$

$D^{\text{Node}}_{\text{minp}}$ is the minimum possible distance from any point in *Node* to q. This is computed using the bound given by Equation (1) and the fact that all points covered by a node must be covered by its parent. This property implies that D^{Node}_{minp} will never be less than the minimum distance of its ancestors. Step 2 of section 4 explains this optimization further. See Figure 2 for details.
Experimental results show that KNS1 (conventional ball-tree-based k-NN search) achieves significant speedup over Naive k-NN when the dimension d of the data set is moderate (less than 30). In the best case, the complexity of KNS1 can be as good as $O(d \log R)$, given a data set with R points. However, with d increasing, the benefit achieved by KNS1 degrades, and when d is really large, in the worst case, the complexity of KNS1 can be as bad as $O(dR)$. Sometimes it is even slower than Naive k-NN search, due to the curse of dimensionality.

In the following sections, we describe our new algorithms KNS2 and KNS3, these two algorithms are both based on ball-tree structure, but by using different search strategies, we explore how much speedup can be achieved beyond KNS1.

4. KNS2: Faster k-NN Classification for Skewed-Class Data

In many binary classification domains, one class is much more frequent than the other. For example, in High Throughput Screening data sets, (described in section 7.2), it is far more common for the result of an experiment to be negative than positive. In detection of fraud telephone calls (Fawcett and Provost, 1997) or credit card transactions (Stolfo et al., 1997), the number of legitimate transactions is far more common than fraudulent ones. In insurance risk modeling (Pednault et al., 2000), a very small percentage of the policy holders file one or more claims in a given time period. There are many other examples of domains with similar intrinsic imbalance, and therefore, classification with a skewed distribution is important. Various researches have been focused on designing clever

```
Procedure BallKNN (PS^in, Node)
begin
  if (D^Node_minp ≥ D_sofar) then          /* If this condition is satisfied, then impossible
    Return PS^in unchanged.                  for a point in Node to be closer than the
                                             previously discovered k^th nearest neighbor.*/
  else if (Node is a leaf)
    PS^out = PS^in
    ∀x ∈ Points(Node)
    if (| x − q |< D_sofar) then           /* If a leaf, do a naive linear scan */
      add x to PS^out
      if (| PS^out |== k+1) then
        remove furthest neighbor from PS^out
        update D_sofar
  else                                     /*If a non-leaf, explore the nearer of the two
    node_1 = child of Node closest to q      child nodes, then the further. It is likely that
    node_2 = child of Node furthest from q   further search will immediately prune itself.*/
    PS^temp = BallKNN(PS^in, node_1)
    PS^out = BallKNN(PS^temp, node_2)
end
```

Figure 2: A call of BallKNN({},Root) returns the k nearest neighbors of q in the ball-tree.

methods to solve this type of problem (Cardie and Howe, 1997). The new algorithm introduced in this section, KNS2, is designed to accelerate k-NN based classification in such skewed data scenarios.

KNS2 answers type(b) question described in the introduction, namely, "How many of the k nearest neighbors are in the positive class?" The key idea of KNS2 is we can answer question (b) without explicitly finding the k-NN set.

KNS2 attacks the problem by building two ball-trees: A *Postree* for the points from the positive (small) class, and a *Negtree* for the points from the negative (large) class. Since the number of points from the positive class(small) is so small, it is quite cheap to find the exact k nearest positive points of q by using KNS1. And the idea of KNS2 is first search *Postree* using KNS1 to find the k nearest positive neighbors set $Posset_k$, and then search *Negtree* while using $Posset_k$ as bounds to prune nodes far away, and at the same time estimating the number of negative points to be inserted to the true nearest neighbor set. The search can be stopped as soon as we get the answer to question (b). Empirically, much more pruning can be achieved by KNS2 than conventional ball-tree search. A concrete description of the algorithm is as follows:

Let $Root_{pos}$ be the root of *Postree*, and $Root_{neg}$ be the root of *Negtree*. Then, we classify a new query point q in the following fashion

- Step 1 — **" Find positive"**: Find the k nearest positive class neighbors of q (and their distances to q) using conventional ball-tree search.
- Step 2 — **"Insert negative"**: Do sufficient search on the negative tree to prove that the number of positive data points among k nearest neighbors is n for some value of n.

Step 2 is achieved using a new recursive search called *NegCount*. In order to describe *NegCount* we need the following four definitions.

- **The Dists Array.** *Dists* is an array of elements $Dists_1 \ldots Dists_k$ consisting of the distances to the k nearest positive neighbors found so far of q, sorted in increasing order of distance. For notational convenience we will also write $Dists_0 = 0$ and $Dists_{k+1} = \infty$.
- **Pointset V.** Define pointset V as the set of points in the negative balls visited so far in the search.
- **The Counts Array (n,C)** $(n \leq k+1)$. C is an array of counts containing n+1 array elements $C_0, C_1, ...C_n$. Say *(n,C)* summarize interesting negative points for pointset V if and only if

  1. $\forall i = 0, 1, ..., n,$
  $$C_i = \mid V \cap \{x : \mid x - q \mid < Dists_i\} \mid \quad (6)$$
  Intuitively C_i is the number of points in V whose distances to q are closer than $Dists_i$. In other words, C_i is the number of negative points in V closer than the i^{th} positive neighbor to q.

  2. $C_i + i \leq k (i < n), C_n + n > k.$
  This simply declares that the length n of the C array is as short as possible while accounting for the k members of V that are nearest to q. Such an n exists since $C_0 = 0$ and $C_{k+1} =$ Total number of negative points. To make the problem interesting, we assume that the number of negative points and the number of positive points are both greater than k.

- $D^{\mathbf{Node}}_{\mathbf{minp}}$ and $D^{\mathbf{Node}}_{\mathbf{maxp}}$

  Here we will continue to use $D^{\text{Node}}_{\text{minp}}$ which is defined in equation (4).
  Symmetrically, we also define $D^{\text{Node}}_{\text{maxp}}$ as follows:

  $$\text{Let } D^{Node}_{maxp} = \begin{cases} \min(|q - Node.Pivot| + Node.Radius,\ D^{Node.parent}_{maxp}) & if\ Node \neq Root \\ |q - Node.Pivot| + Node.Radius & if\ Node == Root \end{cases} \quad (7)$$

  $D^{\text{Node}}_{\text{maxp}}$ is the maximum possible distance from any point in Node to q. This is computed using the bound in Equation (1) and the property of a ball-tree that all the points covered by a node must be covered by its parent. This property implies that $D^{\text{Node}}_{\text{maxp}}$ will never be greater than the maximum possible distance of its ancestors.

  Figure 3 gives a good example. There are 3 nodes p, $c1$ and $c2$. $c1$ and $c2$ are p's children. q is the query point. In order to compute D^{c1}_{minp}, first we compute $|q - c1.pivot| - c1.radius$,

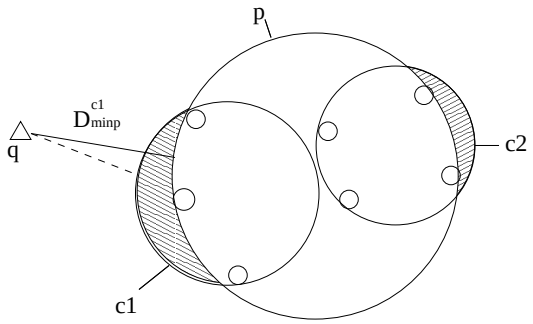


Figure 3: An example to illustrate how to compute D^{Node}_{minp}

which is the dotted line in the figure, but D^{c1}_{minp} can be further bounded by D^{p}_{minp}, since it is impossible for any point to be in the shaded area. Similarly, we get the equation for D^{c1}_{maxp}. $D^{\text{Node}}_{\text{minp}}$ and $D^{\text{Node}}_{\text{maxp}}$ are used to estimate the counts array (n,C). Again we take advantage of the triangle inequality of ball-tree. For any Node, if there exists an i ($i \in [1,n]$), such that $Dists_{i-1} \leq D^{\text{Node}}_{\text{maxp}} < Dists_i$, then for $\forall x \in$ *Points(Node)*, $Dists_{i-1} \leq \mid x - \mathsf{q} \mid < Dists_i$. According to the definition of C, we can add $\mid$ *Points(Node)* $\mid$ to $C_i, C_{i+1}, \ldots C_n$. The function of $D^{\text{Node}}_{\text{minp}}$ similar to KNS1, is used to help prune uninteresting nodes.

Step 2 of KNS2 is implemented by the recursive function below:

$$(n^{out}, C^{out}) = NegCount(n^{in}, C^{in}, Node, j_{parent}, Dists)$$

See Figure 4 for the detailed implementation of NegCount.
Assume that on entry (n^{in}, C^{in}) summarize interesting negative points for pointset V, where V is the set of points visited so far during the search. This algorithm efficiently ensures that, on exit, (n^{out}, C^{out}) summarize interesting negative points for $V \cup$ *Points(Node)*. In addition, j_{parent} is a temporary variable used to prevent multiple counts for the same point. This variable relates to the implementation of KNS2, and we do not want to go into the details in this paper.

We can stop the procedure when n^{out} becomes 1 (which means all the k nearest neighbors of q are in the negative class) or when we run out of nodes. n^{out} represents the number of positive points in the k nearest neighbors of q.The top-level call is

$$NegCount(k, C^0, NegTree.Root, k+1, Dists)$$

where C^0 is an array of zeroes and *Dists* are defined in step 2 and obtained by applying KNS1 to the *Postree*.

There are at least two situations where this algorithm can run faster than simply finding k-NN . First of all, when $n = 1$, we can stop and exit, since this means we have found at least k negative points closer than the nearest positive neighbor to q. Notice that the k negative points we have found are not necessarily the exact k nearest neighbors to q, but this won't change the answer to

Procedure NegCount ($n^{in}, C^{in}, Node, j_{parent}, Dists$)
begin
  n^{out} := n^{in} /* Variables to be returned by the search.
  C^{out} := C^{in} Initialize them here. */

  Compute $D^{\text{Node}}_{\text{minp}}$ and $D^{\text{Node}}_{\text{maxp}}$
  Search for $i, j \in [1, n^{out}]$, such that
  $Dists_{\text{i-1}} \leq D^{\text{Node}}_{\text{minp}} < Dists_{\text{i}}$
  $Dists_{\text{j-1}} \leq D^{\text{Node}}_{\text{maxp}} < Dists_{\text{j}}$

  For all index $\in [j, j_{parent})$ /* Re-estimate C^{out} */
    Update $C^{out}_{index} := C^{out}_{index} +$ | *Points(Node)* | /* Only update the count less than j_{parent}
  Update n^{out}, such that to avoid counting twice. */
    $C^{out}_{n^{out}-1} + (n^{out} - 1) \leq k, C^{out}_{n^{out}} + n^{out} > k$

  Set $Dists_{n^{out}}$:= ∞

  (1) **if** ($n^{out} == 1$) /* At least k negative points closer to q
    Return(1, C^{out}) than the closest positive one: done! */
  (2) **if** ($i == j$) /* Node is located between two adjacent
    Return(n^{out}, C^{out}) positive points, no need to split. */
  (3) **if**(Node is a leaf)
    Forall $x \in Points(Node)$
      Compute | $x - q$ |
    Update and return (n^{out}, C^{out})
  (4) **else**
    $node_1$:= child of Node closest to q
    $node_2$:= child of Node furthest from q
    (n^{temp}, C^{temp}) := NegCount($n^{in}, C^{in}, node_1, j, Dists$)
    if ($n^{temp} == 1$)
      Return (1, C^{out})
    else (n^{out}, C^{out}) := NegCount($n^{temp}, C^{temp}, node_2, j, Dists$)
end

Figure 4: Procedure NegCount.

our question. This situation happens frequently for skewed data sets. The second situation is as follows: A Node can also be pruned if it is located exactly between two adjacent positive points, or it is farther away than the n^{th} positive point. This is because that in these situations, there is no need to figure out which negative point is closer within the Node. Especially as n gets smaller, we have more chance to prune a node, because $Dists_{n^{in}}$ decreases as n^{in} decreases.

Omachi and Aso (2000) proposed a k-NN method based on branch and bound. For simplicity, we call their algorithm KNSV. KNSV is similar to KNS2, in that for the binary class case, it also

builds two trees, one for each class. For consistency, let's still call them *Postree* and *Negtree*. KNSV first searches the tree whose center of gravity is closer to q. Without loses of generality, we assume *Negtree* is closer, so KNSV will search *Negtree* first. Instead of fully exploring the tree, it does a greedy depth first search only to find k candidate points. Then KNSV moves on to search *Postree*. The search is the same as conventional ball-tree search (KNS1), except that it uses the k^{th} candidate negative point to bound the distance. After the search of *Postree* is done. KNSV counts how many of the k nearest neighbors so far are from the negative class. If the number is more than $k/2$, the algorithm stops. Otherwise, KNSV will go back to search *Negtree* for the second time, this time fully search the tree. KNSV has advantages and disadvantages. The first advantage is that it is simple, and thus it is easy to extend to many-class case. Also if the first guess of KNSV is correct and the k candidate points are good enough to prune away many nodes, it will be faster than conventional ball-tree search. But there are some obvious drawbacks of the algorithm. First, the guess of the winner class is only based on which class's center of gravity is the closest to q. Notice that this is a pure heuristic, and the probability of making a mistake is high. Second, using a greedy search to find the k candidate nearest neighbors has a high risk, since these candidates might not even be close to the true nearest neighbors. In that case, the chance for pruning away nodes from the other class becomes much smaller. We can imagine that in many situations, KNSV will end up searching the first tree for yet another time. Finally, we want to point out that KNSV claims it can perform well for many-class nearest neighbors, but this is based on the assumption that the winner class contains at least $k/2$ points within the nearest neighbors, which is often not true for the many-class case. Comparing to KNSV, KNS2's advantages are (i) it uses the skewness property of a data set, which can be robustly detected before the search, and (ii) more careful design gives KNS2 more chance to speedup the search.

5. KNS3: Are at Least t of the k Nearest Neighbors Positive?

In this paper's second new algorithm, we remove KNS2's constraint of an assumed skewness in the class distribution. Instead, we answer a weaker question: "are at least t of the k nearest neighbors positive?", where the questioner must supply t and k. In the usual k-NN rule, t represents a majority with respect to k, but here we consider the slightly more general form which might be used for example during classification with known false positive and false negative costs.

In KNS3, we define two important quantities:

$$D_t^{pos} = distance\ of\ the\ t^{th}\ nearest\ positive\ neighbor\ of\ \mathsf{q} \quad (8)$$

$$D_m^{neg} = distance\ of\ the\ m^{th}\ nearest\ negative\ neighbor\ of\ \mathsf{q} \quad (9)$$

where $m+t=k+1$.

Before introducing the algorithm, we state and prove an important proposition, which relate the two quantities D_t^{pos} and D_m^{neg} with the answer to KNS3.

Proposition 1 *$D_t^{pos} \leq D_m^{neg}$ if and only if at least t of the k nearest neighbors of* q *from the positive class.*

Proof:
If $D_t^{pos} \leq D_m^{neg}$, then there are at least t positive points closer than the m^{th} negative point to q. This also implies that if we draw a ball centered at q, and with its radius equal to D_m^{neg}, then there are exactly m negative points and at least t positive points within the ball. Since $t+m=k+1$, if we use D_k to denote the distance of the k^{th} nearest neighbor, we get $D_k \leq D_m^{neg}$, which means that there are at most $m-1$ of the k nearest neighbors of q from the negative class. It is equivalent to say that there are at least t of the k nearest neighbors of q are from the positive class. On the other hand, if there are at least t of the k nearest neighbors from the positive class, then $D_t^{pos} \leq D_k$, the number of negative points is at most $k-t<m$, so $D_k \leq D_m^{neg}$. This implies that $D_t^{pos} \leq D_m^{neg}$ is true. ■

Figure 5 provides an illustration. In this example, $k=5, t=3$. We use black dots to represent positive points, and white dots to represent negative points. The reason to redefine the problem of

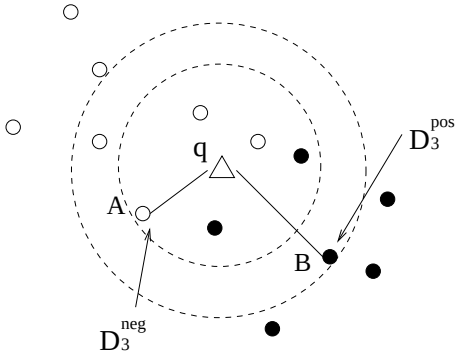


Figure 5: An example of D_t^{pos} and D_m^{neg}

KNS3 is to transform a k nearest neighbor searching problem to a much simpler counting problem. In fact, in order to answer the question, we do not even have to compute the exact value of D_t^{pos} and D_m^{neg}, instead, we can estimate them. We define $Lo(D_t^{pos})$ and $Up(D_t^{pos})$ as the lower and upper bounds of D_t^{pos}, and similarly we define $Lo(D_m^{neg})$ and $Up(D_m^{neg})$ as the lower and upper bounds of D_m^{neg}. If at any point, $Up(D_t^{pos}) \leq Lo(D_m^{neg})$, we know $D_t^{pos} \leq D_m^{neg}$, on the other hand, if $Up(D_m^{neg}) \leq Lo(D_t^{pos})$, we know $D_m^{neg} \leq D_t^{pos}$.

Now our computational task is to efficiently estimate $Lo(D_t^{pos})$, $Up(D_t^{pos})$, $Lo(D_m^{neg})$ and $Up(D_m^{neg})$. And it is very convenient for a ball-tree structure to do so. Below is the detailed description:

At each stage of KNS3 we have two sets of balls in use called P and N, where P is a set of balls from *Postree* built from positive data points, and N consists of balls from *Negtree* built from negative data points.

Both sets have the property that if a ball is in the set, then neither its ball-tree ancestors nor descendants are in the set, so that each point in the training set is a member of one or zero balls in $P \cup N$. Initially, $P = \{PosTree.root\}$ and $N = \{NegTree.root\}$. Each stage of KNS3 analyzes P to estimate $Lo(D_t^{pos})$, $Up(D_t^{pos})$, and analyzes N to estimate $Lo(D_m^{neg})$, $Up(D_m^{neg})$. If possible, KNS3 terminates with the answer, else it chooses an appropriate ball from P or N, and replaces that ball with its two children, and repeats the iteration. Figure (6) shows one stage of KNS3. The balls

involved are labeled *a* through *g* and we have

$$P = \{a, b, c, d\}$$
$$N = \{e, f, g\}$$

Notice that although c and d are inside b, they are not descendants of b. This is possible because when a ball is splitted, we only require the pointset of its children be disjoint, but the balls covering the children node may be overlapped.

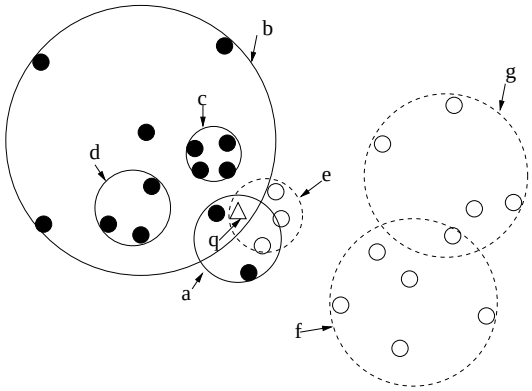


Figure 6: A configuration at the start of a stage.

In order to compute $Lo(D_t^{pos})$, we need to sort the balls $u \in P$, such that

$$\forall u_i, u_j \in P, i < j \Rightarrow D^i_{minp} \leq D^j_{minp}$$

Then

$$Lo(D_t^{pos}) = D^{u_j}_{minp}, \text{ where } \sum_{i=1}^{j-1} \mid Points(u_i) \mid < t \text{ and } \sum_{i=1}^{j} \mid Points(u_i) \mid \geq t$$

Symmetrically, in order to compute $Up(D_t^{pos})$, we sort $u \in P$, such that

$$\forall u_i, u_j \in P, i < j \Rightarrow D^i_{maxp} \leq D^j_{maxp}.$$

Then

$$Up(D_t^{pos}) = D^{u_j}_{maxp}, \text{ where } \sum_{i=1}^{j-1} \mid Points(u_i) \mid < t \text{ and } \sum_{i=1}^{j} \mid Points(u_i) \mid \geq t$$

Similarly, we can compute $Lo(D_m^{neg})$ and $Up(D_m^{neg})$.

To illustrate this, it is useful to depict a ball as an interval, where the two ends of the interval denote the minimum and maximum possible distances of a point owned by the ball to the query. Figure 5(a) shows an example. Notice, we also mark "+5" above the interval to denote the number of points owned by the ball *B*. After we have this representation, both *P* and *N* can be represented as a set of intervals, each interval corresponds to a ball. This is shown in 5(b). For example, the second horizontal line denotes the fact that ball *b* contains four positive points, and that the distance from

any location in b to q lies in the range $[0,5]$. The value of $Lo(D_t^{pos})$ can be understood as the answer to the following question: what if we tried to slide all the positive points within their bounds as far to the left as possible, where would the t^{th} closest positive point lie? Similarly, we can estimate $Up(D_t^{pos})$ by sliding all the positive points to the right ends within their bounds.

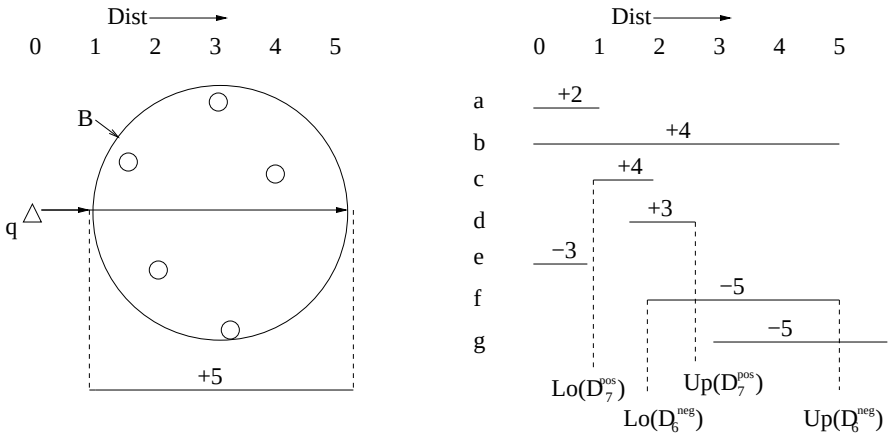


Figure 7: (a) The interval representation of a ball B. (b) The interval representation of the configuration in Figure 6

.

For example, in Figure 6, let $k = 12$ and $t = 7$. Then $m = 12-7+1 = 6$. We can estimate ($Lo(D_7^{pos})$, $Up(D_7^{pos})$) and ($Lo(D_6^{neg})$, $Up(D_6^{neg})$), and the results are shown in Figure 5. Since the two intervals ($Lo(D_7^{pos})$, $Up(D_7^{pos})$) and ($Lo(D_6^{neg})$,$Up(D_6^{neg})$) have overlap now, no conclusion can be made at this stage. Further splitting needs to be done to refine the estimation.

Below is the pseudo code of KNS3 algorithm: We define a loop procedure called *PREDICT* with the following input and output.

$$Answer = PREDICT(P,N,t,m)$$

The *Answer*, a boolean value, is TRUE, if there are at least t of the k nearest neighbors from the positive class; and False otherwise. Initially, P = $\{PosTree.root\}$ and N = $\{NegTree.root\}$. The threshold t is given, and $m = k-t+1$.

Before we describe the algorithm, we first introduce two definitions.
Define:

$$(Lo(D_i^S), Up(D_i^S)) = Estimate\_bound(S,i) \tag{10}$$

Here S is either set P or N, and we are interested in the i^{th} nearest neighbor of q from set S. The output is the lower and upper bounds. The concrete procedure for estimating the bounds was just described.

Notice that the estimation of the upper and lower bounds could be very loose in the beginning,

and will not give us enough information to answer the question. In this case, we will need to split a ball-tree node and re-estimate the bounds. With more and more nodes being splitted, our estimation becomes more and more precise, and the procedure can be stopped as soon as $Up(D_t^{pos}) \leq Lo(D_m^{neg})$ or $Up(D_m^{neg}) \leq Lo(D_t^{pos})$. The function of $Pick(P,N)$ below is to choose one node either from P or N to split. There are different strategies for picking a node, for simplicity, our implementation only randomly pick a node to split.
Define:

$$split\_node = Pick(P,N) \tag{11}$$

Here split_node is the node chosen to be split. See Figure 8.

```
Procedure PREDICT ( P, N, t, m)
begin
  Repeat
    (Lo(D_t^pos), Up(D_t^pos)) = Estimate_bound(P, t)        /* See Definition 10. */
    (Lo(D_m^neg), Up(D_m^neg)) = Estimate_bound(N, m)
    if (Up(D_t^pos) ≤ Lo(D_m^neg)) then
      Return TRUE
    if (Up_(m^neg) ≤ Lo(D_m^neg)) then
      Return FALSE

    split_node = Pick(P, N)
    remove split_node from P or N
    insert split_node.child1 and split_node.child2 to P or N
end
```

Figure 8: Procedure PREDICT.

.

Our explanation of KNS3 was simplified for clarity. In order to avoid frequent searches over the full lengths of sets N and P, they are represented as priority queues. Each set in fact uses two queues: one prioritized by D^u_{maxp} and the other by D^u_{minp}.This ensures that the costs of all argmins, deletions and splits are logarithmic in the queue size.

Some people may ask the question: “It seems that KNS3 has more advantages than KNS2, it removes the assumption of skewness of the data set. In general, it has more chances to prune away nodes, etc. Why we still need KNS2?” The answer is KNS2 does have its own advantages. It answers a more difficult question than KNS3. To know exact how many of the nearest neighbors are from the positive class can be especially useful when the threshold for deciding a class is not known. In that case, KNS3 doesn’t work at all since we can not provide a static t for answering the question (c). But KNS2 can still work very well. On the other hand, the implementation of KNS2 is much simpler than KNS3. For instance, it does not need the priority queues we just described. So there does exist some cases where KNS2 is faster than KNS3.

6. Experimental Results

To evaluate our algorithms, we used both real data sets (from UCI and KDD repositories) and also synthetic data sets designed to exercise the algorithms in various ways.

6.1 Synthetic Data Sets

We have six synthetic data sets. The first synthetic data set we have is called `Ideal`, as illustrated in Figure 6.1(a). All the data in the left upper area are assigned to the positive class, and all the data in the right lower area are assigned to the negative class. The second data set we have is called `Diag2d`, as illustrated in Figure 6.1(b). The data are uniformly distributed in a 10 by 10 square. The data above the diagonal are assigned to the positive class, below diagonal are assigned to the negative class. We made several variants of Diag2d to test the robustness of KNS3. `Diag2d(10%)` has 10% data of `Diag2d`. `Diag3d` is a cube with uniformly distributed data and classified by a diagonal-plane. `Diag10d` is a 10 dimensional hypercube with uniformly distributed data and classified by a hyper-diagonal-plane. `Noise-diag2d` has the same data as `Diag2d(10%)`, but 1% of the data was assigned to the wrong class.

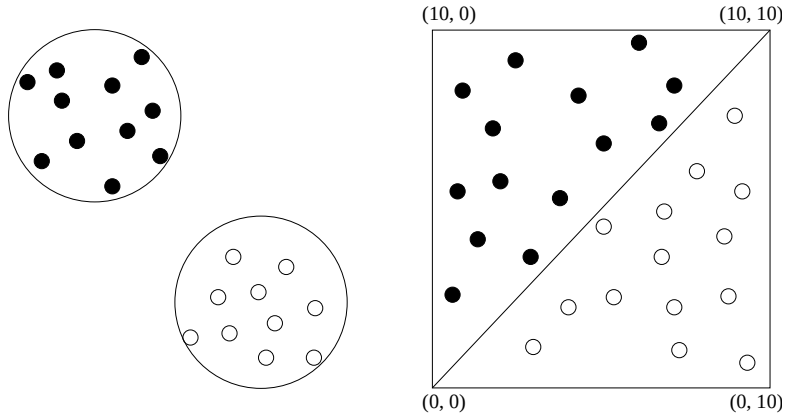


Figure 9: Synthetic Data Sets

Table6.1 is a summary of the data sets in the empirical analysis.

6.2 Real-World Data Sets

We used UCI & KDD data (listed in Table 6.2), but we also experimented with data sets of particular current interest within our laboratory.

Life Sciences. These were proprietary data sets (*ds1* and *ds2*) similar to the publicly available Open Compound Database provided by the National Cancer Institute (NCI Open Compound Database, 2000). The two data sets are sparse. We also present results on data sets derived from *ds1*, denoted *ds1.10pca*, *ds1.100pca* and *ds2.100anchor* by linear projection using principal component analysis

Data Set	Num. of records	Num. of Dimensions	Num. of positive	Num.pos/Num.neg
Ideal	10000	2	5000	1
Diag2d(10%)	10000	2	5000	1
Diag2d	100000	2	50000	1
Diag3d	100000	3	50000	1
Diag10d	100000	10	50000	1
Noise2d	10000	2	5000	1

Table 1: Synthetic Data Sets

(PCA).

Link Detection. The first, Citeseer, is derived from the Citeseer web site (Citeseer,2002) and lists the names of collaborators on published materials. The goal is to predict whether J_Lee (the most common name) was a collaborator for each work based on who else is listed for that work. We use *J_Lee.100pca* to represent the linear projection of the data to 100 dimensions using PCA. The second link detection data set is derived from the Internet Movie Database (IMDB, 2002) and is denoted *imdb* using a similar approach, but to predict the participation of Mel Blanc (again the most common participant).

UCI/KDD data. We use four large data sets from KDD/UCI repository (Bay, 1999). The data sets can be identified from their names. They were converted to binary classification problems. Each categorical input attribute was converted into n binary attributes by a 1-of-n encoding (where n is the number of possible values of the attribute).

1. *Letter* originally had 26 classes: A-Z. We performed binary classification using the letter A as the positive class and "Not A" as negative.

2. *Ipums* (from ipums.la.97). We predict *farm status*, which is binary.

3. *Movie* is a data set from (informedia digital video library project, 2001). The TREC-2001 Video Track organized by NIST shot boundary Task. 4 hours of video or 13 MPEG-1 video files at slightly over 2GB of data.

4. *Kdd99(10%)* has a binary prediction: Normal vs. Attack.

6.3 Methodology

The data set *ds2* is particular interesting, because its dimension is 1.1×10^6. Our first experiment is especially designed for it. We use k=9, and $t = \lceil k/2 \rceil$, then we print out the distribution of time taken for queries of three algorithms: KNS1, KNS2, and KNS3. This is aimed at understanding the range of behavior of the algorithms under huge dimensions (some queries will be harder, or take longer, for an algorithm than other queries). We randomly took 1% negative records (881) and 50% positive records (105) as test data (total 986 points), and train on the remaining 87372 data points.

Data Set	Num. of records	Num. of Dimensions	Num.of positive	Num.pos/Num.neg
ds1	26733	6348	804	0.03
ds1.10pca	26733	10	804	0.03
ds1.100pca	26733	100	804	0.03
ds2	88358	1.1×10^6	211	0.002
ds2.100anchor	88358	100	211	0.002
J_Lee.100pca	181395	100	299	0.0017
Blanc_Mel	186414	10	824	0.004
Data Set	**Num. records**	**Num. of Dimensions**	**Num.of positive**	**Num.pos/Num.neg**
Letter	20000	16	790	0.04
Ipums	70187	60	119	0.0017
Movie	38943	62	7620	0.24
Kdd99(10%)	494021	176	97278	0.24

Table 2: Real Data Sets

For our second set of experiments, we did 10-fold cross-validation on all the data sets. For each data set, we tested $k = 9$ and $k = 101$, in order to show the effect of a small value and a large value. For KNS3, we used $t = \lceil k/2 \rceil$: a data point is classified as positive iff the majority of its k nearest neighbors are positive. Since we use cross-validation, thus each experiment required R k-NN classification queries (where R is the umber of records in the data set) and each query involved the k-NN among $0.9R$ records. A naive implementation with no ball trees would thus require $0.9R^2$ distance computations. We want to emphasize here that these algorithms are all exact. No approximations were used in the classifications.

6.4 Results

Figure 10 shows the histograms of times and speed-ups for queries on the ds2 data set. For Naive k-NN , all the queries take 87372 distance computations. For KNS1, all the queries take more than 1.0×10^4 distance computations, (the average number of distances computed is 1.3×10^5) which is greater than 87372 and thus traditional ball-tree search is worse than "naive" linear scan. For KNS2, most of the queries take less than 4.0×10^4 distance computations, a few points take longer time. The average number of distances computed is 6233. For KNS3, all the queries take less than 1.0×10^4 distance computations, the average number of distances computed is 3411. The lower three figures illustrate speed-up achieved for KNS1, KNS2 and KNS3 over naive linear scan. The figures show the distribution of the speedup obtained for each query. From 10(d) we can see that on average, KNS1 is even slower than the naive algorithm. KNS2 can get from 2- to 250-fold speedups. On average, it has a 14-fold speedup. KNS3 can get from 2- to 2500-fold speedups. On average, it has a 26-fold speedups.

Table 6.4 shows the results for the second set of experiments. The second column lists the computational cost of naive k-NN , both in terms of the number of distance computations and the wall-clock

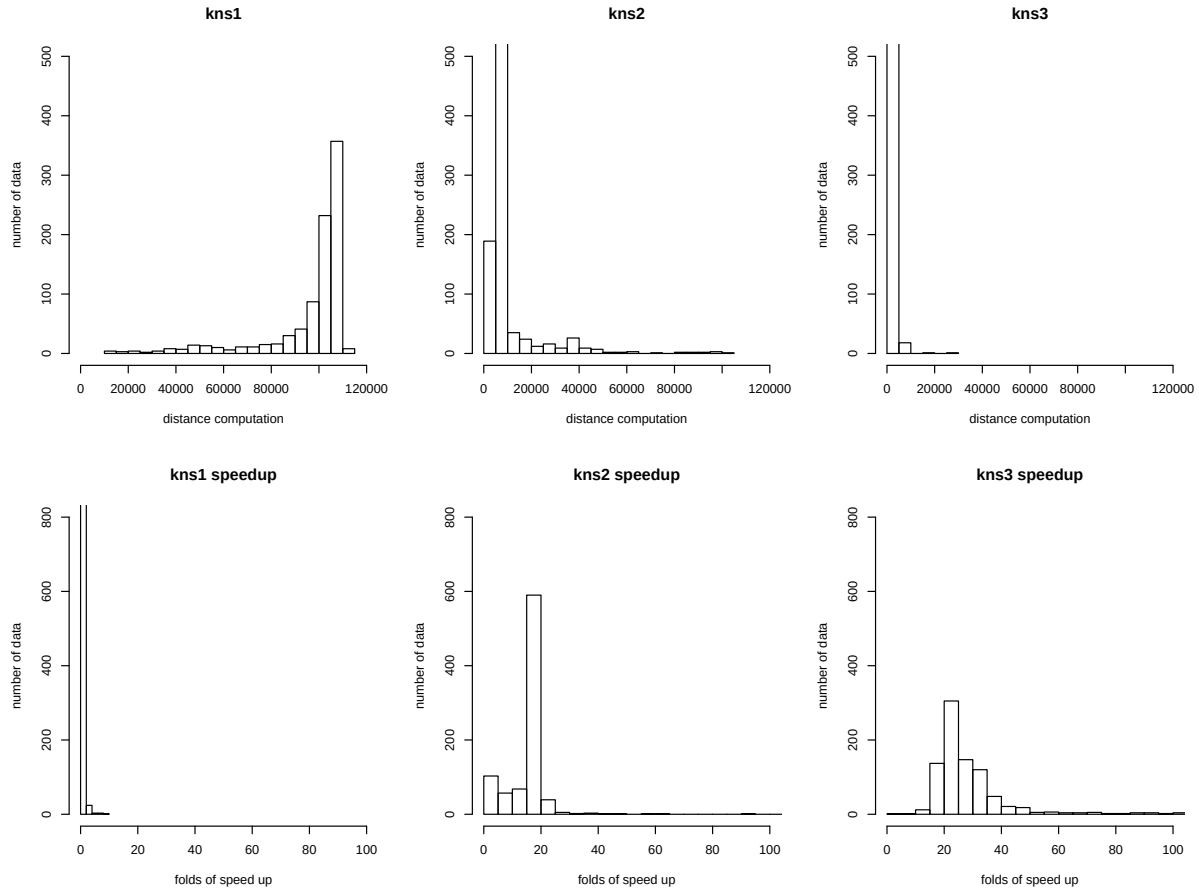


Figure 10: (a) Distribution of times taken for queries of KNS1 (b) Distribution of times taken for queries of KNS2 (c) Distribution of times taken for queries of KNS3 (d) Distribution of speedup for queries achieved for KNS1 (e) Distribution of speedup for queries achieved for KNS2 (f) Distribution of speedup for queries achieved for KNS3

time on an unloaded 2 GHz Pentium. We then examine the speedups of KNS1 (traditional use of a ball-tree) and our two new ball-tree methods (KNS2 and KNS3). Generally speaking, the speedup achieved for distance computations on all three algorithms are greater than the corresponding speedup for wall-clock time. This is expected, because the wall-clock time also includes the time for building ball trees, generating priority queues and searching. We can see that for the synthetic data sets, KNS1 and KNS2 yield 2-700 fold speedup over naive. KNS3 yields a 2-4500 fold speedup. Notice that KNS2 can't beat KNS1 for these data sets, because KNS2 is designed to speedup k-NN search on data sets with unbalanced output classes. Since all the synthetic data sets have equal number of data from positive and negative classes, KNS2 has no advantage.

It is notable that for some high-dimensional data sets, KNS1 does not produce an acceleration

over naive. KNS2 and KNS3 do, however, and in some cases they are hundreds of times faster than KNS1.

		NAIVE		**KNS1**		**KNS2**		**KNS3**	
		dists	time (secs)	dists speedup	time speedup	dists speedup	time speedup	dists speedup	time speedup
ideal	k=9	9.0×10^7	30	96.7	56.5	112.9	78.5	4500	**486**
	k=101			23.0	10.2	24.7	14.7	4500	**432**
Diag2d(10%)	k=9	9.0×10^7	30	91	51.1	88.2	**52.4**	282	27.1
	k=101			22.3	8.7	21.3	9.3	167.9	**15.9**
Diag2d	k=9	9.0×10^9	3440	738	366	664	**372**	2593	287
	k=101			202.9	104	191	107.5	2062	**287**
Diag3d	k=9	9.0×10^9	4060	361	**184.5**	296	**184.5**	1049	176.5
	k=101			111	56.4	95.6	48.9	585	**78.1**
Diag10d	k=9	9.0×10^9	6080	7.1	**5.3**	7.3	5.2	12.7	2.2
	k=101			3.3	**2.5**	3.1	1.9	6.1	0.7
Noise2d	k=9	9.0×10^7	40	91.8	20.1	79.6	30.1	142	**42.7**
	k=101			22.3	4	16.7	4.5	94.7	**43.5**
ds1	k=9	6.4×10^8	4830	1.6	1.0	4.7	3.1	12.8	**5.8**
	k=101			1.0	0.7	1.6	1.1	10	**4.2**
ds1.10pca	k=9	6.4×10^8	420	11.8	11.0	33.6	**21.4**	71	20
	k=101			4.6	3.4	6.5	4.0	40	**6.1**
ds1.100pca	k=9	6.4×10^8	2190	1.7	1.8	7.6	7.4	23.7	**29.6**
	k=101			0.97	1.0	1.6	1.6	16.4	**6.8**
ds2	k=9	8.5×10^9	105500	0.64	0.24	14.0	2.8	25.6	**3.0**
	k=101			0.61	0.24	2.4	0.83	28.7	**3.3**
ds2.100-	k=9	7.0×10^9	24210	15.8	14.3	185.3	144	580	**311**
	k=101			10.9	14.3	23.0	19.4	612	**248**
J_Lee.100-	k=9	3.6×10^{10}	142000	2.6	2.4	28.4	**27.2**	15.6	12.6
	k=101			2.2	1.9	12.6	11.6	37.4	**27.2**
Blanc__Mel	k=9	3.8×10^{10}	44300	3.0	3.0	47.5	**60.8**	51.9	60.7
	k=101			2.9	3.1	7.1	33	203	**134.0**
Letter	k=9	3.6×10^8	290	8.5	7.1	42.9	**26.4**	94.2	25.5
	k=101			3.5	2.6	9.0	5.7	45.9	**9.4**
Ipums	k=9	4.4×10^9	9520	195	136	665	501	1003	**515**
	k=101			69.1	50.4	144.6	121	5264	**544**
Movie	k=9	1.4×10^9	3100	16.1	13.8	29.8	**24.8**	50.5	22.4
	k=101			9.1	7.7	10.5	8.1	33.3	**11.6**
Kddcup99	k=9	2.7×10^{11}	1670000	4.2	4.2	574	**702**	4	4.1
(10%)	k=101			4.2	4.2	187.7	**226.2**	3.9	3.9

Table 3: Number of distance computations and wall-clock-time for Naive *k*-NN classification (2nd column). Acceleration for normal use of KNS1 (in terms of num. distances and time). Accelerations of new methods KNS2 and KNS3 in other columns. Naive times are independent of *k*.

7. Comments and Related Work

Why *k*-NN ? *k*-NN is an old classification method, often not achieving the highest possible accuracies when compared to more complex methods. Why study it? There are many reasons. *k*-NN is a useful sanity check or baseline against which to check more sophisticated algorithms *provided* *k*-NN is tractable. It is often the first line of attack in a new complex problem due to its simplicity and flexibility. The user need only provide a sensible distance metric. The method is easy to interpret once this distance metric is understood. We have already mentioned its compelling theo-

retical properties, which explains its surprisingly good performance in practice in many cases. For these reason and others, *k*-NN is still popular in some fields that need classification, for example Computer Vision and QSAR analysis of High Throughput Screening data (e.g., Zheng and Tropsha, 2000). Finally, we believe that the same insights that accelerate *k*-NN will apply to more modern algorithms. From a theoretical viewpoint, many classification algorithms can be viewed simply as the nearest-neighbor method with a certain broader notion of distance function; see for example Baxter and Bartlett (1998) for such a broad notion. RKHS kernel methods use another example of a broadened notion of distance function. More concretely, we have applied similar ideas to speed up nonparametric Bayes classifiers, in work to be submitted.

Applicability of other proximity query work. For the problem of "find the *k* nearest datapoints" (as opposed to our question of "perform *k*-NN or Kernel classification") in high dimensions, the frequent failure of a traditional ball-tree to beat naive has lead to some very ingenious and innovative alternatives, based on random projections, hashing discretized cubes, and acceptance of approximate answers. For example Gionis et al. (1999) gives a hashing method that was demonstrated to provide speedups over a ball-tree-based approach in 64 dimensions by a factor of 2-5 depending on how much error in the approximate answer was permitted. Another approximate *k*-NN idea is in Arya et al. (1998), one of the first *k*-NN approaches to use a priority queue of nodes, in this case achieving a 3-fold speedup with an approximation to the true *k*-NN . In (Liu et al., 2004a), we introduced a variant of ball-tree structures which allow non-backtracking search to speed up approximate nearest neighbor, and we observed up to 700-fold accelerations over conventional ball-tree based *k*-NN . Similar idea has been proposed by Indyk (2001). However, these approaches are based on the notion that any points falling within a factor of $(1+\varepsilon)$ times the true nearest neighbor distance are acceptable substitutes for the true nearest neighbor. Noting in particular that distances in high-dimensional spaces tend to occupy a decreasing range of continuous values (Hammersley, 1950), it remains unclear whether schemes based upon the absolute values of the distances rather than their *ranks* are relevant to the classification task. Our approach, because it need not find the *k*-NN to answer the relevant statistical question, finds an answer without approximation. The fact that our methods are easily modified to allow $(1+\varepsilon)$ approximation in the manner of Arya et al. (1998) suggests an obvious avenue for future research.

No free lunch. For uniform high-dimensional data no amount of trickery can save us. The explanation for the promising empirical results is that all the inter-dependences in the data mean we are working in a space of much lower intrinsic dimensionality (Maneewongvatana and Mount, 2001). Note though, that in experiments not reported here, QSAR and vision *k*-NN classifiers give better performance on the original data than on PCA-projected low dimensional data, indicating that some of these dependencies are non-linear.

Summary. This paper has introduced and evaluated two new algorithms for more effectively exploiting spatial data structures during *k*-NN classification. We have shown significant speedups on high dimensional data sets without resorting to approximate answers or sampling. The result is that the *k*-NN method now scales to many large high-dimensional data sets that previously were not tractable for it, and are still not tractable for many popular methods such as support vector machines.

References

S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998. URL citeseer.ist.psu.edu/arya94optimal.html.

J. Baxter and P. Bartlett. The Canonical Distortion Measure in Feature Space and 1-NN Classification. In *Advances in Neural Information Processing Systems 10*. Morgan Kaufmann, 1998.

S. D. Bay. UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Dept of Information and Computer Science, 1999.

C. Cardie and N. Howe. Improving minority class prediction using case-specific feature weights. In *Proceedings of 14th International Conference on Machine Learning*, pages 57–65. Morgan Kaufmann, 1997. URL citeseer.nj.nec.com/cardie97improving.html.

C. L. Chang. Finding prototypes for nearest neighbor classifiers. *IEEE Trans. Computers*, C-23 (11):1179–1184, November 1974.

P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd VLDB International Conference*, September 1997.

K. Clarkson. Nearest Neighbor Searching in Metric Spaces: Experimental Results for sb(S). , 2002.

S. Cost and S. Salzberg. A Weighted Nearest Neighbour Algorithm for Learning with Symbolic Features. *Machine Learning*, 10:57–67, 1993.

T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, IT-13,no.1:21–27, 1967.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

K. Deng and A. W. Moore. Multiresolution Instance-based Learning. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 1233–1239, San Francisco, 1995. Morgan Kaufmann.

L. Devroye and T. J. Wagner. *Nearest neighbor methods in discrimination*, volume 2. P.R. Krishnaiah and L. N. Kanal, eds., North-Holland, 1982.

A. Djouadi and E. Bouktache. A fast algorithm for the nearest-neighbor classifier. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(3):277–282, March 1997.

N. R. Draper and H. Smith. *Applied Regression Analysis, 2nd ed.* John Wiley, New York, 1981.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

C. Faloutsos and D. W. Oard. A survey of information retrieval and filtering methods. Technical Report CS-TR-3514, Carnegie Mellon University Computer Science Department, 1995.

C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and William Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3 (3/4):231–262, 1994.

T. Fawcett and F. J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1 (3):291–316, 1997. URL `citeseer.nj.nec.com/fawcett97adaptive.html`.

F. P. Fisher and E. A. Patrick. A preprocessing algorithm for nearest neighbor decision rules. In *Proc. Nat'l Electronic Conf.*, volume 26, pages 481–485, December 1970.

M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *IEEE Computer*, 28:23–32, 1995.

J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

K. Fukunaga and P.M. Narendra. A Branch and Bound Algorithm for Computing K-Nearest Neighbors. *IEEE Trans. Computers*, C-24(7):750–753, 1975.

G. W. Gates. The reduced nearest neighbor rule. *IEEE Trans. Information Theory*, IT-18(5):431–433, May 1972.

A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th VLDB Conference*, 1999.

A. Gray and A. W. Moore. N-Body Problems in Statistical Learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*. Assn for Computing Machinery, April 1984.

J. M. Hammersley. The Distribution of Distances in a Hypersphere. *Annals of Mathematical Statistics*, 21:447–452, 1950.

P. E. Hart. The condensed nearest neighbor rule. *IEEE Trans. Information Theory*, IT-14(5):515–516, May 1968.

T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(6):607–615, June 1996.

P. Indyk. On approximate nearest neighbors under l_∞ norm. *Journal of Computer and System Sciences*, 63(4), 2001.

CMU informedia digital video library project. The trec-2001 video trackorganized by nist shot boundary task, 2001.

V. Koivune and S. Kassam. Nearest neighbor filters for multivariate data. In *IEEE Workshop on Nonlinear Signal and Image Processing*, 1995.

P. Komarek and A. W. Moore. Fast robust logistic regression for large sparse datasets with binary outputs. In *Artificial Intelligence and Statistics*, 2003.

C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM conference on Computer and communications security table of contents*, pages 251–261, 2003.

E. Lee and S. Chae. Fast design of reduced-complexity nearest-neighbor classifiers using triangular inequality. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(5):562–566, May 1998.

T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In *Proceedings of Neural Information Processing Systems*, 2004a.

T. Liu, K. Yang, and A. Moore. The ioc algorithm: Efficient many-class non-parametric classification for high-dimensional data. In *Proceedings of the conference on Knowledge Discovery in Databases (KDD)*, 2004b.

S. Maneewongvatana and D. M. Mount. The analysis of a probabilistic approach to nearest neighbor searching. In *In Proceedings of WADS 2001*, 2001.

A. W. Moore. The Anchors Hierarchy: Using the Triangle Inequality to Survive High-Dimensional Data. In *Twelfth Conference on Uncertainty in Artificial Intelligence*. AAAI Press, 2000.

S. Omachi and H. Aso. A fast algorithm for a k-nn classifier based on branch and bound method and computational quantity estimation. In *In Systems and Computers in Japan, vol.31, no.6, pp.1-9*, 2000.

S. M. Omohundro. Bumptrees for Efficient Function, Constraint, and Classification Learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, 1991.

S. M. Omohundro. Efficient Algorithms with Neural Network Behaviour. *Journal of Complex Systems*, 1(2):273–347, 1987.

A. M. Palau and R. R. Snapp. The labeled cell classifier: A fast approximation to k nearest neighbors. In *Proceedings of the 14th International Conference on Pattern Recognition*, 1998.

E. P. D. Pednault, B. K. Rosen, and C. Apte. Handling imbalanced data sets in insurance risk modeling, 2000.

D. Pelleg and A. W. Moore. Accelerating Exact *k*-means Algorithms with Geometric Reasoning. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*. ACM, 1999.

A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases, 1994. URL `citeseer.ist.psu.edu/pentland95photobook.html`.

F. P. Preparata and M. Shamos. *Computational Geometry*. Springer-Verlag, 1985.

Y. Qi, A. Hauptman, and T. Liu. Supervised classification for video shot segmentation. In *Proceedings of IEEE International Conference on Multimedia and Expo*, 2003.

G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour. An algorithm for a selective nearest neighbor decision rule. *IEEE Trans. Information Theory*, IT-21(11):665–669, November 1975.

G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, New York, NY, 1983.

I. K. Sethi. A fast algorithm for recognizing nearest neighbors. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-11(3):245–248, March 1981.

A. Smeulders and R. Jain, editors. *Image Databases and Multi-media Search*. World Scientific Publishing Company, 1996.

S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Credit card fraud detection using meta-learning: Issues and initial results, 1997. URL `citeseer.nj.nec.com/stolfo97credit.html`.

Y. Hamamoto S. Uchimura and S. Tomita. A bootstrap technique for nearest neighbor classifier design. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(1):73–79, 1997.

J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, 1991.

K. Woods, K. Bowyer, and W. P. Kegelmeyer Jr. Combination of multiple classifiers using local accuracy estimates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

B. Zhang and S. Srihari. Fast k-nearest neighbor classification using cluster-based trees. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(4):525–528, April 2004.

T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : PODS 1996*. ACM, 1996.

W. Zheng and A. Tropsha. A Novel Variable Selection QSAR Approach based on the K-Nearest Neighbor Principle. *J. Chem. Inf.Comput. Sci.*, 40(1):185–194, 2000.

Journal of Machine Learning Research 7 (2006) 1159–1182 Submitted 2/05; Revised 4/06; Published 7/06

A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis

Enrique Castillo CASTIE@UNICAN.ES
Department of Applied Mathematics and Computational Sciences
University of Cantabria and University of Castilla-La Mancha
Avda de Los Castros s/n, 39005 Santander, Spain

Bertha Guijarro-Berdiñas CIBERTHA@UDC.ES
Oscar Fontenla-Romero OFONTENLA@UDC.ES
Amparo Alonso-Betanzos CIAMPARO@UDC.ES
Department of Computer Science
Faculty of Informatics, University of A Coruña
Campus de Elviña s/n, 15071 A Coruña, Spain

Editor: Yoshua Bengio

Abstract

This paper introduces a learning method for two-layer feedforward neural networks based on sensitivity analysis, which uses a linear training algorithm for each of the two layers. First, random values are assigned to the outputs of the first layer; later, these initial values are updated based on sensitivity formulas, which use the weights in each of the layers; the process is repeated until convergence. Since these weights are learnt solving a linear system of equations, there is an important saving in computational time. The method also gives the local sensitivities of the least square errors with respect to input and output data, with no extra computational cost, because the necessary information becomes available without extra calculations. This method, called the Sensitivity-Based Linear Learning Method, can also be used to provide an initial set of weights, which significantly improves the behavior of other learning algorithms. The theoretical basis for the method is given and its performance is illustrated by its application to several examples in which it is compared with several learning algorithms and well known data sets. The results have shown a learning speed generally faster than other existing methods. In addition, it can be used as an initialization tool for other well known methods with significant improvements.

Keywords: supervised learning, neural networks, linear optimization, least-squares, initialization method, sensitivity analysis

1. Introduction

There are many alternative learning methods and variants for neural networks. In the case of feedforward multilayer networks the first successful algorithm was the classical backpropagation (Rumelhart et al., 1986). Although this approach is very useful for the learning process of this kind of neural networks it has two main drawbacks:

- Convergence to local minima.
- Slow learning speed.

In order to solve these problems, several variations of the initial algorithm and also new methods have been proposed. Focusing the attention on the problem of the slow learning speed, some algorithms have been developed to accelerate it:

- *Modifications of the standard algorithms*: Some relevant modifications of the backpropagation method have been proposed. Sperduti and Antonina (1993) extend the backpropagation framework by adding a gradient descent to the sigmoids steepness parameters. Ihm and Park (1999) present a novel fast learning algorithm to avoid the slow convergence due to weight oscillations at the error surface narrow valleys. To overcome this difficulty they derive a new gradient term by modifying the original one with an estimated downward direction at valleys. Also, stochastic backpropagation—which is opposite to batch learning and updates the weights in each iteration—often decreases the convergence time, and is specially recommended when dealing with large data sets on classification problems (see LeCun et al., 1998).

- *Methods based on linear least-squares*: Some algorithms based on linear least-squares methods have been proposed to initialize or train feedforward neural networks (Biegler-König and Bärmann, 1993; Pethel et al., 1993; Yam et al., 1997; Cherkassky and Mulier, 1998; Castillo et al., 2002; Fontenla-Romero et al., 2003). These methods are mostly based on minimizing the mean squared error (MSE) between the signal of an output neuron, before the output nonlinearity, and a modified desired output, which is exactly the actual desired output passed through the inverse of the nonlinearity. Specifically, in (Castillo et al., 2002) a method for learning a single layer neural network by solving a linear system of equations is proposed. This method is also used in (Fontenla-Romero et al., 2003) to learn the last layer of a neural network, while the rest of the layers are updated employing any other non-linear algorithm (for example, conjugate gradient). Again, the linear method in (Castillo et al., 2002) is the basis for the learning algorithm proposed in this article, although in this case all layers are learnt by using a system of linear equations.

- *Second order methods*: The use of second derivatives has been proposed to increase the convergence speed in several works (Battiti, 1992; Buntine and Weigend, 1993; Parker, 1987). It has been demonstrated (LeCun et al., 1991) that these methods are more efficient, in terms of learning speed, than the methods based only on the gradient descent technique. In fact, second order methods are among the fastest learning algorithms. Some of the most relevant examples of this type of methods are the quasi-Newton, Levenberg-Marquardt (Hagan and Menhaj, 1994; Levenberg, 1944; Marquardt, 1963) and the conjugate gradient algorithms (Beale, 1972). Quasi-Newton methods use a local quadratic approximation of the error function, like the Newton's method, but they employ an approximation of the inverse of the hessian matrix to update the weights, thus getting a lowest computational cost. The two most common updating procedures are the Davidson-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Dennis and Schnabel, 1983). The Levenberg-Marquardt method combines, in the same weight updating rule, both the gradient and the Gauss-Newton approximation of the hessian of the error function. The influence of each term is determined by an adaptive parameter, which is automatically updated. Regarding the conjugate gradient methods, they use, at each iteration of the algorithm, different search directions in a way that the component of the gradient is parallel to the previous search direction. Several algorithms based on

conjugate directions were proposed such as the Fletcher-Reeves (Fletcher and Reeves, 1964; Hagan et al., 1996), Polak-Ribiére (Fletcher and Reeves, 1964; Hagan et al., 1996), Powell-Beale (Powell, 1977) and scaled conjugate gradient algorithms (Moller, 1993). Also, based on these previous approaches, several new algorithms have been developed, like those of Chella et al. (1993) and Wilamowski et al. (2001). Nevertheless, second-order methods are not practicable for large neural networks trained in batch mode, although some attempts to reduce their computational cost or to obtain stochastic versions have appeared (LeCun et al., 1998; Schraudolph, 2002).

- *Adaptive step size*: In the standard backpropagation method the learning rate, which determines the magnitude of the changes in the weights for each iteration of the algorithm, is fixed at the beginning of the learning process. Several heuristic methods for the dynamical adaptation of the learning rate have been developed (Hush and Salas, 1988; Jacobs, 1988; Vogl et al., 1988). Other interesting algorithm is the superSAB, proposed by Tollenaere (Tollenaere, 1990). This method is an adaptive acceleration strategy for error backpropagation learning that converges faster than the gradient descent with optimal step size value, reducing the sensitivity to parameter values. Moreover, in (Weir, 1991) a method for the self-determination of this parameter has also been presented. More recently, in Orr and Leen (1996), an algorithm for fast stochastic gradient descent, which uses a nonlinear adaptive momentum scheme to optimize the slow convergence rate was proposed. Also, in Almeida et al. (1999), a new method for step size adaptation in stochastic gradient optimization was presented. This method uses independent step sizes for all parameters and adapts them employing the available derivatives estimates in the gradient optimization procedure. Additionally, a new online algorithm for local learning rate adaptation was proposed (Schraudolph, 2002).

- *Appropriate weights initialization*: The starting point of the algorithm, determined by the initial set of weights, also influences the method convergence speed. Thus, several solutions for the appropriate initialization of weights have been proposed. Nguyen and Widrow assign each hidden processing element an approximate portion of the range of the desired response (Nguyen and Widrow, 1990), and Drago and Ridella use the statistically controlled activation weight initialization, which aims to prevent neurons from saturation during the adaptation process by estimating the maximum value that the weights should take initially (Drago and Ridella, 1992). Also, in (Ridella et al., 1997), an analytical technique, to initialize the weights of a multilayer perceptron with vector quantization (VQ) prototypes given the equivalence between circular backpropagation networks and VQ classifiers, has been proposed.

- *Rescaling of variables*: The error signal involves the derivative of the neural function, which is multiplied in each layer. Therefore, the elements of the Jacobian matrix can differ greatly in magnitude for different layers. To solve this problem Rigler et al. (1991) have proposed a rescaling of these elements.

On the other hand, sensitivity analysis is a very useful technique for deriving how and how much the solution to a given problem depends on data (see, for example, Castillo et al., 1997, 1999, 2000). However, in this paper we show that sensitivity formulas can also be used for learning, and a novel supervised learning algorithm for two-layer feedforward neural networks that presents a high convergence speed is proposed. This algorithm, the Sensitivity-Based Linear Learning Method

(SBLLM), is based on the use of the sensitivities of each layer's parameters with respect to its inputs and outputs, and also on the use of independent systems of linear equations for each layer, to obtain the optimal values of its parameters. In addition, this algorithm gives the sensitivities of the sum of squared errors with respect to the input and output data.

The paper is structured as follows. In Section 2 a method for learning one layer neural networks that consists of solving a system of linear equations is presented, and formulas for the sensitivities of the sum of squared errors with respect to the input and output data are derived. In Section 3 the SBLLM method, which uses the previous linear method to learn the parameters of two-layer neural networks and the sensitivities of the total sum of squared errors with respect to the intermediate output layer values, which are modified using a standard gradient formula until convergence, is presented. In Section 4 the proposed method is illustrated by its application to several practical problems, and also it is compared with some other fast learning methods. In Section 5 the SBLLM method is presented as an initialization tool to be used with other learning methods. In Section 6 these results are discussed and some future work lines are presented. Finally, in Section 7 some conclusions and recommendations are given.

2. One-Layer Neural Networks

Consider the one-layer network in Figure 1. The set of equations relating inputs and outputs is given by

$$y_{js} = f_j\left(\sum_{i=0}^{I} w_{ji} x_{is}\right);\ \ j = 1,2,\ldots,J;\ \ s = 1,2,\ldots,S,$$

where I is the number of inputs, J the number of outputs, $x_{0s} = 1$, w_{ji} are the weights associated with neuron j and S is the number of data points.

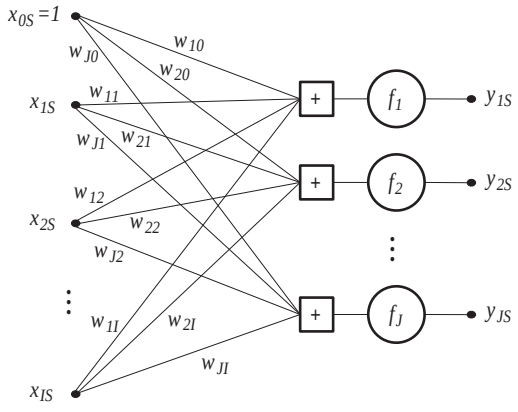


Figure 1: One-layer feedforward neural network.

To learn the weights w_{ji}, the following sum of squared errors between the real and the desired output of the networks is usually minimized:

$$P = \sum_{s=1}^{S}\sum_{j=1}^{J}\delta_{js}^2 = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(y_{js} - f_j\left(\sum_{i=0}^{I} w_{ji}x_{is}\right)\right)^2.$$

Assuming that the nonlinear activation functions, f_j, are invertible (as it is the case for the most commonly employed functions), alternatively, one can minimize the sum of squared errors before the nonlinear activation functions (Castillo et al., 2002), that is,

$$Q = \sum_{s=1}^{S}\sum_{j=1}^{J}\varepsilon_{js}^2 \quad = \quad \sum_{s=1}^{S}\sum_{j=1}^{J}\left(\sum_{i=0}^{I} w_{ji}x_{is} - f_j^{-1}(y_{js})\right)^2, \tag{1}$$

which leads to the system of equations:

$$\frac{\partial Q}{\partial w_{jp}} = 2\sum_{s=1}^{S}\left(\sum_{i=0}^{I} w_{ji}x_{is} - f_j^{-1}(y_{js})\right)x_{ps} \quad = \quad 0; \quad p = 0,1,\ldots,I; \quad \forall j,$$

that is,

$$\sum_{i=0}^{I} w_{ji}\sum_{s=1}^{S} x_{is}x_{ps} \quad = \quad \sum_{s=1}^{S} f_j^{-1}(y_{js})x_{ps}; \quad p = 0,1,\ldots,I; \quad \forall j$$

or

$$\sum_{i=0}^{I} A_{pi}w_{ji} \quad = \quad b_{pj}; \quad p = 0,1,\ldots,I; \quad \forall j, \tag{2}$$

where

$$\begin{aligned} A_{pi} &= \sum_{s=1}^{S} x_{is}x_{ps}; \quad p = 0,1,\ldots,I; \quad \forall i \\ b_{pj} &= \sum_{s=1}^{S} f_j^{-1}(y_{js})x_{ps}; \quad p = 0,1,\ldots,I; \quad \forall j. \end{aligned}$$

Moreover, for the neural network shown in Figure 1, the sensitivities (see Castillo et al., 2001, 2004, 2006) of the new cost function, Q, with respect to the output and input data can be obtained as:

$$\frac{\partial Q}{\partial y_{pq}} \quad = \quad -\frac{2\left(\sum_{i=0}^{I} w_{pi}x_{iq} - f_p^{-1}(y_{pq})\right)}{f_p'(y_{pq})}; \quad \forall p,q \tag{3}$$

$$\frac{\partial Q}{\partial x_{pq}} \quad = \quad 2\sum_{j=1}^{J}\left(\sum_{i=0}^{I} w_{ji}x_{iq} - f_j^{-1}(y_{jq})\right)w_{jp}; \quad \forall p,q. \tag{4}$$

3. The Proposed Sensitivity-Based Linear Learning Method

The learning method and the sensitivity formulas given in the previous section can be used to develop a new learning method for two-layer feedforward neural networks, as it is described below.

Consider the two-layer feedforward neural network in Figure 2 where I is the number of inputs, J the number of outputs, K the number of hidden units, $x_{0s}=1$, $z_{0s}=1$, S the number of data samples and the superscripts (1) and (2) are used to refer to the first and second layer, respectively. This network can be considered to be composed of two one-layer neural networks. Therefore, assuming that the intermediate layer outputs $\mathbf{z}$ are known, using equation (1), a new cost function for this network is defined as

$$\begin{aligned} Q(\mathbf{z}) &= Q^{(1)}(\mathbf{z})+Q^{(2)}(\mathbf{z})= \\ &= \sum_{s=1}^{S}\left[\sum_{k=1}^{K}\left(\sum_{i=0}^{I}w_{ki}^{(1)}x_{is}-f_k^{(1)^{-1}}(z_{ks})\right)^2+\sum_{j=1}^{J}\left(\sum_{k=0}^{K}w_{jk}^{(2)}z_{ks}-f_j^{(2)^{-1}}(y_{js})\right)^2\right]. \end{aligned}$$

Thus, using the outputs z_{ks} we can learn, for each layer independently, the weights $w_{ki}^{(1)}$ and $w_{jk}^{(2)}$ by solving the corresponding linear system of equations (2). After that, the sensitivities (see equations (3) and (4)) with respect to z_{ks} are calculated as:

$$\begin{aligned} \frac{\partial Q}{\partial z_{ks}} &= \frac{\partial Q^{(1)}}{\partial z_{ks}}+\frac{\partial Q^{(2)}}{\partial z_{ks}}= \\ &= -\frac{2\left(\sum_{i=0}^{I}w_{ki}^{(1)}x_{is}-f_k^{(1)^{-1}}(z_{ks})\right)}{f_k^{\prime(1)}(z_{ks})}+2\sum_{j=1}^{J}\left(\sum_{r=0}^{K}w_{jr}^{(2)}z_{rs}-f_j^{(2)^{-1}}(y_{js})\right)w_{jk}^{(2)} \end{aligned}$$

with $k=1,\ldots,K$, as $z_{0s}=1,\forall s$.

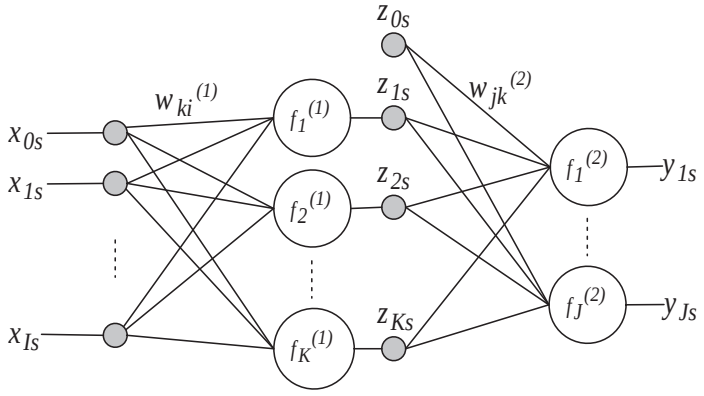


Figure 2: Two-layer feedforward neural network.

Next, the values of the intermediate outputs $\mathbf{z}$ are modified using the Taylor series approximation:

$$Q(\mathbf{z}+\Delta\mathbf{z})=Q(\mathbf{z})+\sum_{k=1}^{K}\sum_{s=1}^{S}\frac{\partial Q(\mathbf{z})}{\partial z_{ks}}\Delta z_{ks}\approx 0,$$

which leads to the following increments

$$\Delta \mathbf{z} = -\rho \frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q, \tag{5}$$

where ρ is a relaxation factor or step size.

The proposed method is summarized in the following algorithm.

Algorithm SBLLM

Input. The data set (input, x_{is}, and desired data, y_{js}), two threshold errors (ε and ε') to control convergence, and a step size ρ .

Output. The weights of the two layers and the sensitivities of the sum of squared errors with respect to input and output data.

Step 0: Initialization. Assign to the outputs of the intermediate layer the output associated with some random weights $\mathbf{w}^{(1)}(0)$ plus a small random error, that is:

$$z_{ks} = f_k^{(1)} \left(\sum_{i=0}^{I} w_{ki}^{(1)}(0) x_{is} \right) + \varepsilon_{ks}; \quad \varepsilon_{ks} \sim U(-\eta, \eta); k = 1, \ldots, K,$$

where η is a small number, and initialize $Q_{previous}$ and $MSE_{previous}$ to some large number, where MSE measures the error between the obtained and the desired output.

Step 1: Subproblem solution. Learn the weights of layers 1 and 2 and the associated sensitivities solving the corresponding systems of equations, that is,

$$\begin{aligned} \sum_{i=0}^{I} A_{pi}^{(1)} w_{ki}^{(1)} &= b_{pk}^{(1)} \\ \sum_{k=0}^{K} A_{qk}^{(2)} w_{jk}^{(2)} &= b_{qj}^{(2)}, \end{aligned}$$

where $A_{pi}^{(1)} = \sum_{s=1}^{S} x_{is} x_{ps}; \quad b_{pk}^{(1)} = \sum_{s=1}^{S} f_k^{(1)^{-1}}(z_{ks}) x_{ps}; \quad p = 0, 1, \ldots, I; \quad k = 1, 2, \ldots, K$

and $A_{qk}^{(2)} = \sum_{s=1}^{S} z_{ks} z_{qs}; \quad b_{qj}^{(2)} = \sum_{s=1}^{S} f_j^{(2)^{-1}}(y_{js}) z_{qs}; \quad q = 0, 1, \ldots, K; \quad \forall j.$

Step 2: Evaluate the sum of squared errors. Evaluate Q using

$$\begin{aligned} Q(\mathbf{z}) &= Q^{(1)}(\mathbf{z}) + Q^{(2)}(\mathbf{z}) \\ &= \sum_{s=1}^{S} \left[\sum_{k=1}^{K} \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks}) \right)^2 + \sum_{j=1}^{J} \left(\sum_{k=0}^{K} w_{jk}^{(2)} z_{ks} - f_j^{(2)^{-1}}(y_{js}) \right)^2 \right] \end{aligned}$$

and evaluate also the MSE.

Step 3: Convergence checking. If $|Q - Q_{previous}| < \varepsilon$ or $|MSE_{previous} - MSE| < \varepsilon'$ stop and return the weights and the sensitivities. Otherwise, continue with Step 4.

Step 4: Check improvement of Q. If $Q > Q_{previous}$ reduce the value of ρ, that is, $\rho = \rho/2$, and return to the previous position, that is, restore the weights, $\mathbf{z} = \mathbf{z}_{previous}$, $Q = Q_{previous}$ and go to Step 5. Otherwise, store the values of Q and $\mathbf{z}$, that is, $Q_{previous} = Q$, $MSE_{previous} = MSE$ and $\mathbf{z}_{previous} = \mathbf{z}$ and obtain the sensitivities using:

$$\frac{\partial Q}{\partial z_{ks}} = -\frac{2\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks})\right)}{f_k^{'(1)}(z_{ks})} + 2\sum_{j=1}^{J}\left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - f_j^{(2)^{-1}}(y_{js})\right) w_{jk}^{(2)}; k = 1,\ldots,K.$$

Step 5: Update intermediate outputs. Using the Taylor series approximation in equation (5), update the intermediate outputs as

$$\mathbf{z} = \mathbf{z} - \rho \frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q$$

and go to Step 1.

The complexity of this method is determined by the complexity of Step 1 which solves a linear system of equations for each network's layer. Several efficient methods can be used to solve this kind of systems with a complexity of $O(n^2)$, where n is the number of unknowns. Therefore, the resulting complexity of the proposed learning method is also $O(n^2)$, being n the number of weights of the network.

4. Examples of Applications of the SBLLM to Train Neural Networks

In this section the proposed method, SBLLM,[1] is illustrated by its application to five system identification problems. Two of them are small/medium size problems (Dow-Jones and Leuven competition time series), while the other three used large data sets and networks (Lorenz time series, and the MNIST and UCI Forest databases). Also, in order to check the performance of the SBLLM, it was compared with five of the most popular learning methods. Three of these methods are the gradient descent (GD), the gradient descent with adaptive momentum and step sizes (GDX), and the stochastic gradient descent (SGD), whose complexity is $O(n)$. The other methods are the scaled conjugated gradient (SCG), with complexity of $O(n^2)$, and the Levenberg-Marquardt (LM) (complexity of $O(n^3)$). All experiments were carried out in MATLAB® running on a Compaq HPC 320 with an Alpha EV68 1 GHz processor and 4GB of memory. For each experiment all the learning methods shared the following conditions:

- The network topology and neural functions. In all cases, the logistic function was used for hidden neurons, while for output neurons the linear function was used for regression problems and the logistic function was used for classification problems. It is important to remark that the aim here is not to investigate the optimal topology, but to check the performance of the algorithms in both small and large networks.

1. MATLAB® demo code available at http://www.dc.fi.udc.es/lidia/downloads/SBLLM.

- Initial step size equal to 0.05, except for the stochastic gradient descent. In this last case, we used a step size in the interval $[0.005, 0.2]$. These step sizes were tuned in order to obtain good results.
- The input data set was normalized (mean = 0 and standard deviation = 1).
- Several simulations were performed using for each one a different set of initial weights. This initial set was the same for all the algorithms (except for the SBLLM), and was obtained by the Nguyen-Widrow (Nguyen and Widrow, 1990) initialization method.
- Finally, statistical tests were performed in order to check whether the differences in accuracy and speed were significant among the different training algorithms. Specifically, first the non-parametric Kruskal-Wallis test (Hollander and Wolfe, 1973) was applied to check the hypothesis that all mean performances are equal. When this hypothesis is rejected, a multiple comparison test of means based on the Tukey's honestly significant difference criterion (Hsu, 1996) was applied to know which pairs of means are different. In all cases, a significance level of 0.05 was used.

4.1 Dow-Jones Time Series

The first data set is the time series corresponding to the Dow-Jones index values for years 1994-1996 (Ley, 1996). The goal of the network in this case is to predict the index for a given day based on the index of five previous days. For this data set a 5-7-1 topology (5 inputs, 7 hidden neurons and 1 output neuron) was used. Also, 900 samples were employed for the learning process. In order to obtain the MSE curves during the learning process 100 simulations of 3000 iterations each, were done.

Figure 3(a) shows, for each method, the mean error curve calculated over the 100 simulations. Also, in Figure 3(b) the box-whisker plots are shown for the 100 MSEs obtained by each method at the end of the training. In this graphic the box corresponds to the interquartile range, the bar inside the box represents the median, the whiskers extend to the farthest points that are not outliers, and outliers are represented by the plus sign.

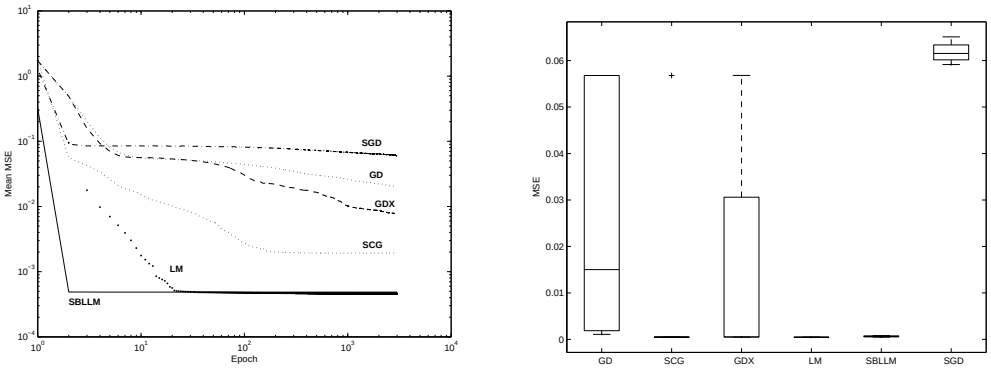
Also, different measures were calculated and collected in Table 4.1. These measures are:

- M_1: Mean and standard deviation of the minimum MSEs obtained by each method over the 100 simulations.
- M_2: Mean epoch and corresponding standard deviation in which each of the other methods reaches the minimum MSE obtained by the SBLLM.
- M_3: MSE and standard deviation for each of the other methods at the epoch in which the SBLLM gets its minimum.

In this case, the best mean MSE is achieved by the LM method (see M_1 in Table 4.1). Also, applying the multiple comparison test, it was found that the difference between this mean and those from the others methods was statistically significant.

Finally, the mean CPU times and the corresponding standard deviations for each of the methods are shown in Table 4.1. In this table, the variables $tepoch_{mean}$ and $tepoch_{std}$ are the mean and standard deviation CPU time (in seconds) per epoch, respectively, while the variables $ttotal_{mean}$ and

$ttotal_{std}$ correspond to the mean and standard deviation of the time needed to reach the minimum MSE. The *Ratio* column contains the relation between the $ttotal_{mean}$ of each algorithm and the fastest one. Again, the multiple comparison test applied over the $ttotal_{mean}$ revealed that the speed of the fastest method, that is the SBLLM, was only comparable to that of the SCG.



(a) Mean error curves over 100 simulations

(b) Boxplot of the 100 MSE values obtained at the end of the training

Figure 3: Results of the learning process for the Dow-Jones data.

	M_1	M_2	M_3
SBLLM	$4.866\times10^{-4}\pm2.252\times10^{-6}$	2.08 ± 0.394	$4.866\times10^{-4}\pm2.252\times10^{-6}$
LM	$4.601\times10^{-4}\pm1.369\times10^{-5}$	20 ± 11.5	$9.099\times10^{-2}\pm2.190\times10^{-1}$
SCG	$1.928\times10^{-3}\pm8.959\times10^{-3}$	354 ± 150 $^{(*1)}$	$5.517\times10^{-2}\pm6.034\times10^{-3}$
GDX	$7.747\times10^{-3}\pm1.802\times10^{-2}$	$2,180\pm635$ $^{(*2)}$	$4.717\times10^{-1}\pm5.962\times10^{-1}$
GD	$2.020\times10^{-2}\pm2.369\times10^{-2}$	>3000	$5.517\times10^{-2}\pm5.950\times10^{-1}$
SGD	$5.995\times10^{-2}\pm1.360\times10^{-3}$	>3000	$9.001\times10^{-2}\pm1.356\times10^{-2}$

$(*1)$ 4% of the curves did not get the minimum of SBLLM

$(*2)$ 99.5% of the curves did not get the minimum of SBLLM

Table 1: Comparative measures for the Dow-Jones data.

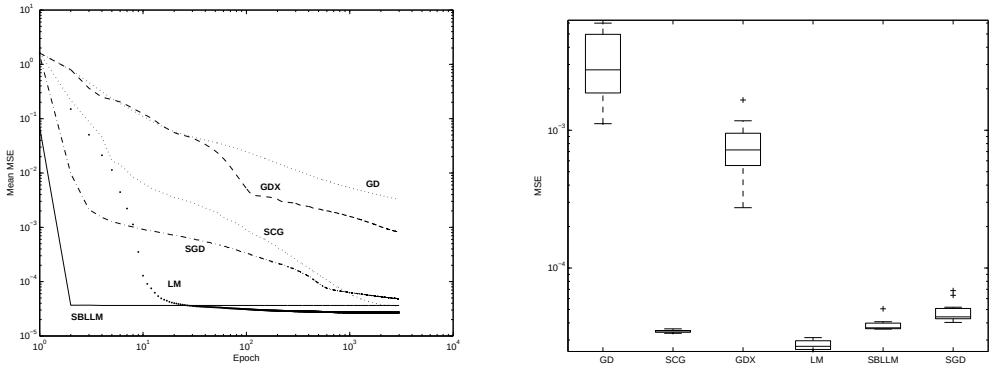
	$tepoch_{mean}$	$tepoch_{std}$	$ttotal_{mean}$	$ttotal_{std}$	*Ratio*
GD	0.0077	9.883×10^{-5}	23.125	0.297	223.4
GDX	0.0078	1.548×10^{-4}	22.764	3.548	219.9
SBLLM	0.0089	1.600×10^{-3}	0.104	0.115	1
SCG	0.0165	2.800×10^{-3}	15.461	8.595	149.4
LM	0.0395	3.460×10^{-2}	115.619	106.068	1,117.1
SGD	0.2521	1.814×10^{-3}	756.570	5.445	7,274.7

Table 2: CPU time comparison for the Dow-Jones data.

4.2 K.U. Leuven Competition Data

The K.U. Leuven time series prediction competition data (Suykens and Vandewalle, 1998) were generated from a computer simulated 5-scroll attractor, resulting from a generalized Chua's circuit which is a paradigm for chaos. 1800 data points of this time series were used for training. The aim of the neural network is to predict the current sample using only 4 previous data points. Thus the training set is reduced to 1796 input patterns corresponding to the number of 4-samples sliding windows over the initial training set. For this problem a 4-8-1 topology was used. As for the previous experiment 100 simulations of 3000 iterations each were carried out. Results are shown in Figure 4, and Tables 4.2 and 4.2.

In this case, the best mean MSE is achieved by the LM method (see M_1 in Table 4.2). However, the multiple comparison test did not show any significant difference with respect to the means of the SCG and the SBLLM. Regarding the $ttotal_{mean}$, the multiple comparison test showed that the speed of the fastest method, that is the SBLLM, was only comparable to those of the GDX and the GD.



(a) Mean error curves over 100 simulations

(b) Boxplot of the 100 MSE values obtained at the end of the training

Figure 4: Results of the learning process for the Leuven competition data.

	M_1	M_2	M_3
SBLLM	$3.639 \times 10^{-5} \pm 2.098 \times 10^{-7}$	2.2 ± 0.471	$3.639 \times 10^{-5} \pm 2.098 \times 10^{-7}$
LM	$2.7064 \times 10^{-5} \pm 2.439 \times 10^{-6}$	23.5 ± 16.3	$1.323 \times 10^{-1} \pm 3.143 \times 10^{-1}$
SCG	$3.517 \times 10^{-5} \pm 9.549 \times 10^{-7}$	2160 ± 445 $(*)$	$1.949 \times 10^{-1} \pm 2.083 \times 10^{-1}$
GDX	$8.121 \times 10^{-4} \pm 4.504 \times 10^{-4}$	> 3000	$7.190 \times 10^{-1} \pm 6.651 \times 10^{-1}$
GD	$3.280 \times 10^{-3} \pm 1.698 \times 10^{-3}$	> 3000	$1.949 \times 10^{-1} \pm 6.621 \times 10^{-1}$
SGD	$4.748 \times 10^{-5} \pm 9.397 \times 10^{-6}$	> 3000	$8.458 \times 10^{-3} \pm 5.292 \times 10^{-3}$

$(*)$ 9.8% of the curves did not get the minimum of SBLLM

Table 3: Comparative measures for the Leuven competition data.

	$tepoch_{mean}$	$tepoch_{std}$	$ttotal_{mean}$	$ttotal_{std}$	*Ratio*
GDX	0.0114	3.256×10^{-4}	34.309	0.977	710.3
GD	0.0117	3.208×10^{-4}	35.018	0.963	725
SBLLM	0.0173	2.362×10^{-3}	0.048	0.017	1
SCG	0.0238	6.271×10^{-4}	69.571	5.022	1440.4
LM	0.0669	5.440×10^{-2}	196.816	164.539	4074.9
SGD	0.5083	2.982×10^{-3}	1,525.34	8.949	31,777.9

Table 4: CPU time comparison for the Leuven competition data.

4.3 Lorenz Time Series

A Lorenz system (Lorenz, 1963) is described by the solution of three simultaneous differential equations:

$$dx/dt = -\sigma x + \sigma y$$

$$dy/dt = -xz + rx - y$$

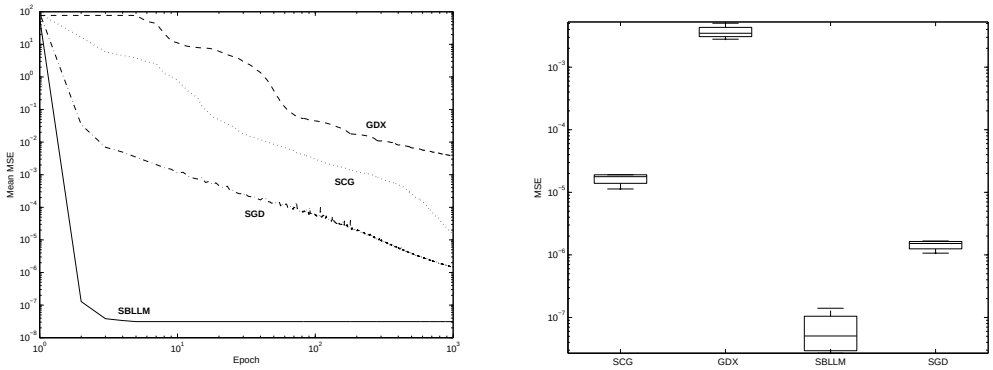
$$dz/dt = xy - bz,$$

where σ, r and b are constants. For this work, we employed $\sigma = 10$, $r = 28$, and $b = 8/3$, for which the system presents a chaotic dynamics. The goal of the network is to predict the current sample based on the four previous samples. For this data set a 8-100-1 topology was used. Also, 150000 samples were employed for the learning process. In this case, and due to the large size of both the data set and the neural networks, the conditions of the experiments were the following:

- The number of simulations, which were carried out to obtain the MSE curves during the learning process was reduced to 30, of 1000 iterations each.

- Neither the GD nor the LM methods were used. The results of the GD will not be presented because the method performed poorly, and the LM is impractical in these cases as it is highly computationally demanding (LeCun et al., 1998).

Results are shown in Figure 5, and Tables 4.3 and 4.3. In this case, the SBLLM was the best both in mean MSE and total CPU time, confirmed by the multiple comparison test.



(a) Mean error curves over 30 simulations

(b) Boxplot of the 30 MSE values obtained at the end of the training

Figure 5: Results of the learning process for the Lorenz data.

	M_1	M_2	M_3
SBLLM	$3.118 \times 10^{-8} \pm 2.151 \times 10^{-8}$	2.47 ± 0.776	$3.118 \times 10^{-8} \pm 2.151 \times 10^{-8}$
SGD	$1.426 \times 10^{-6} \pm 2.710 \times 10^{-7}$	> 1000	$2.512 \times 10^{-2} \pm 1.345 \times 10^{-2}$
SCG	$1.545 \times 10^{-5} \pm 3.922 \times 10^{-6}$	> 1000	$1.286 \times 10^{1} \pm 5.538$
GDX	$3.774 \times 10^{-3} \pm 8.409 \times 10^{-4}$	> 1000	$7.722 \times 10^{1} \pm 1.960 \times 10^{1}$

Table 5: Comparative measures for the Lorenz data.

	$tepoch_{mean}$	$tepoch_{std}$	$ttotal_{mean}$	$ttotal_{std}$	$Ratio$
GDX	9.75	0.04	9,750.92	42.65	849.7
SCG	21.01	0.63	21,111.2	633.62	1830.9
SGD	56.56	0.39	56,611.9	395.15	986.6
SBLLM	22.55	0.40	57.38	20.57	1

Table 6: CPU time comparison for the Lorenz data.

4.4 MNIST Data Set

The MNIST database, available at http://yann.lecun.com/exdb/mnist/, contains grey level images of handwritten digits of 28×28 pixels. It is a real classification problem whose goal is to determine the written number which is always an integer in the range between 0 and 9. This database is originally divided into a training set of 60,000 examples, and a test set of 10,000 examples. Further we extracted 10,000 samples from the training set to be used as a validation set.

For this data set we used 784 inputs neurons fed by the 28×28 pixels of each input pattern, and one output neuron per class. Specifically a 784-800-10 topology was used. In this case, and due to the large size of both the data set and the neural network, the conditions of the experiments were the following:

- The number of simulations, which were carried out to obtain the classification error was reduced to 20.

- Allowing a maximum of 200 iterations per simulation, the early stopping criteria using the validation set was employed to halt learning.

- The LM method was not used, since it is impractical in these cases as it is highly computationally demanding (LeCun et al., 1998).

- Concerning the other batch methods only the SCG was used since it is the one that clearly obtains the best results and convergence speed in the previous experiments.

Results are shown in Tables 4.4 and 4.4. As mentioned, the training process of the three methods were halted using the stop learning criteria. In this case, as can be observed the SGD achieved the best mean test accuracy, confirmed by the multiple comparison test. Besides, in all simulations the SBLLM always stops in iteration 75 achieving a worse accuracy than the SGD but employing a total time lesser than the other methods. In order to check if this result could be improved we did some other experiments allowing the SBLLM to run as long as the Stochastic Gradient Descent (SGD). However, results were not improved. Therefore, the presented tables show the most favourable situation for each algorithm.

Regarding the total CPU time, again the fastest method is the SBLLM, with a $ttotal_{mean}$ significantly different from the other two methods.

	$Train_{mean \pm std}$	$Validation_{mean \pm std}$	$Test_{mean \pm std}$
SGD	99.93 ± 0.04	97.87 ± 0.09	97.70 ± 0.08
SCG	78.12 ± 22.46	77.21 ± 21.97	77.03 ± 22.05
SBLLM	85.73 ± 0.03	86.52 ± 0.15	86.08 ± 0.26

Table 7: Classification accuracy for the MNIST data.

	$tepoch_{mean \pm std}$	$ttotal_{mean \pm std}$	$iterations_{mean \pm std}$	$Ratio$
SGD	$1,209.86 \pm 6.70$	$87,607.1 \pm 485.41$	72.4 ± 16.99	2.73
SCG	310.64 ± 2.93	$61,311.4 \pm 1,537$	197.4 ± 5.81	1.92
SBLLM	422.82 ± 1.28	$32,134.4 \pm 97$	75 ± 0	1

Table 8: CPU time comparison for the MNIST data.

4.5 Forest

The Forest CoverType database, on-line at http://kdd.ics.uci.edu/databases/covertype/covertype.html, contains data describing the wilderness areas and soil types for 30×30 meter cells obtained from US Forest Service Region 2 Resource Information System data. It is also a real classification problem whose goal is to determine the forest cover type from 54 input variables. Originally, the problem consider 7 cover classes, although in this case we have employed the 2-class version of the problem that consist of distinguishing the most frequent class from the other six (Collobert et al., 2003). This database contains 500,000 examples from which we built a training set of 101,241 examples, a validation set of 10,123 and a test set of 50,620 examples. These sets preserve the same proportion of samples for each of the seven classes as in the original data set.

For this data set a 54-500-2 topology was used. Regarding the number of simulations, stopping criteria and learning methods, the conditions were the same as those of the MNIST experiment described in the previous section.

As in the previous section, the most favourable results for each algorithm are shown in Tables 4.5 and 4.5. In this data set, the SGD achieved the best mean test accuracy, confirmed by the multiple comparison test. Regarding the total CPU time, the fastest method is the SBLLM, with a $ttotal_{mean}$ significantly different from the other two methods. It is important to remark that although in this case the SBLLM is the fastest method, this is due to the stop of the learning process in an early stage. This does not allow the SBLLM to achieve a good accuracy, as it is shown in Table 4.5. These results confirm that, for classification problems the SGD seems to be better in error than the SBLLM, which is similar in error but faster than the SCG.

	$Train_{mean \pm std}$	$Validation_{mean \pm std}$	$Test_{mean \pm std}$
SGD	89.60 ± 0.92	88.21 ± 0.69	88.22 ± 0.56
SCG	79.03 ± 1.29	78.69 ± 1.15	79.08 ± 1.16
SBLLM	79.87 ± 1.05	79.65 ± 0.22	79.92 ± 0.15

Table 9: Classification accuracy for the forest cover type data.

	$tepoch_{mean \pm std}$	$ttotal_{mean \pm std}$	$iterations_{mean \pm std}$	$Ratio$
SGD	106.95 ± 1.92	$15,210.63 \pm 273.34$	142.2 ± 55.36	109.14
SCG	100.20 ± 3.40	$17,903.66 \pm 3063.92$	178.60 ± 29.66	92.58
SBLLM	139.37 ± 0.72	139.37 ± 0.72	1 ± 0	1

Table 10: CPU time comparison for the forest cover type data.

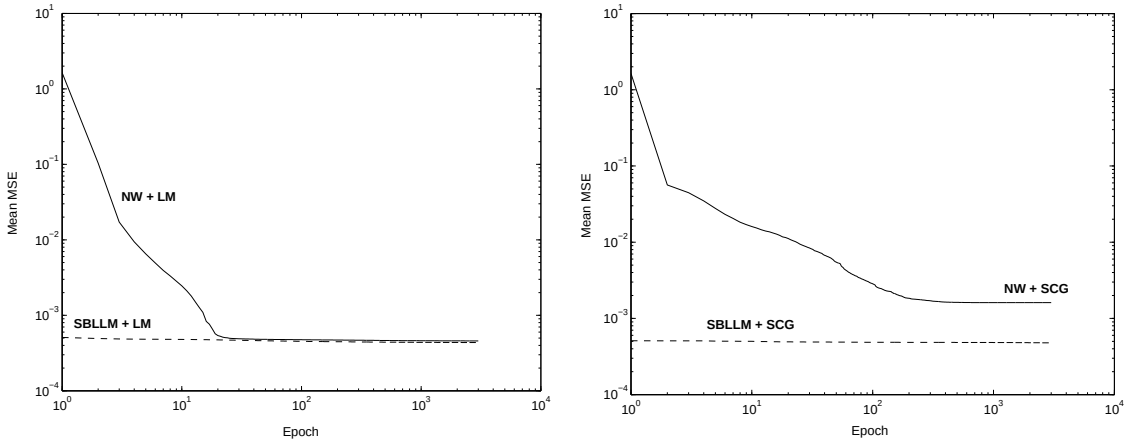
5. The SBLLM as Initialization Method

As has been shown in the previous section, the SBLLM achieves a small error rate in very few epochs. Although this error rate is very small and, in general, better than the errors obtained by other learning methods, as can be seen in Figures 3(a), 4(a) and 5(a), once the SBLLM gets this point the variation in the MSE in further epochs is not significant. For this reason, an interesting alternative is to combine the SBLLM with other learning methods.

In this section, the results of the SBLLM used as an initialization method instead of as a learning algorithm are presented. Thus, several experiments were accomplished using the SBLLM only to get the initial values of the weights of the neural network. Afterwards, the LM and SCG were used as learning methods from these initial values. The experiments were carried out using the Dow-Jones, Leuven and Lorenz time series. For these three data sets the experimental conditions were the same as those described in Section 4.

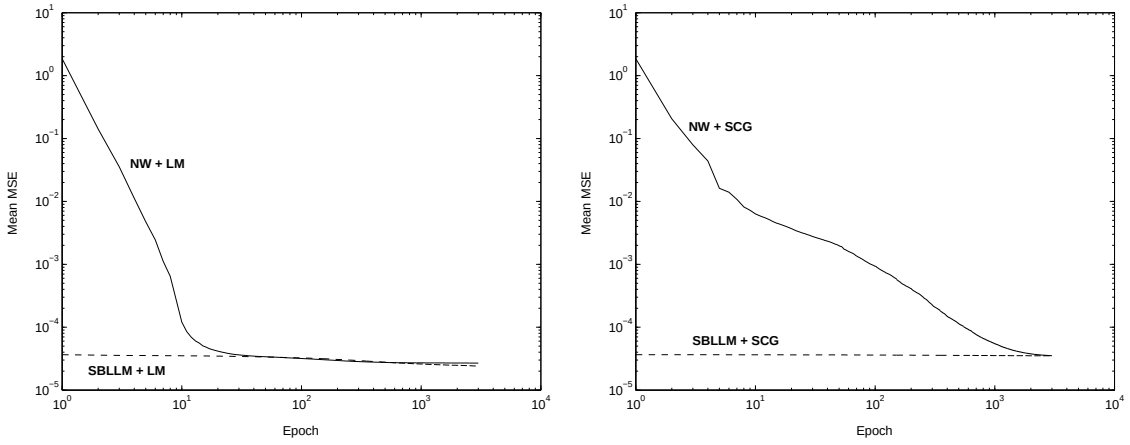
For every experiment 100 simulations were done of 3000 iterations each. In all cases, the SBLLM performed at most three iterations to get the initial weights. Moreover, in order to accomplish a comparative study, the obtained results were confronted with the ones achieved by the same learning methods (LM and SCG) but using the Nguyen-Widrow (NW) initialization method (Nguyen and Widrow, 1990), one of the most popular, to obtain the initial weights.

Figures 6(a), 7(a) and 8(a) show the corresponding mean curves (over the 100 simulations) of the learning process using the SBLLM and the NW as initialization methods and the LM as the learning algorithm. Figures 6(b), 7(b) and 8(b) show the same mean curves of the learning process using this time the SCG as learning algorithm.



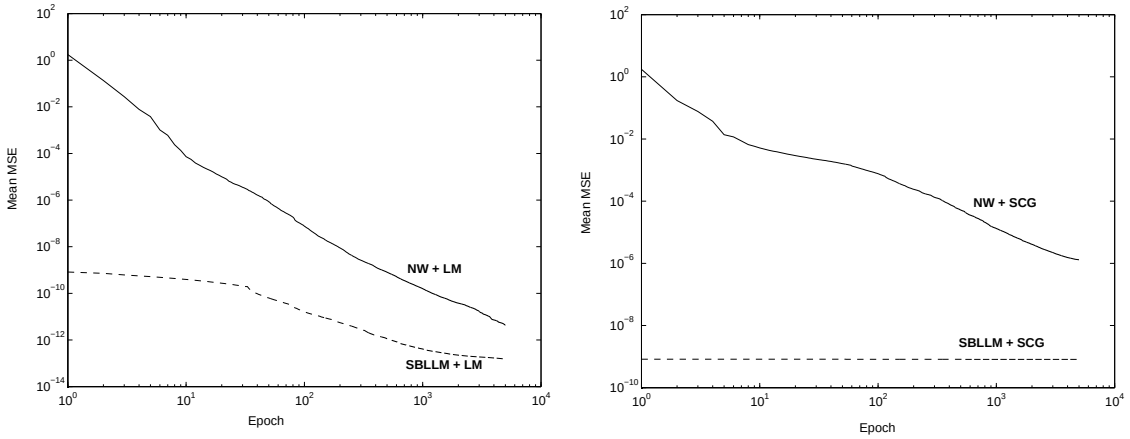
(a) Mean error curves for the LM method (b) Mean error curves for the SCG method

Figure 6: Mean error curves over 100 simulations for the Dow-Jones time series using the SBLLM and the NW as initialization methods.



(a) Mean error curves for the LM method (b) Mean error curves for the SCG method

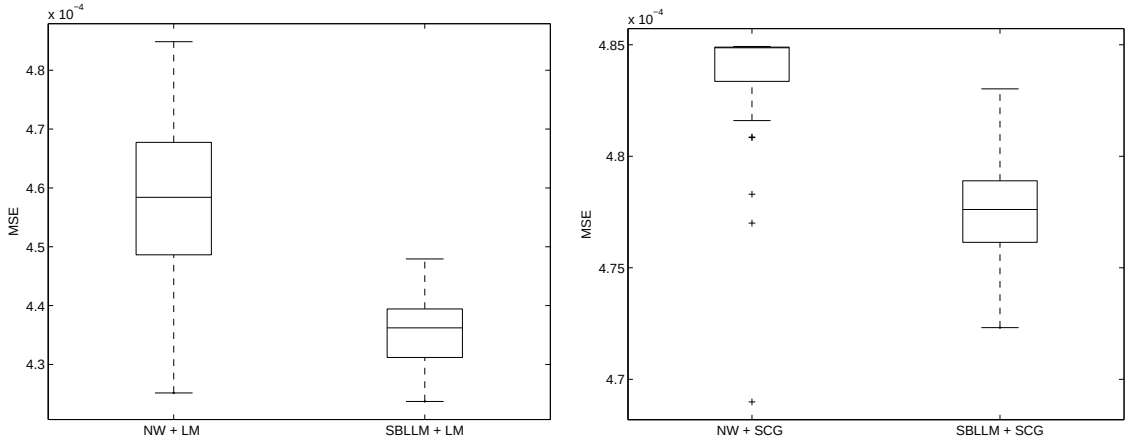
Figure 7: Mean error curves over 100 simulations for the Leuven competition time series using the SBLLM and the NW as initialization methods.



(a) Mean error curves for the LM method (b) Mean error curves for the SCG method

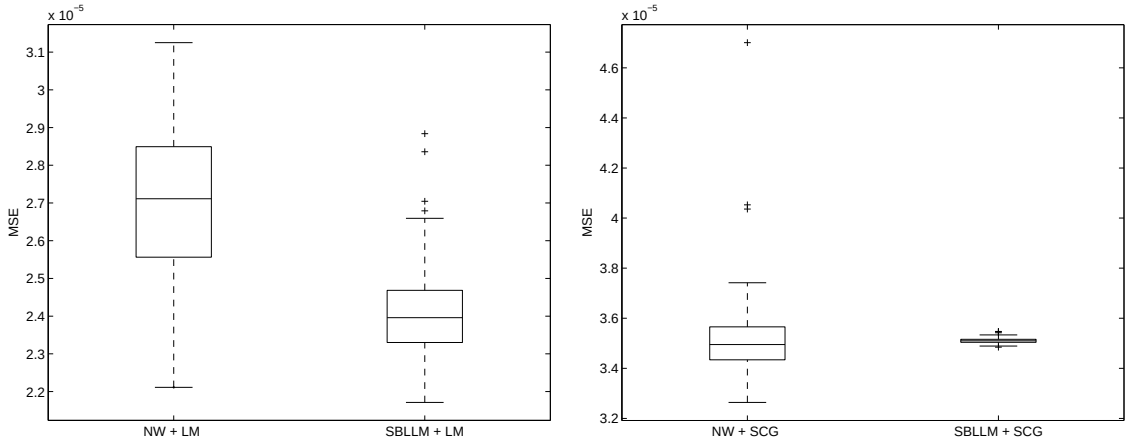
Figure 8: Mean error curves over 100 simulations for the Lorenz time series using the SBLLM and the NW as initialization methods.

Figures 9(a), 10(a) and 11(a) contain the boxplots of the methods in the last epoch of training (3000) using the SBLLM and NW as initialization methods and the LM as the learning algorithm. Figures 9(b), 10(b) and 11(b) depict the same boxplots when using the SCG as the learning algorithm.



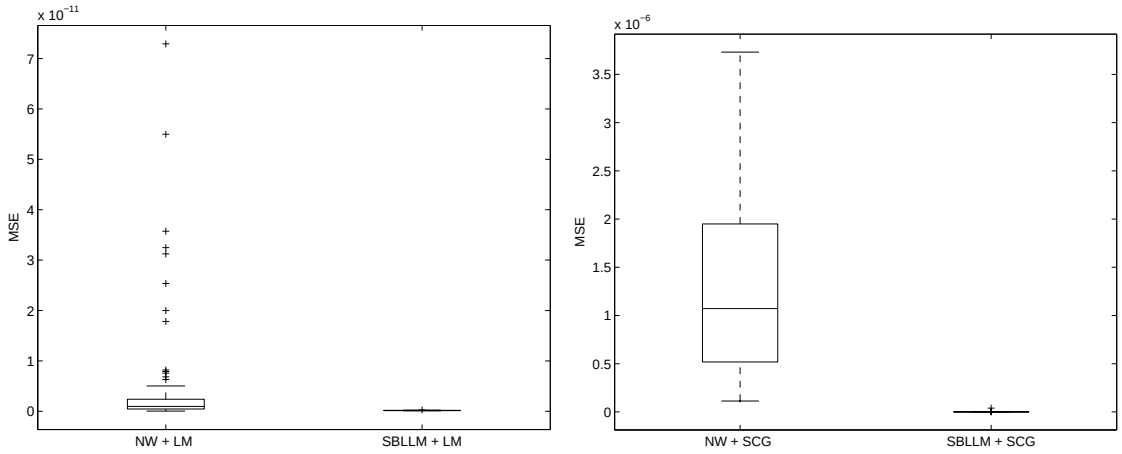
(a) Boxplot for the LM method (b) Boxplot for the SCG method

Figure 9: Boxplot of the 100 MSE values at the last epoch of training for the Dow-Jones time series using the SBLLM and NW as initialization methods.



(a) Boxplot for the LM method (b) Boxplot for the SCG method

Figure 10: Boxplot of the 100 MSE values at the last epoch of training for the Leuven competition time series using the SBLLM and NW as initialization methods.



(a) Boxplot for the LM method (b) Boxplot for the SCG method

Figure 11: Boxplot of the 100 MSE values at the last epoch of training for the Lorenz time series using the SBLLM and NW as initialization methods.

6. Discussion

Regarding the behavior of the SBLLM as a *learning algorithm*, and from the experiments made and the results presented in Section 4, there are three main features of the SBLLM that stand out:

1. *High speed in reaching the minimum error*. For the first three problems (Dow-Jones, Leuven and Lorenz time series), this feature can be observed in Figures 3(a), 4(a) and 5(a), and the measure M_2 in Tables 4.1, 4.2 and 4.3, where it can be seen that in all cases the SBLLM

obtains its minimum MSE (minMSE) just before the first 4 iterations and also sooner than the rest of the algorithms. Moreover, and generally speaking, measure M_3 reflects that the SBLLM gets its minimum in an epoch for which the other algorithms are far from similar MSE values.

If we take into account the CPU times in Tables 4.1, 4.2, 4.3, 4.4 and 4.5 we can see that, as expected, the CPU time per epoch of the SBLLM is similar to that of the SCG (both of $O(n^2)$), and when we consider the total CPU time per simulation the SBLLM is, in the worst case, more than 150 times faster than the fastest algorithm for the regression examples and approximately 2 times faster for the classification examples. It is also important to remark that despite of the advantages of the LM method, it could not be applied in the experiments that involved large data sets and neural networks as it is impractical for such cases (LeCun et al., 1998).

2. *A good performance*. From Figures 3(a), 4(a) and 5(a), and the measure M_1 in Tables 4.1, 4.2 and 4.3, it can be deduced that not only the SBLLM stabilizes soon, but also the minMSE that it reaches is quite good and comparable to that obtained by the second order methods. On the other hand, the GD and the GDX learning methods never succeeded in attaining this minMSE before the maximum number of epochs, as reflected in measure M_2 (Tables 4.1, 4.2 and 4.3). Finally, the SCG algorithm presents an intermediate behavior, although seldom achieves the levels of performance of the LM and SBLLM.

   Regarding the classification problems, from Tables 4.4 and 4.5, it can be deduced that the SBLLM performs similar or better than the other batch method, that is the SCG, while the stochastic method (SGD) is the best algorithm for this kind of problems (LeCun et al., 1998).

   Although the ability of the proposed algorithm to get a minimum in just very few epochs is usually an advantage, it can also be noticed that once it achieves this minimum (local or global) it gets stuck in this point. This causes that, sometimes like in the classification examples included, the algorithm is not able to obtain a high accuracy. This behavior could be explained by the initialization method and the updating rule for the step size employed.

3. *Homogeneous behavior*. This feature comprehends several aspects:

   - The SBLLM learning curve stabilizes soon, as can be observed in Figures 3(a), 4(a) and 5(a).
   - Regarding the minimum MSE reached at the end of the learning process, it can be observed from Figures 3(b), 4(b) and 5(b) that, in any case, the GD and GDX algorithms present a wider dispersion, given even place to the appearance of outliers. On the other hand, the SGD, SCG, LM and SBLLM algorithms tend to always obtain nearer values of MSE. This fact is also reflected by the standard deviations of measure M_1.
   - The SBLLM behaves homogeneously not only if we consider just the end of the learning process, as commented, but also during the whole process, in such a way that very similar learning curves where obtained for all iterations of the first three experiments. This is, in a certain way, reflected in the standard deviation of measure M_3 which corresponds to the MSE value taken at some intermediate point of the learning process.

- Finally, from Tables 4.4 and 4.5 it can be observed that for these last two experiments (MNIST and Forest databases) this homogeneus behaviour stands for the SGD and the SBLLM, while the SCG presents a wider dispersion in its classification errors.

With respect to the use of the SBLLM as *initialization method*, as it can be observed in Figures 6, 7 and 8, the SBLLM combined with the LM or the SCG achieves a faster convergence speed than the same methods using the NW as initialization method. Also, the SBLLM obtains a very good initial point, and thus a very low MSE in a few epochs of training. Moreover, in this case, most of the times the final MSE achieved is smaller than the one obtained using the NW initialization method. This result is better illustrated in the boxplots of the corresponding time series where it can be observed, in addition, that the final MSE obtained with NW presents a higher variability than that achieved by the SBLLM, that is, the SBLLM helps the learning algorithms to obtain a more homogeneous MSE at the end of the training process. Thus, experiments confirm the utility of the SBLLM as an initialization method, which effect is to speed up the convergence.

7. Conclusions and Future Work

The main conclusions that can be drawn from this paper are:

1. The sensitivities of the sum of squared errors with respect to the outputs of the intermediate layer allow an efficient and fast gradient method to be applied.

2. Over the experiments made the SBLLM offers an interesting combination of speed, reliability and simplicity.

3. Regarding the employed regression problems only second order methods, and more specifically the LM, seem to obtain similar results although at a higher computational cost.

4. With respect to the employed classification problems, the SBLLM performs similar or better than the other batch method, although requiring less computational time. Besides, the stochastic gradient (SGD) is the one that obtains the lowest classification error. This result is in accordance with that obtained by other authors (LeCun et al., 1998) that recommend this method for large data sets and networks in classification tasks.

5. The SBLLM used as an initialization method significantly improves the performance of a learning algorithm.

Finally, there are some aspects of the proposed algorithm that need an in depth study, and will be addressed in a future work:

1. A more appropriate method to set the initial values of the outputs $\mathbf{z}$ of hidden neurons (step 0 of the proposed algorithm).

2. A more efficient updating rule for the step size ρ, like a method based on a line search (hard or soft).

3. An adaptation of the algorithm to improve its performance on classification problems, specifically for large data sets.

Acknowledgments

We would like to acknowledge support for this project from the Spanish Ministry of Science and Technology (Projects DPI2002-04172-C04-02 and TIC2003-00600, this last partially supported by FEDER funds) and the Xunta de Galicia (project PGIDT04PXIC10502PN). Also, we thank the Supercomputing Center of Galicia (CESGA) for allowing us the use of the high performance computing servers.

References

L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov. Parameter adaptation in stochastic optimization. In D. Saad, editor, *On-line Learning in Neural Networks*, chapter 6, pages 111–134. Cambridge University Press, 1999.

R. Battiti. First and second order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166, 1992.

E. M. L. Beale. A derivation of conjugate gradients. In F. A. Lootsma, editor, *Numerical methods for nonlinear optimization*, pages 39–43. Academic Press, London, 1972.

F. Biegler-König and F. Bärmann. A learning algorithm for multilayered neural networks based on linear least-squares problems. *Neural Networks*, 6:127–131, 1993.

W. L. Buntine and A. S. Weigend. Computing second derivatives in feed-forward networks: A review. *IEEE Transactions on Neural Networks*, 5(3):480–488, 1993.

E. Castillo, J. M. Gutiérrez, and A. Hadi. Sensitivity analysis in discrete bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics*, 26(7):412–423, 1997.

E. Castillo, A. Cobo, J. M. Gutiérrez, and R. E. Pruneda. Working with differential, functional and difference equations using functional networks. *Applied Mathematical Modelling*, 23(2):89–107, 1999.

E. Castillo, A. Cobo, J. M. Gutiérrez, and R. E. Pruneda. Functional networks. a new neural network based methodology. *Computer-Aided Civil and Infrastructure Engineering*, 15(2):90–106, 2000.

E. Castillo, A. Conejo, P. Pedregal, R. García, and N. Alguacil. *Building and Solving Mathematical Programming Models in Engineering and Science*. John Wiley & Sons Inc., New York., 2001.

E. Castillo, O. Fontenla-Romero, A. Alonso Betanzos, and B. Guijarro-Berdiñas. A global optimum approach for one-layer neural networks. *Neural Computation*, 14(6):1429–1449, 2002.

E. Castillo, A. S. Hadi, A. Conejo, and A. Fernández-Canteli. A general method for local sensitivity analysis with application to regression models and other optimization problems. *Technometrics*, 46(4):430–445, 2004.

E. Castillo, C. Castillo A. Conejo and, R. Mínguez, and D. Ortigosa. A perturbation approach to sensitivity analysis in nonlinear programming. *Journal of Optimization Theory and Applications*, 128(1):49–74, 2006.

A. Chella, A. Gentile, F. Sorbello, and A. Tarantino. Supervised learning for feed-forward neural networks: a new minimax approach for fast convergence. *Proceedings of the IEEE International Conference on Neural Networks*, 1:605 – 609, 1993.

V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.

R. Collobert, Y. Bengio, and S. Bengio. Scaling large learning problems with hard parallel mixtures. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):349–365, 2003.

J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

G. P. Drago and S. Ridella. Statistically controlled activation weight initialization (SCAWI). *IEEE Transactions on Neural Networks*, 3:899–905, 1992.

R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7 (149–154), 1964.

O. Fontenla-Romero, D. Erdogmus, J.C. Principe, A. Alonso-Betanzos, and E. Castillo. Linear least-squares based methods for neural networks learning. *Lecture Notes in Computer Science*, 2714(84–91), 2003.

M. T. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.

M. T. Hagan, H. B. Demuth, and M. H. Beale. *Neural Network Design*. PWS Publishing, Boston, MA, 1996.

M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*. John Wiley & Sons, 1973.

J. C. Hsu. *Multiple Comparisons. Theory and Methods*. Chapman&Hall/CRC, Boca Raton, FL, 1996.

D. R. Hush and J. M. Salas. Improving the learning rate of back-propagation with the gradient reuse algorithm. *Proceedings of the IEEE Conference of Neural Networks*, 1:441–447, 1988.

B. C. Ihm and D. J. Park. Acceleration of learning speed in neural networks by reducing weight oscillations. *Proceedings of the International Joint Conference on Neural Networks*, 3:1729–1732, 1999.

R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1 (4):295–308, 1988.

Y. LeCun, I. Kanter, and S.A. Solla. Second order properties of error surfaces: Learning time and generalization. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Neural Information Processing Systems*, volume 3, pages 918–924, San Mateo, CA, 1991. Morgan Kaufmann.

Y. LeCun, L. Bottou, G.B. Orr, and K.-R. Müller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the trade*, number 1524 in LNCS. Springer-Verlag, 1998.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quaterly Journal of Applied Mathematics*, 2(2):164–168, 1944.

E. Ley. On the peculiar distribution of the U.S. stock indeces' first digits. *The American Statistician*, 50(4):311–314, 1996.

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

D. W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441, 1963.

M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of the International Joint Conference on Neural Networks*, 3:21–26, 1990.

G. B. Orr and T. K. Leen. Using curvature information for fast stochastic search. In M.I. Jordan, M.C. Mozer, and T. Petsche, editors, *Neural Information Processing Systems*, volume 9, pages 606–612, Cambridge, 1996. MIT Press.

D. B. Parker. Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order hebbian learning. *Proceedings of the IEEE Conference on Neural Networks*, 2:593–600, 1987.

S. Pethel, C. Bowden, and M. Scalora. Characterization of optical instabilities and chaos using MLP training algorithms. *SPIE Chaos Opt.*, 2039:129–140, 1993.

M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254, 1977.

S. Ridella, S. Rovetta, and R. Zunino. Circular backpropagation networks for classification. *IEEE Transactions on Neural Networks*, 8(1):84–97, January 1997.

A. K. Rigler, J. M. Irvine, and T. P. Vogl. Rescaling of variables in back propagation learning. *Neural Networks*, 4:225–229, 1991.

D. E. Rumelhart, G. E. Hinton, and R. J. Willian. Learning representations of back-propagation errors. *Nature*, 323:533–536, 1986.

N. N. Schraudolph. Fast curvature matrix-vector products for second order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

A. Sperduti and S. Antonina. Speed up learning and network optimization with extended back propagation. *Neural Networks*, 6:365–383, 1993.

J. A. K. Suykens and J. Vandewalle, editors. *Nonlinear Modeling: advanced black-box techniques*. Kluwer Academic Publishers Boston, 1998.

T. Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3(561–573), 1990.

T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of back-propagation method. *Biological Cybernetics*, 59:257–263, 1988.

M. K. Weir. A method for self-determination of adaptive learning rates in back propagation. *Neural Networks*, 4:371–379, 1991.

B. M. Wilamowski, S. Iplikci, O. Kaynak, and M. O. Efe. An algorithm for fast convergence in training neural networks. *Proceedings of the International Joint Conference on Neural Networks*, 2:1778–1782, 2001.

J. Y. F. Yam, T. W. S Chow, and C. T Leung. A new method in determining the initial weights of feedforward neural networks. *Neurocomputing*, 16(1):23–32, 1997.

Journal of Machine Learning Research 7 (2006) 1183–1204 Submitted 12/05; Revised 3/06; Published 7/06

Computational and Theoretical Analysis of Null Space and Orthogonal Linear Discriminant Analysis

Jieping Ye JIEPING.YE@ASU.EDU
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287, USA

Tao Xiong TXIONG@ECE.UMN.EDU
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA

Editor: David Madigan

Abstract

Dimensionality reduction is an important pre-processing step in many applications. Linear discriminant analysis (LDA) is a classical statistical approach for supervised dimensionality reduction. It aims to maximize the ratio of the between-class distance to the within-class distance, thus maximizing the class discrimination. It has been used widely in many applications. However, the classical LDA formulation requires the nonsingularity of the scatter matrices involved. For undersampled problems, where the data dimensionality is much larger than the sample size, all scatter matrices are singular and classical LDA fails. Many extensions, including null space LDA (NLDA) and orthogonal LDA (OLDA), have been proposed in the past to overcome this problem. NLDA aims to maximize the between-class distance in the null space of the within-class scatter matrix, while OLDA computes a set of orthogonal discriminant vectors via the simultaneous diagonalization of the scatter matrices. They have been applied successfully in various applications.

In this paper, we present a computational and theoretical analysis of NLDA and OLDA. Our main result shows that under a mild condition which holds in many applications involving high-dimensional data, NLDA is equivalent to OLDA. We have performed extensive experiments on various types of data and results are consistent with our theoretical analysis. We further apply the regularization to OLDA. The algorithm is called regularized OLDA (or ROLDA for short). An efficient algorithm is presented to estimate the regularization value in ROLDA. A comparative study on classification shows that ROLDA is very competitive with OLDA. This confirms the effectiveness of the regularization in ROLDA.

Keywords: linear discriminant analysis, dimensionality reduction, null space, orthogonal matrix, regularization

1. Introduction

Dimensionality reduction is important in many applications of data mining, machine learning, and bioinformatics, due to the so-called *curse of dimensionality* (Bellmanna, 1961; Duda et al., 2000; Fukunaga, 1990; Hastie et al., 2001). Many methods have been proposed for dimensionality reduction, such as principal component analysis (PCA) (Jolliffe, 1986) and linear discriminant analysis

(LDA) (Fukunaga, 1990). LDA aims to find the optimal discriminant vectors (transformation) by maximizing the ratio of the between-class distance to the within-class distance, thus achieving the maximum class discrimination. It has been applied successfully in many applications including information retrieval (Berry et al., 1995; Deerwester et al., 1990), face recognition (Belhumeour et al., 1997; Swets and Weng, 1996; Turk and Pentland, 1991), and microarray gene expression data analysis (Dudoit et al., 2002). However, classical LDA requires the so-called *total scatter matrix* to be nonsingular. In many applications such as those mentioned above, all scatter matrices in question can be singular since the data points are from a very high-dimensional space and in general the sample size does not exceed this dimensionality. This is known as the *singularity* or *undersampled problem* (Krzanowski et al., 1995).

In recent years, many approaches have been proposed to deal with such high-dimensional, undersampled problem, including null space LDA (NLDA) (Chen et al., 2000; Huang et al., 2002), orthogonal LDA (OLDA) (Ye, 2005), uncorrelated LDA (ULDA) (Ye et al., 2004a; Ye, 2005), subspace LDA (Belhumeour et al., 1997; Swets and Weng, 1996), regularized LDA (Friedman, 1989), and pseudo-inverse LDA (Raudys and Duin, 1998; Skurichina and Duin, 1996). Null space LDA computes the discriminant vectors in the null space of the within-class scatter matrix. Uncorrelated LDA and orthogonal LDA are among a family of algorithms for generalized discriminant analysis proposed in (Ye, 2005). The features in ULDA are uncorrelated, while the discriminant vectors in OLDA are orthogonal to each other. Subspace LDA (or PCA+LDA) applies an intermediate dimensionality reduction stage such as PCA to reduce the dimensionality of the original data before classical LDA is applied. Regularized LDA uses a scaled multiple of the identity matrix to make the scatter matrix nonsingular. Pseudo-inverse LDA employs the pseudo-inverse to overcome the singularity problem. More details on these methods, as well as their relationship, can be found in (Ye, 2005). In this paper, we present a detailed computational and theoretical analysis of null space LDA and orthogonal LDA.

In (Chen et al., 2000), the null space LDA (NLDA) was proposed, where the between-class distance is maximized in the null space of the within-class scatter matrix. The singularity problem is thus implicitly avoided. Similar idea has been mentioned briefly in (Belhumeour et al., 1997). (Huang et al., 2002) improved the efficiency of the algorithm by first removing the null space of the total scatter matrix, based on the observation that the null space of the total scatter matrix is the intersection of the null space of the between-class scatter matrix and the null space of the within-class scatter matrix.

In orthogonal LDA (OLDA), a set of orthogonal discriminant vectors is computed, based on a generalized optimization criterion (Ye, 2005). The optimal transformation is computed through the simultaneous diagonalization of the scatter matrices, while the singularity problem is overcome implicitly. Discriminant analysis with orthogonal transformations has been studied in (Duchene and Leclerq, 1988; Foley and Sammon, 1975). By a close examination of the computations involved in OLDA, we can decompose the OLDA algorithm into three steps: first remove the null space of the total scatter matrix; followed by classical uncorrelated LDA (ULDA), a variant of classical LDA (details can be found in Section 2.1); and finally apply an orthogonalization step to the transformation.

Both the NLDA algorithm (Huang et al., 2002) and the OLDA algorithm (Ye, 2005) result in orthogonal transformations. However, they applied different schemes in deriving the optimal transformations. NLDA computes an orthogonal transformation in the null space of the within-class scatter matrix, while OLDA computes an orthogonal transformation through the simultaneous diagonaliza-

tion of the scatter matrices. Interestingly, we show in Section 5 that NLDA is equivalent to OLDA, under a mild condition C1,[1] which holds in many applications involving high-dimensional data (see Section 7). Based on the equivalence result, an improved algorithm for NLDA, called iNLDA, is presented, which further reduces the computational cost of the original NLDA algorithm.

We extend the OLDA algorithm by applying the regularization technique, which is commonly used to stabilize the sample covariance matrix estimation and improve the classification performance (Friedman, 1989). The algorithm is called regularized OLDA (or ROLDA for short). The key idea in ROLDA is to add a constant λ to the diagonal elements of the total scatter matrix. Here $\lambda > 0$ is known as the *regularization parameter*. Choosing an appropriate regularization value is a critical issue in ROLDA, as a large λ may significantly disturb the information in the scatter matrix, while a small λ may not be effective in improving the classification performance. Cross-validation is commonly used to estimate the optimal λ from a finite set of candidates. Selecting an optimal value for a parameter such as λ is called *model selection* (Hastie et al., 2001). The computational cost of model selection for ROLDA can be expensive, especially when the candidate set is large, since it requires expensive matrix computations for each λ. We show in Section 6 that the computations in ROLDA can be decomposed into two components: the first component involves matrices of high dimensionality but independent of λ, while the second component involves matrices of low dimensionality. When searching for the optimal λ from a set of candidates via cross-validation, we repeat the computations involved in the second component only, thus reducing the computational cost of model selection in ROLDA.

We have conducted experiments using 14 data sets from various data sources, including low-dimensional data from UCI Machine Learning Repository[2] and high-dimensional data such as text documents, face images, and gene expression data. (Details on these data sets can be found in Section 7.) We did a comparative study of NLDA, iNLDA, OLDA, ULDA, ROLDA, and Support Vector Machines (SVM) (Schökopf and Smola, 2002; Vapnik, 1998) in classification. Experimental results show that

- For all low-dimensional data sets, the null space of the within-class scatter matrix is empty, and both NLDA and iNLDA do not apply. However, OLDA is applicable and the reduced dimensionality of OLDA is in general $k-1$, where k is the number of classes. Condition C1 holds for most high-dimensional data sets (eight out of nine data sets). NLDA, iNLDA, and OLDA achieve the same classification performance, in all cases when condition C1 holds. For cases where condition C1 does not hold, OLDA outperforms NLDA and iNLDA, as OLDA has a larger number of reduced dimensions than NLDA and iNLDA. These empirical results are consistent with our theoretical analysis.

- iNLDA and NLDA achieve similar performance in all cases. OLDA is very competitive with ULDA. This confirms the effectiveness of the final orthogonalization step in OLDA. ROLDA achieves a better classification performance than OLDA, which shows the effectiveness of the regularization in ROLDA. Overall, ROLDA and SVM are very competitive with other methods in classification.

The rest of the paper is organized as follows. An overview of classical LDA and classical uncorrelated LDA is given in Section 2. NLDA and OLDA are discussed in Section 3 and Section 4,

1. Condition C1 requires that the rank of the total scatter matrix equals to the sum of the rank of the between-class scatter matrix and the rank of the within-class scatter matrix. More details will be given in Section 5.
2. http://www.ics.uci.edu/∼mlearn/MLRepository.html

Notation	Description	Notation	Description
A	data matrix	n	number of training data points
m	data dimensionality	ℓ	reduced dimensionality
k	number of classes	S_b	between-class scatter matrix
S_w	within-class scatter matrix	S_t	total scatter matrix
G	transformation matrix	S_i	covariance matrix of the i-th class
c_i	centroid of the i-th class	n_i	sample size of the i-th class
c	global centroid	K	number of neighbors in K-NN
t	rank of S_t	q	rank of S_b

Table 1: Notation.

respectively. The relationship between NLDA and OLDA is studied in Section 5. The ROLDA algorithm is presented in Section 6. Section 7 includes the experimental results. We conclude in Section 8.

For convenience, Table 1 lists the important notation used in the rest of this paper.

2. Classical Linear Discriminant Analysis

Given a data set consisting of n data points $\{a_j\}_{j=1}^n$ in $\mathbb{R}^m$, classical LDA computes a linear transformation $G \in \mathbb{R}^{m\times\ell}$ ($\ell < m$) that maps each a_j in the m-dimensional space to a vector $\hat{a}_j$ in the ℓ-dimensional space by $\hat{a}_j = G^T a_j$. Define three matrices H_w, H_b, and S_t as follows:

$$H_w = \frac{1}{\sqrt{n}}[(A_1 - c_1 e^T), \cdots, (A_k - c_k e^T)], \quad (1)$$

$$H_b = \frac{1}{\sqrt{n}}[\sqrt{n_1}(c_1 - c), \cdots, \sqrt{n_k}(c_k - c)], \quad (2)$$

$$H_t = \frac{1}{\sqrt{n}}(A - ce^T), \quad (3)$$

where $A = [a_1, \cdots, a_n]$ is the data matrix, A_i, c_i, S_i, and n_i are the data matrix, the centroid, the covariance matrix, and the sample size of the i-th class, respectively, c is the global centroid, k is the number of classes, and e is the vector of all ones. Then the *between-class scatter matrix* S_b, the *within-class scatter matrix* S_w, and the *total scatter matrix* S_t are defined as follows (Fukunaga, 1990):

$$S_w = H_w H_w^T,\ S_b = H_b H_b^T,\ \text{and}\ S_t = H_t H_t^T.$$

It follows from the definition (Ye, 2005) that trace(S_w) measures the within-class cohesion, trace(S_b) measures the between-class separation, and trace(S_t) measures the variance of the data set, where the trace of a square matrix is the summation of its diagonal entries (Golub and Van Loan, 1996). It is easy to verify that $S_t = S_b + S_w$. In the lower-dimensional space resulting from the linear transformation G, the scatter matrices become $S_w^L = G^T S_w G$, $S_b^L = G^T S_b G$, and $S_t^L = G^T S_t G$. An optimal transformation G would maximize trace(S_b^L) and minimize trace(S_w^L). Classical LDA

aims to compute the optimal G by solving the following optimization problem:

$$G = \arg \max_{G \in \mathbb{R}^{m \times \ell} : G^T S_w G = I_\ell} \text{trace} \left(\left(G^T S_w G \right)^{-1} G^T S_b G \right). \tag{4}$$

Other optimization criteria, including those based on the determinant could also be used instead (Duda et al., 2000; Fukunaga, 1990). The solution to the optimization problem in Eq. (4) is given by the eigenvectors of $S_w^{-1} S_b$ corresponding to the nonzero eigenvalues, provided that the within-class scatter matrix S_w is nonsingular (Fukunaga, 1990). The columns of G form the discriminant vectors of classical LDA. Since the rank of the between-class scatter matrix is bounded from above by $k-1$, there are at most $k-1$ discriminant vectors in classical LDA. Note that classical LDA does not handle singular scatter matrices, which limits its applicability to low-dimensional data. Several methods, including null space LDA and orthogonal LDA subspace LDA, were proposed in the past to deal with such singularity problem as discussed in Section 1.

2.1 Classical Uncorrelated LDA

Classical uncorrelated LDA (cULDA) is an extension of classical LDA. A key property of cULDA is that the features in the transformed space are uncorrelated, thus reducing the redundancy in the transformed space.

cULDA aims to find the optimal discriminant vectors that are S_t-orthogonal.[3] Specifically, suppose r vectors $\phi_1, \phi_2, \cdots, \phi_r$ are obtained, then the $(r+1)$-th vector ϕ_{r+1} is the one that maximizes the Fisher criterion function (Jin et al., 2001):

$$f(\phi) = \frac{\phi^T S_b \phi}{\phi^T S_w \phi}, \tag{5}$$

subject to the constraints: $\phi_{r+1}^T S_t \phi_i = 0$, for $i = 1, \cdots, r$.

The algorithm in (Jin et al., 2001) finds the discriminant vectors ϕ_i's successively by solving a sequence of generalized eigenvalue problems, which is expensive for large and high-dimensional data sets. However, it has been shown (Ye et al., 2004a) that the discriminant vectors of cULDA can be computed efficiently by solving the following optimization problem:

$$G = \arg \max_{G \in \mathbb{R}^{m \times \ell} : G^T S_t G = I_\ell} \text{trace} \left(\left(G^T S_w G \right)^{-1} G^T S_b G \right), \tag{6}$$

where $G = [\phi_1, \cdots, \phi_\ell]$, if there exist ℓ discriminant vectors in cULDA. Note that in Eq. (6), all discriminant vectors in G are computed simultaneously. The optimization problem above is a variant of the one in Eq. (4). The optimal G is given by the eigenvectors of $S_t^{-1} S_b$.

3. Null Space LDA

(Chen et al., 2000) proposed the null space LDA (NLDA) for dimensionality reduction, where the between-class distance is maximized in the null space of the within-class scatter matrix. The basic idea behind this algorithm is that the null space of S_w may contain significant discriminant information if the projection of S_b is not zero in that direction (Chen et al., 2000; Lu et al., 2003).

3. Two vectors x and y are S_t-orthogonal, if $x^T S_t y = 0$.

The singularity problem is thus overcome implicitly. The optimal transformation of NLDA can be computed by solving the following optimization problem:

$$G = \text{argmax}_{G^T S_w G = 0} \text{trace}(G^T S_b G). \tag{7}$$

The computation of the optimal G involves the computation of the null space of S_w, which may be large for high-dimensional data. Indeed, the dimensionality of the null space of S_w is at least $m+k-n$, where m is the data dimensionality, k is the number of classes, and n is the sample size. In (Chen et al., 2000), a pixel grouping method was used to extract geometric features and reduce the dimensionality of samples, and then NLDA was applied in the new feature space. (Huang et al., 2002) improved the efficiency of the algorithm in (Chen et al., 2000) by first removing the null space of the total scatter matrix S_t. It is based on the observation that the null space of S_t is the intersection of the null space of S_b and the null space of S_w, as $S_t = S_w + S_b$.

We can efficiently remove the null space of S_t as follows. Let $H_t = U\Sigma V^T$ be the Singular Value Decomposition (SVD) (Golub and Van Loan, 1996) of H_t, where H_t is defined in Eq. (3), U and V are orthogonal,

$$\Sigma = \begin{pmatrix} \Sigma_t & 0 \\ 0 & 0 \end{pmatrix},$$

$\Sigma_t \in \mathbb{R}^{t \times t}$ is diagonal with the diagonal entries sorted in the non-increasing order, and $t = \text{rank}(S_t)$. Then

$$S_t = H_t H_t^T = U\Sigma V^T V \Sigma^T U^T = U\Sigma\Sigma^T U^T = U \begin{pmatrix} \Sigma_t^2 & 0 \\ 0 & 0 \end{pmatrix} U^T. \tag{8}$$

Let $U = (U_1, U_2)$ be a partition of U with $U_1 \in \mathbb{R}^{m \times t}$ and $U_2 \in \mathbb{R}^{m \times (m-t)}$. Then the null space of S_t can be removed by projecting the data onto the subspace spanned by the columns of U_1. Let $\tilde{S}_b$, $\tilde{S}_w$, and $\tilde{S}_t$ be the scatter matrices after the removal of the null space of S_t. That is,

$$\tilde{S}_b = U_1^T S_b U_1, \ \tilde{S}_w = U_1^T S_w U_1, \text{ and } \tilde{S}_t = U_1^T S_t U_1.$$

Note that only U_1 is involved for the projection. We can thus apply the reduced SVD computation (Golub and Van Loan, 1996) on H_t with the time complexity of $O(mn^2)$, instead of $O(m^2 n)$. When the data dimensionality m is much larger than the sample size n, this leads to a big reduction in terms of the computational cost.

With the computed U_1, the optimal transformation of NLDA is given by $G = U_1 N$, where N is obtained by solving the following optimization problem:

$$N = \text{argmax}_{N^T \tilde{S}_w N = 0} \text{trace}(N^T \tilde{S}_b N). \tag{9}$$

That is, the columns of N lie in the null space of $\tilde{S}_w$, while maximizing $\text{trace}(N^T \tilde{S}_b N)$.

Let W be the matrix so that the columns of W span the null space of $\tilde{S}_w$. Then $N = WM$, for some matrix M, which is to be determined next. Since the constraint in Eq. (9) is satisfied with $N = WM$ for any M, the optimal M can be computed by maximizing

$$\text{trace}(M^T W^T \tilde{S}_b W M).$$

By imposing the orthogonality constraint on M (Huang et al., 2002), the optimal M is given by the eigenvectors of $W^T \tilde{S}_b W$ corresponding to the nonzero eigenvalues. With the computed U_1, W, and M above, the optimal transformation of NLDA is given by

$$G = U_1 W M.$$

Algorithm 1: NLDA (Null space LDA)

Input: data matrix A

Output: transformation matrix G

1. Form the matrix H_t as in Eq. (3);
2. Compute the reduced SVD of H_t as $H_t = U_1 \Sigma_t V_1^T$;
3. Form the matrices $\tilde{S}_b = U_1^T S_b U_1$ and $\tilde{S}_w = U_1^T S_w U_1$;
4. Compute the null space, W, of $\tilde{S}_w$, via the eigen-decomposition;
5. Construct the matrix M, consisting of the top eigenvectors of $W^T \tilde{S}_b W$;
6. $G \leftarrow U_1 W M$.

In (Huang et al., 2002), the matrix W is computed via the eigen-decomposition of $\tilde{S}_w$. More specifically, let

$$\tilde{S}_w = [W, \tilde{W}] \begin{pmatrix} 0 & 0 \\ 0 & \Delta_w \end{pmatrix} [W, \tilde{W}]^T$$

be its eigen-decomposition, where $[W, \tilde{W}]$ is orthogonal and Δ_w is diagonal with positive diagonal entries. Then W forms the null space of $\tilde{S}_w$. The pseudo-code for the NLDA algorithm is given in **Algorithm 1**.

4. Orthogonal LDA

Orthogonal LDA (OLDA) was proposed in (Ye, 2005) as an extension of classical LDA. The discriminant vectors in OLDA are orthogonal to each other. Furthermore, OLDA is applicable even when all scatter matrices are singular, thus overcoming the singularity problem. It has been applied successfully in many applications, including document classification, face recognition, and gene expression data classification. The optimal transformation in OLDA can be computed by solving the following optimization problem:

$$G = \text{argmax}_{G \in \mathbb{R}^{m \times \ell} : G^T G = I_\ell} \text{trace}\left((G^T S_t G)^+ G^T S_b G \right), \tag{10}$$

where M^+ denotes the pseudo-inverse of matrix M (Golub and Van Loan, 1996). The orthogonality condition is imposed in the constraint. The computation of the optimal transformation of OLDA is based on the simultaneous diagonalization of the three scatter matrices as follows (Ye, 2005).

From Eq. (8), U_2 lies in the null space of both S_b and S_w. Thus,

$$U^T S_b U = \begin{pmatrix} U_1^T S_b U_1 & 0 \\ 0 & 0 \end{pmatrix}, \quad U^T S_w U = \begin{pmatrix} U_1^T S_w U_1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{11}$$

Denote $B = \Sigma_t^{-1} U_1^T H_b$ and let $B = P \tilde{\Sigma} Q^T$ be the SVD of B, where P and Q are orthogonal and $\tilde{\Sigma}$ is diagonal. Define the matrix X as

$$X = U \begin{pmatrix} \Sigma_t^{-1} P & 0 \\ 0 & I_{m-t} \end{pmatrix}. \tag{12}$$

It can be shown (Ye, 2005) that X simultaneously diagonalizes S_b, S_w, and S_t. That is

$$X^T S_b X = D_b, \; X^T S_w X = D_w, \text{ and } X^T S_t X = D_t, \tag{13}$$

Algorithm 2: OLDA (Orthogonal LDA)
Input: data matrix A
Output: transformation matrix G
1. Compute U_1, Σ_t, and P;
2. $X_q \leftarrow U_1\Sigma_t^{-1}P_q$, where $q = \text{rank}(S_b)$;
3. Compute the QR decomposition of X_q as $X_q = QR$;
4. $G \leftarrow Q$.

where D_b, D_w, and D_t are diagonal with the diagonal entries in D_b sorted in the non-increasing order. The main result in (Ye, 2005) has shown that the optimal transformation of OLDA can be computed through the orthogonalization of the columns in X, as summarized in the following theorem:

Theorem 4.1 *Let X be the matrix defined in Eq. (12) and let X_q be the matrix consisting of the first q columns of X, where $q = rank(S_b)$. Let $X_q = QR$ be the QR-decomposition of X_q, where Q has orthonormal columns and R is upper triangular. Then $G = Q$ solves the optimization problem in Eq. (10).*

From Theorem 4.1, only the first q columns of X are used in computing the optimal G. From Eq. (12), the first q columns of X are given by

$$X_q = U_1\Sigma_t^{-1}P_q, \tag{14}$$

where P_q consists of the first q columns of the matrix P. We can observe that U_1 corresponds to the removal of the null space of S_t as in NLDA, while $\Sigma_t^{-1}P_q$ is the optimal transformation when classical ULDA is applied to the intermediate (dimensionality) reduced space by the projection of U_1. The OLDA algorithm can thus be decomposed into three steps: (1) Remove the null space of S_t; (2) Apply classical ULDA as an intermediate step, since the reduced total scatter is nonsingular; and (3) Apply an orthogonalization step to the transformation, which corresponds to the QR decomposition of X_q in Theorem 4.1. The pseudo-code for the OLDA algorithm is given in **Algorithm 2**.

Remark 1 *The ULDA algorithm in (Ye et al., 2004a; Ye, 2005) consists of steps 1 and 2 above, without the final orthogonalization step. Experimental results in Section 7 show that OLDA is competitive with ULDA. The rationale behind this may be that ULDA involves the minimum redundancy in the transformed space and is susceptible to overfitting; OLDA, on the other hand, removes the R matrix through the QR decomposition in the final orthogonalization step, which introduces the redundancy in the reduced space, but may be less susceptible to overfitting.*

5. Relationship Between NLDA and OLDA

Both the NLDA algorithm and the OLDA algorithm result in orthogonal transformations. Our empirical results show that they often lead to similar performance, especially for high-dimensional data. This implies there may exist an intrinsic relationship between these two algorithms. In this section, we take a closer look at the relationship between NLDA and OLDA. More specifically, we show that NLDA is equivalent to OLDA, under a mild condition

$$C1 : \text{rank}(S_t) = \text{rank}(S_b) + \text{rank}(S_w), \tag{15}$$

which holds in many applications involving high-dimensional data (see Section 7). It is easy to verify from the definition of the scatter matrices that $\text{rank}(S_t) \leq \text{rank}(S_b) + \text{rank}(S_w)$.

From Eqs. (8) and (11), the null space, U_2, of S_t can be removed, as follows:

$$\tilde{S}_t = U_1^T S_t U_1 = U_1^T S_b U_1 + U_1^T S_w U_1 = \tilde{S}_w + \tilde{S}_b \in \mathbb{R}^{t\times t}.$$

Since the null space of S_t is the intersection of the null space of S_b and the null space of S_w, the following equalities hold:

$$\text{rank}(\tilde{S}_t) = \text{rank}(S_t) = t, \quad \text{rank}(\tilde{S}_b) = \text{rank}(S_b), \text{ and } \text{rank}(\tilde{S}_w) = \text{rank}(S_w).$$

Thus condition C1 is equivalent to

$$\text{rank}(\tilde{S}_t) = \text{rank}(\tilde{S}_b) + \text{rank}(\tilde{S}_w).$$

The null space of $\tilde{S}_b$ and the null space of $\tilde{S}_w$ are critical in our analysis. The relationship between these two null spaces is studied in the following lemma.

Lemma 5.1 *Let $\tilde{S}_t$, $\tilde{S}_b$, and $\tilde{S}_w$ be defined as above and $t = \text{rank}(\tilde{S}_t)$. Let $\{w_1, \cdots, w_r\}$ forms an orthonormal basis for the null space of $\tilde{S}_w$, and let $\{b_1, \cdots, b_s\}$ forms an orthonormal basis for the null space of $\tilde{S}_b$. Then, $\{w_1, \cdots, w_r, b_1, \cdots, b_s\}$ are linearly independent.*

Proof Prove by contradiction. Assume there exist α_i's and β_j's, not all zeros, such that

$$\sum_{i=1}^{r} \alpha_i w_i + \sum_{j=1}^{s} \beta_j b_j = 0.$$

It follows that

$$0 = \left(\sum_{i=1}^{r} \alpha_i w_i + \sum_{j=1}^{s} \beta_j b_j\right)^T \tilde{S}_w \left(\sum_{i=1}^{r} \alpha_i w_i + \sum_{j=1}^{s} \beta_j b_j\right) = \left(\sum_{j=1}^{s} \beta_j b_j\right)^T \tilde{S}_w \left(\sum_{j=1}^{s} \beta_j b_j\right),$$

since w_i's lie in the null space of $\tilde{S}_w$. Hence,

$$\begin{aligned}\left(\sum_{j=1}^{s} \beta_j b_j\right)^T \tilde{S}_t \left(\sum_{j=1}^{s} \beta_j b_j\right) &= \left(\sum_{j=1}^{s} \beta_j b_j\right)^T \tilde{S}_w \left(\sum_{j=1}^{s} \beta_j b_j\right) + \left(\sum_{j=1}^{s} \beta_j b_j\right)^T \tilde{S}_b \left(\sum_{j=1}^{s} \beta_j b_j\right) \\ &= 0.\end{aligned}$$

Since $\tilde{S}_t$ is nonsingular, we have $\sum_{j=1}^{s} \beta_j b_j = 0$. Thus $\beta_j = 0$, for all j, since $\{b_1, \cdots, b_s\}$ forms an orthonormal basis for the null space of $\tilde{S}_b$.

Similarly, we have

$$0 = \left(\sum_{i=1}^{r} \alpha_i w_i + \sum_{j=1}^{s} \beta_j b_j\right)^T \tilde{S}_b \left(\sum_{i=1}^{r} \alpha_i w_i + \sum_{j=1}^{s} \beta_j b_j\right) = \left(\sum_{i=1}^{r} \alpha_i w_i\right)^T \tilde{S}_b \left(\sum_{i=1}^{r} \alpha_i w_i\right).$$

and

$$\begin{aligned}\left(\sum_{i=1}^{r} \alpha_i w_i\right)^T \tilde{S}_t \left(\sum_{i=1}^{r} \alpha_i w_i\right) &= \left(\sum_{i=1}^{r} \alpha_i w_i\right)^T \tilde{S}_w \left(\sum_{i=1}^{r} \alpha_i w_i\right) + \left(\sum_{i=1}^{r} \alpha_i w_i\right)^T \tilde{S}_b \left(\sum_{i=1}^{r} \alpha_i w_i\right) \\ &= 0.\end{aligned}$$

Hence $\sum_{i=1}^{r} \alpha_i w_i = 0$, and $\alpha_i = 0$, for all i, since $\{w_1, \cdots, w_r\}$ forms are orthonormal basis for the null space of $\tilde{S}_w$. This contradicts our assumption that not all of the α_i's and the β_j's are zero, Thus, $\{w_1, \cdots, w_r, b_1, \cdots, b_s\}$ are linearly independent. ■

Next, we show how to compute the optimal transformation of NLDA using these two null spaces. Recall that in NLDA, the null space of S_t may be removed first. In the following discussion, we work on the reduced scatter matrices $\tilde{S}_w$, $\tilde{S}_b$, and $\tilde{S}_t$ directly as in Lemma 5.1. The main result is summarized in the following theorem.

Theorem 5.1 *Let U_1, $\tilde{S}_t$, $\tilde{S}_b$, and $\tilde{S}_w$ be defined as above and $t = rank(\tilde{S}_t)$. Let $R = [W, B]$, where $W = [w_1, \cdots, w_r]$, $B = [b_1, \cdots, b_s]$, and $\{w_1, \cdots, w_r, b_1, \cdots, b_s\}$ are defined as in Lemma 5.1. Assume that condition C1: $rank(S_t) = rank(S_b) + rank(S_w)$ holds. Then $G = U_1 W M$ solves the optimization problem in Eq. (9), where the matrix M, consisting of the eigenvectors of $W^T \tilde{S}_b W$, is orthogonal.*

Proof From Lemma 5.1, $\{w_1, \cdots, w_r, b_1, \cdots, b_s\} \in \mathbb{R}^t$ is linearly independent. Condition C1 implies that $t = r + s$. Thus $\{w_1, \cdots, w_r, b_1, \cdots, b_s\}$ forms a basis for $\mathbb{R}^t$, that is, $R = [W, B]$ is nonsingular. It follows that

$$\begin{aligned} R^T \tilde{S}_t R &= R^T \tilde{S}_b R + R^T \tilde{S}_w R \\ &= \begin{pmatrix} W^T \tilde{S}_b W & W^T \tilde{S}_b B \\ B^T \tilde{S}_b W & B^T \tilde{S}_b B \end{pmatrix} + \begin{pmatrix} W^T \tilde{S}_w W & W^T \tilde{S}_w B \\ B^T \tilde{S}_w W & B^T \tilde{S}_w B \end{pmatrix} \\ &= \begin{pmatrix} W^T \tilde{S}_b W & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & B^T \tilde{S}_w B \end{pmatrix}. \end{aligned}$$

Since matrix $R^T \tilde{S}_t R$ has full rank, $W^T \tilde{S}_b W$, the projection of $\tilde{S}_b$ onto the null space of $\tilde{S}_w$, is nonsingular. Let $W^T \tilde{S}_b W = M \Delta_b M^T$ be the eigen-decomposition of $W^T \tilde{S}_b W$, where M is orthogonal and Δ_b is diagonal with positive diagonal entries (note that $W^T \tilde{S}_b W$ is positive definite). Then, from Section 3, the optimal transformation G of NLDA is given by $G = U_1 W M$. ■

Recall that the matrix M in NLDA is computed so that $\text{trace}(M^T W^T \tilde{S}_b W M)$ is maximized. Since $\text{trace}(QAQ^T) = \text{trace}(A)$ for any orthogonal Q, the solution in NLDA is invariant under an arbitrary orthogonal transformation. Thus $G = U_1 W$ is also a solution to NLDA, since M is orthogonal, as summarized in the following corollary.

Corollary 5.1 *Assume condition C1: $rank(S_t) = rank(S_b) + rank(S_w)$ holds. Let U_1 and W be defined as in Theorem 5.1. Then $G = U_1 W$ solves the optimization problem in Eq. (9). That is, $G = U_1 W$ is an optimal transformation of NLDA.*

Corollary 5.1 implies that when condition C1 holds, Step 5 in **Algorithm 1** may be removed, as well as the formation of $\tilde{S}_b$ in Step 3 and the multiplication of $U_1 W$ with M in Step 6. This improves the efficiency of the NLDA algorithm. The improved NLDA (iNLDA) algorithm is given in **Algorithm 3**. Note that it is recommended in (Liu et al., 2004) that the maximization of the between-class distance in Step 5 of **Algorithm 1** should be removed to avoid possible overfitting. However, Corollary 5.1 shows that under condition C1, the removal of Step 5 has no effect on the performance of the NLDA algorithm.

Next, we show the equivalence relationship between NLDA and OLDA, when condition C1 holds. The main result is summarized in the following theorem.

Algorithm 3: iNLDA (improved NLDA)
Input: data matrix A
Output: transformation matrix G
1. Form the matrix H_t as in Eq. (3);
2. Compute the reduced SVD of H_t as $H_t = U_1 \Sigma_t V_1^T$;
3. Construct the matrix $\tilde{S}_w = U_1^T S_w U_1$;
4. Compute the null space, W, of $\tilde{S}_w$, via the eigen-decomposition;
5. $G \leftarrow U_1 W$.

Theorem 5.2 *Assume that condition C1: $rank(S_t) = rank(S_b) + rank(S_w)$ holds. Let U_1 and W be defined as in Theorem 5.1. Then, $G = U_1 W$ solves the optimization problem in Eq. (10). That is, under the given assumption, OLDA and NLDA are equivalent.*

Proof Recall that the optimization involved in OLDA is

$$G = \text{argmax}_{G \in \mathbb{R}^{m \times \ell} : G^T G = I_\ell} \text{trace}\left((S_t^L)^+ S_b^L\right), \tag{16}$$

where $S_t^L = G^T S_t G$ and $S_b^L = G^T S_b G$. From Section 4, the maximum number, ℓ, of discriminant vectors is no larger than q, which is the rank of S_b. Recall that

$$q = \text{rank}(S_b) = \text{rank}(\tilde{S}_b) = \text{rank}(\tilde{S}_t) - \text{rank}(\tilde{S}_w) = r,$$

where r is the dimension of the null space of $\tilde{S}_w$.

Based on the property of the trace of matrices, we have

$$\text{trace}\left((S_t^L)^+ S_b^L\right) + \text{trace}\left((S_t^L)^+ S_w^L\right) = \text{trace}\left((S_t^L)^+ S_t^L\right) = \text{rank}(S_t^L) \leq q = r,$$

where the second equality follows since $\text{trace}(A^+ A) = \text{rank}(A)$ for any square matrix A, and the inequality follows since the rank of $S_t^L \in \mathbb{R}^{\ell \times \ell}$ is at most $\ell \leq q$.

It follows that $\text{trace}\left((S_t^L)^+ S_b^L\right) \leq r$, since $\text{trace}\left((S_t^L)^+ S_w^L\right)$, the trace of the product of two positive semi-definite matrices, is always nonnegative. Next, we show that the maximum is achieved, when $G = U_1 W$.

Recall that the dimension of the null space, W, of $\tilde{S}_w$ is r. That is, $W \in \mathbb{R}^{t \times r}$. It follows that $(U_1 W)^T S_t (U_1 W) \in \mathbb{R}^{r \times r}$, and $\text{rank}((U_1 W)^T S_t (U_1 W)) = r$. Furthermore,

$$(U_1 W)^T S_w (U_1 W) = W^T \tilde{S}_w W = 0,$$

as W forms the null space of $\tilde{S}_w$. It follows that,

$$\text{trace}\left(\left((U_1 W)^T S_t (U_1 W)\right)^+ (U_1 W)^T S_w (U_1 W)\right) = 0.$$

Hence,

$$\begin{aligned} \text{trace}\left(\left((U_1 W)^T S_t (U_1 W)\right)^+ (U_1 W)^T S_b (U_1 W)\right) &= \text{rank}\left((U_1 W)^T S_t (U_1 W)\right) \\ - \quad \text{trace}\left(\left((U_1 W)^T S_t (U_1 W)\right)^+ \left((U_1 W)^T S_w (U_1 W)\right)\right) &= r. \end{aligned}$$

Thus $G = U_1 W$ solves the optimization problem in Eq. (10). That is, OLDA and NLDA are equivalent. ■

Theorem 5.2 above shows that under condition C1, OLDA and NLDA are equivalent. Next, we show that condition C1 holds when the data points are linearly independent as summarized below.

Theorem 5.3 *Assume that condition C2, that is, the n data points in the data matrix $A \in \mathbb{R}^{m\times n}$ are linearly independent, holds. Then condition C1: $rank(S_t) = rank(S_b) + rank(S_w)$ holds.*

Proof Since the n columns in A are linearly independent, $H_t = A - ce^T$ is of rank $n-1$. That is, $\mathrm{rank}(S_t) = n-1$. Next we show that $\mathrm{rank}(S_b) = k-1$ and $\mathrm{rank}(S_w) = n-k$. Thus condition C1 holds.

It is easy to verify that $\mathrm{rank}(S_b) \le k-1$ and $\mathrm{rank}(S_w) \le n-k$. We have

$$n-1 = \mathrm{rank}(S_t) \le \mathrm{rank}(S_b) + \mathrm{rank}(S_w) \le (k-1) + (n-k) = n-1. \tag{17}$$

It follows that all inequalities in Eq. (17) become equalities. That is,

$$\mathrm{rank}(S_b) = k-1,\ \mathrm{rank}(S_w) = n-k,\ \text{and } \mathrm{rank}(S_t) = \mathrm{rank}(S_b) + \mathrm{rank}(S_w). \tag{18}$$

Thus, condition C1 holds. ■

Our experimental results in Section 7 show that for high-dimensional data, the linear independence condition C2 holds in many cases, while condition C1 is satisfied in most cases. This explains why NLDA and OLDA often achieve the same performance in many applications involving high-dimensional data, such as text documents, face images, and gene expression data.

6. Regularized Orthogonal LDA

Recall that OLDA involves the pseudo-inverse of the total scatter matrix, whose estimation may not be reliable especially for undersampled data, where the number of dimensions exceeds the sample size. In such case, the parameter estimates can be highly unstable, giving rise to high variance. By employing a method of regularization, one attempts to improve the estimates by regulating this bias variance trade-off (Friedman, 1989). We employ the regularization technique to OLDA by adding a constant λ to the diagonal elements of the total scatter matrix. Here $\lambda > 0$ is known as the *regularization parameter*. The algorithm is called regularized OLDA (ROLDA). The optimal transformation, G^r, of ROLDA can be computed by solving the following optimization problem:

$$G^r = \mathrm{argmax}_{G\in\mathbb{R}^{m\times\ell}:\, G^TG=I_\ell} \mathrm{trace}\left(\left(G^T(S_t+\lambda I_m)G\right)^+ G^T S_b G\right). \tag{19}$$

The optimal G^r can be computed by solving an eigenvalue problem as summarized in the following theorem (The proof follows Theorem 3.1 in (Ye, 2005) and is thus omitted):

Theorem 6.1 *Let X_q^r be the matrix consisting of the first q eigenvectors of the matrix*

$$(S_t + \lambda I_m)^{-1} S_b \tag{20}$$

corresponding to the nonzero eigenvalues, where $q = rank(S_b)$. Let $X_q^r = QR$ be the QR-decomposition of X_q^r, where Q has orthonormal columns and R is upper triangular. Then $G = Q$ solves the optimization problem in Eq. (19).

Theorem 6.1 implies that the main computation involved in ROLDA is the eigen-decomposition of the matrix $(S_t+\lambda I_m)^{-1}S_b$. Direct formation of the matrix is expensive for high-dimensional data, as it is of size m by m. In the following, we present an efficient way of computing the eigen-decomposition. Denote

$$B^r = (\Sigma_t^2+\lambda I_t)^{-1/2}U_1^T H_b \tag{21}$$

and let

$$B^r = P^r\tilde{\Sigma}^r(Q^r)^T \tag{22}$$

be the SVD of B^r. From Eqs. (8) and (11), we have

$$\begin{aligned}
(S_t+\lambda I_m)^{-1}S_b &= U\begin{pmatrix}(\Sigma_t^2+\lambda I_t)^{-1} & 0\\ 0 & \lambda^{-1}I_{m-t}\end{pmatrix}U^TU\begin{pmatrix}U_1^TS_bU_1 & 0\\ 0 & 0\end{pmatrix}U^T\\
&= U\begin{pmatrix}(\Sigma_t^2+\lambda I_t)^{-1}U_1^TH_bH_b^TU_1 & 0\\ 0 & 0\end{pmatrix}U^T\\
&= U\begin{pmatrix}(\Sigma_t^2+\lambda I_t)^{-1/2}B^r(B^r)^T(\Sigma_t^2+\lambda I_t)^{1/2} & 0\\ 0 & 0\end{pmatrix}U^T\\
&= U\begin{pmatrix}(\Sigma_t^2+\lambda I_t)^{-1/2}P^r\tilde{\Sigma}^r(\tilde{\Sigma}^r)^T(P^r)^T(\Sigma_t^2+\lambda I_t)^{1/2} & 0\\ 0 & 0\end{pmatrix}U^T.
\end{aligned}$$

It follows that the columns of the matrix

$$U_1(\Sigma_t^2+\lambda I_t)^{-1/2}P_q^r$$

form the eigenvectors of $(S_t+\lambda I_m)^{-1}S_b$ corresponding to the top q nonzero eigenvalues, where P_q^r denotes the first q columns of P^r. That is, X_q^r in Theorem 6.1 is given by

$$X_q^r = U_1(\Sigma_t^2+\lambda I_t)^{-1/2}P_q^r. \tag{23}$$

The pseudo-code for the ROLDA algorithm is given in **Algorithm 4**. The computations in ROLDA can be decomposed into two components: the first component involves the matrix, $U_1\in\mathbb{R}^{m\times t}$, of high dimensionality but independent of λ, while the second component involves the matrix,

$$(\Sigma_t^2+\lambda I_t)^{-1/2}P_q^r\in\mathbb{R}^{t\times q},$$

of low dimensionality. When we apply cross-validation to search for the optimal λ from a set of candidates, we repeat the computations involved in the second component only, thus making the computational cost of model selection small.

More specifically, let

$$\Lambda = \{\lambda_1,\cdots,\lambda_{|\Lambda|}\} \tag{24}$$

be the candidate set for the regularization parameter λ, where $|\Lambda|$ denotes the size of the candidate set Λ. We apply v-fold cross-validation for model selection (we choose $v=5$ in our experiment), where the data is divided into v subsets of (approximately) equal size. All subsets are mutually exclusive, and in the i-th fold, the i-th subset is held out for testing and all other subsets are used for training. For each $\lambda_j\,(j=1,\cdots,|\Lambda|)$, we compute the cross-validation accuracy, $\mathrm{Accu}(j)$, defined as the mean of the accuracies for all folds. The optimal regularization value λ_{j^*} is the one with

$$j^* = \arg\max_j \mathrm{Accu}(j). \tag{25}$$

Algorithm 4: ROLDA (Regularized OLDA)

Input: data matrix A and regularization value λ
Output: transformation matrix G^r

1. Compute U_1, Σ_t, and P_q^r, where $q = \text{rank}(S_b)$;
2. $X_q^r \leftarrow U_1(\Sigma_t^2 + \lambda I_t)^{-1/2} P_q^r$;
3. Compute the QR decomposition of X_q^r as $X_q^r = QR$;
4. $G^r \leftarrow Q$.

The K-Nearest Neighbor algorithm with $K = 1$, called 1-NN, is used for computing the accuracy. The pseudo-code for the model selection procedure in ROLDA is given in **Algorithm 5**. Note that we apply the QR decomposition to

$$(\Sigma_t^2 + \lambda I_t)^{-1/2} P_q^r \in \mathbb{R}^{t \times q} \tag{26}$$

instead of

$$X_q^r = U_1(\Sigma_t^2 + \lambda I_t)^{-1/2} P_q^r \in \mathbb{R}^{m \times q}, \tag{27}$$

as done in Theorem 6.1, since U_1 has orthonormal columns.

Algorithm 5: Model selection for ROLDA

Input: data matrix A and candidate set $\Lambda = \{\lambda_1, \cdots, \lambda_{|\Lambda|}\}$
Output: optimal regularization value λ_{j^*}

1. For $i = 1 : v$ /* v-fold cross-validation */
2. Construct A^i and $A^{\hat{i}}$;
   /* $A^i = i$-th fold, for training and $A^{\hat{i}} =$ rest, for testing */
3. Construct H_b and H_t using A^i as in Eqs. (2) and (3), respectively;
4. Compute the reduced SVD of H_t as $H_t = U_1 \Sigma_t V_1^T$; $t \leftarrow \text{rank}(H_t)$;
5. $H_{b,L} \leftarrow U_1^T H_b$, $q \leftarrow \text{rank}(H_b)$;
6. $A_L^i \leftarrow U_1^T A^i$; $A_L^{\hat{i}} \leftarrow U_1^T A^{\hat{i}}$; /* Projection by U_1 */
7. For $j = 1 : |\Lambda|$ /* $\lambda_1, \cdots, \lambda_{|\Lambda|}$ */
8. $D_j \leftarrow (\Sigma_t^2 + \lambda_j I_t)^{-1/2}$; $B^r \leftarrow D_j H_{b,L}$
9. Compute the SVD of B^r as $B^r = P^r \tilde{\Sigma}^r (Q^r)^T$;
10. $D_{q,P} \leftarrow D_j P_q^r$; Compute the QR decomposition of $D_{q,P}$ as $D_{q,P} = QR$;
11. $A_L^i \leftarrow Q^T A_L^i$; $A_L^{\hat{i}} \leftarrow Q^T A_L^{\hat{i}}$;
12. Run 1-NN on $\left(A_L^i, A_L^{\hat{i}}\right)$ and compute the accuracy, denoted as $\text{Accu}(i, j)$;
13. EndFor
14. EndFor
15. $\text{Accu}(j) \leftarrow \frac{1}{v} \sum_{i=1}^{v} \text{Accu}(i, j)$;
16. $j^* \leftarrow \arg\max_j \text{Accu}(j)$;
17. Output λ_{j^*} as the optimal regularization value.

6.1 Time Complexity

We conclude this section by analyzing the time complexity of the model selection procedure described above.

Line 4 in **Algorithm 5** takes $O(n^2m)$ time for the reduced SVD computation. Lines 5 and 6 take $O(mtk) = O(mnk)$ and $O(tmn) = O(mn^2)$ time, respectively, for the matrix multiplications. For each λ_j, for $j = 1, \cdots, |\Lambda|$, of the "For" loop, Lines 9 and 10 take $O(tk^2) = O(nk^2)$ time for the SVD and QR decomposition and matrix multiplication. Line 11 takes $O(ktn) = O(kn^2)$ time for the matrix multiplication. The computation of the classification accuracy by 1-NN in Line 12 takes $O(n^2k/v)$ time, as the size of the test set, $A_L^{\hat{i}}$, is about n/v. Thus, the time complexity, $T(|\Lambda|)$, of the model selection procedure is

$$T(|\Lambda|) \quad = \quad O\left(v\left(n^2m + mn^2 + mnk + |\Lambda|(nk^2 + kn^2 + n^2k/v)\right)\right).$$

For high-dimensional and undersampled data, where the sample size, n, is much smaller than the dimensionality m, the time complexity is simplified to

$$T(|\Lambda|) = O\left(v(n^2m + |\Lambda|n^2k)\right) = O\left(vn^2m\left(1 + \frac{k}{m}|\Lambda|\right)\right).$$

When the number, k, of classes in the data set is much smaller than the dimensionality, m, the overhead of estimating the optimal regularization value among a large candidate set may be small. Our experiments on a collection of high-dimensional and undersampled data (see Section 7) show that the computational cost of the model selection procedure in ROLDA grows slowly as $|\Lambda|$ increases.

7. Experimental Studies

In this section, we perform extensive experimental studies to evaluate the theoretical results and the ROLDA algorithm presented in this paper. Section 7.1 describes our test data sets. We perform a detailed comparison of NLDA, iNLDA, and OLDA in Section 7.2. Results are consistent with our theoretical analysis. In Section 7.3, we compare the classification performance of NLDA, iNLDA, OLDA, ULDA, ROLDA, and SVM. The K-Nearest-Neighbor (K-NN) algorithm with $K = 1$ is used as the classifier for all LDA based algorithms.

7.1 Data Sets

We used 14 data sets from various data sources in our experimental studies. The statistics of our test data sets are summarized in Table 2.

The first five data sets, including spambase,[4] balance, wine, waveform, and vowel, are low-dimensional data from the UCI Machine Learning Repository. The next nine data sets, including text documents, face images, and gene expression data, have high dimensionality: re1, re0, and tr41 are three text document data sets, where re1 and re0 are derived from *Reuters-21578* text categorization test collection Distribution 1.0,[5] and tr41 is derived from the TREC-5, TREC-6, and TREC-7 collections;[6] ORL,[7] AR,[8] and PIX[9] are three face image data sets; GCM, colon, and ALLAML4 are three gene expression data sets (Ye et al., 2004b).

4. Only a subset of the original spambase data set is used in our study.
5. http://www.daviddlewis.com/resources/testcollections/reuters21578/
6. http://trec.nist.gov
7. http://www.uk.research.att.com/facedatabase.html
8. http://rvl1.ecn.purdue.edu/∼aleix/aleix_face_DB.html
9. http://peipa.essex.ac.uk/ipa/pix/faces/manchester/test-hard/

Data Set	Sample size (n)			# of dimensions	# of classes
	training	test	total	(m)	(k)
spambase	400	600	1000	56	2
balance	416	209	625	4	3
wine	118	60	178	13	3
waveform	300	500	800	21	3
vowel	528	462	990	10	11
re1	—	—	490	3759	5
re0	—	—	320	2887	4
tr41	—	—	210	7454	7
ORL	—	—	400	10304	40
AR	—	—	650	8888	50
PIX	—	—	300	10000	30
GCM	—	—	198	16063	14
colon	—	—	62	2000	2
ALLAML4	—	—	72	7129	4

Table 2: Statistics of our test data sets. For the first five data sets, we used the given partition of training and test sets, while for the last nine data sets, we did random splittings into training and test sets of ratio 2:1.

7.2 Comparison of NLDA, iNLDA, and OLDA

In this experiment, we did a comparative study of NLDA, iNLDA, and OLDA. For the first five low-dimensional data sets from the UCI Machine Learning Repository, we used the given splitting of training and test sets. The result is summarized in Table 3. For the next nine high-dimensional data sets, we performed our study by repeated random splittings into training and test sets. The data was partitioned randomly into a training set, where each class consists of two-thirds of the whole class and a test set with each class consisting of one-third of the whole class. The splitting was repeated 20 times and the resulting accuracies of different algorithms for the first ten splittings are summarized in Table 4. Note that the mean accuracy for the 20 different splittings will be reported in the next section. The rank of three scatter matrices, S_b, S_w, and S_t, for each of the splittings is also reported.

The main observations from Table 3 and Table 4 include:

- For the first five low-dimensional data sets, we have $\text{rank}(S_b) = k-1$, and $\text{rank}(S_w) = \text{rank}(S_t) = m$, where m is the data dimensionality. Thus the null space of $\tilde{S}_w$ is empty, and both NLDA and iNLDA do not apply. However, OLDA is applicable and the reduced dimensionality of OLDA is $k-1$.

- For the next nine high-dimensional data sets, condition C1: $\text{rank}(S_t) = \text{rank}(S_b) + \text{rank}(S_w)$ is satisfied in all cases except the re0 data set. For the re0 data set, either $\text{rank}(S_t) = \text{rank}(S_b) + \text{rank}(S_w)$ or $\text{rank}(S_t) = \text{rank}(S_b) + \text{rank}(S_w) - 1$ holds, that is, condition C1 is not severely violated for re0. Note that re0 has the smallest number of dimensions among the nine high-

		Data Set				
		spambase	balance	wine	waveform	vowel
Method	NLDA	—	—	—	—	—
	iNLDA	—	—	—	—	—
	OLDA	88.17	86.60	98.33	73.20	56.28
Rank	S_b	1	2	2	2	10
	S_w	56	4	13	21	10
	S_t	56	4	13	21	10

Table 3: Comparison of NLDA, iNLDA, and OLDA on classification accuracy (in percentage) using five low-dimensional data sets from UCI Machine Learning Repository. The ranks of three scatter matrices are reported.

dimensional data sets. From the experiments, we may infer that condition C1 is more likely to hold for high-dimensional data.

- NLDA, iNLDA, and OLDA achieve the same classification performance in all cases when condition C1 holds. The empirical result confirms the theoretical analysis in Section 5. This explains why NLDA and OLDA often achieve similar performance for high-dimensional data. We can also observe that NLDA and iNLDA achieve similar performance in all cases.

- The numbers of training data points for the nine high-dimensional data (in the same order as in the table) are 325, 212, 140, 280, 450, 210, 125, 68, and 48, respectively. By examining the rank of S_t in Table 4, we can observe that the training data in six out of nine data sets, including tr41, ORL, AR, GCM, colon, and ALLAML4, are linearly independent. That is, the independence assumption C2 from Theorem 5.3 holds for these data sets. It is clear from the table that for these six data sets, condition C1 holds and NLDA, iNLDA, and OLDA achieve the same performance. These are consistent with the theoretical analysis in Section 5.

- For the re0 data set, where condition C1 does not hold, i.e., $\text{rank}(S_t) < \text{rank}(S_b) + \text{rank}(S_w)$, OLDA achieves higher classification accuracy than NLDA and iNLDA. Recall that the reduced dimensionality of OLDA equals $\text{rank}(S_b) \equiv q$. The reduced dimensionality in NLDA and iNLDA equals the dimension of the null space of $\tilde{S}_w$, which equals $\text{rank}(S_t) - \text{rank}(S_w) < \text{rank}(S_b)$. That is, OLDA keeps more dimensions in the transformed space than NLDA and iNLDA. Experimental results in re0 show that these extra dimensions used in OLDA improve its classification performance.

7.3 Comparative Studies on Classification

In this experiment, we conducted a comparative study of NLDA, iNLDA, OLDA, ULDA, ROLDA, and SVM in terms of classification. For ROLDA, the optimal λ is estimated through cross-validation on a candidate set, $\Lambda = \{\lambda_j\}_{j=1}^{|\Lambda|}$. Recall that $T(|\Lambda|)$ denotes the computational cost of the model selection procedure in ROLDA, where $|\Lambda|$ is the size of the candidate set of the regularization values. We have performed model selection on all nine high-dimensional data sets using different values of

Data Set	Method	Ten different splittings into training and test sets of ratio 2:1									
re1	NLDA	92.73	93.33	93.33	93.94	94.55	95.15	96.36	95.15	92.12	93.94
	iNLDA	92.73	93.33	93.33	93.94	94.55	95.15	96.36	95.15	92.12	93.94
	OLDA	92.73	93.33	93.33	93.94	94.55	95.15	96.36	95.15	92.12	93.94
	S_b	4	4	4	4	4	4	4	4	4	4
	S_w	316	318	319	316	316	320	316	318	317	318
	S_t	320	322	323	320	320	324	320	322	321	322
re0	NLDA	64.81	62.04	64.81	68.52	87.96	70.37	71.30	73.15	87.04	75.93
	iNLDA	65.74	62.04	64.81	69.44	87.96	70.37	71.30	72.22	87.04	75.93
	OLDA	75.93	75.00	77.78	74.07	87.96	80.56	74.07	78.70	87.04	79.63
	S_b	3	3	3	3	3	3	3	3	3	3
	S_w	205	204	203	203	205	204	201	203	203	205
	S_t	207	206	205	205	208	206	203	205	206	207
tr41	NLDA	97.14	95.71	97.14	98.57	97.14	98.57	100.0	95.71	98.57	95.71
	iNLDA	97.14	95.71	97.14	98.57	97.14	98.57	100.0	95.71	98.57	95.71
	OLDA	97.14	95.71	97.14	98.57	97.14	98.57	100.0	95.71	98.57	95.71
	S_b	6	6	6	6	6	6	6	6	6	6
	S_w	133	133	133	133	133	133	133	133	133	133
	S_t	139	139	139	139	139	139	139	139	139	139
ORL	NLDA	99.17	96.67	98.33	98.33	95.00	95.83	98.33	97.50	98.33	95.83
	iNLDA	99.17	96.67	98.33	98.33	95.00	95.83	98.33	97.50	98.33	95.83
	OLDA	99.17	96.67	98.33	98.33	95.00	95.83	98.33	97.50	98.33	95.83
	S_b	39	39	39	39	39	39	39	39	39	39
	S_w	240	240	240	240	240	240	240	240	240	240
	S_t	279	279	279	279	279	279	279	279	279	279
AR	NLDA	96.50	94.50	96.50	94.00	93.50	94.50	93.50	97.00	94.00	96.00
	iNLDA	96.50	94.50	96.50	94.00	93.50	94.50	93.50	97.00	94.00	96.00
	OLDA	96.50	94.50	96.50	94.00	93.50	94.50	93.50	97.00	94.00	96.00
	S_b	49	49	49	49	49	49	49	49	49	49
	S_w	400	400	400	400	400	400	400	400	400	400
	S_t	449	449	449	449	449	449	449	449	449	449
PIX	NLDA	98.89	97.78	98.89	97.78	98.89	98.89	98.89	97.78	98.89	97.78
	iNLDA	98.89	97.78	98.89	97.78	98.89	98.89	98.89	97.78	98.89	97.78
	OLDA	98.89	97.78	98.89	97.78	98.89	98.89	98.89	97.78	98.89	97.78
	S_b	29	29	29	29	29	29	29	29	29	29
	S_w	178	179	179	179	178	180	179	179	180	178
	S_t	207	208	208	208	207	209	208	208	209	207
GCM	NLDA	81.54	80.00	81.54	83.08	84.62	87.69	75.38	78.46	84.62	83.08
	iNLDA	81.54	80.00	81.54	83.08	84.62	87.69	75.38	78.46	84.62	83.08
	OLDA	81.54	80.00	81.54	83.08	84.62	87.69	75.38	78.46	84.62	83.08
	S_b	13	13	13	13	13	13	13	13	13	13
	S_w	111	111	111	111	111	111	111	111	111	111
	S_t	124	124	124	124	124	124	124	124	124	124
colon	NLDA	91.18	94.12	100.0	97.06	91.18	91.18	97.06	94.12	94.12	97.06
	iNLDA	91.18	94.12	100.0	97.06	91.18	91.18	97.06	94.12	94.12	97.06
	OLDA	91.18	94.12	100.0	97.06	91.18	91.18	97.06	94.12	94.12	97.06
	S_b	1	1	1	1	1	1	1	1	1	1
	S_w	66	66	66	66	66	66	66	66	66	66
	S_t	67	67	67	67	67	67	67	67	67	67
ALLAML4	NLDA	95.83	91.67	95.83	95.83	87.50	95.83	95.83	100.0	91.67	95.83
	iNLDA	95.83	91.67	95.83	95.83	87.50	95.83	95.83	100.0	91.67	95.83
	OLDA	95.83	91.67	95.83	95.83	87.50	95.83	95.83	100.0	91.67	95.83
	S_b	3	3	3	3	3	3	3	3	3	3
	S_w	44	44	44	44	44	44	44	44	44	44
	S_t	47	47	47	47	47	47	47	47	47	47

Table 4: Comparison of classification accuracy (in percentage) for NLDA, iNLDA, and OLDA using nine high-dimensional data sets. Ten different splittings into training and test sets of ratio 2:1 (for each of the k classes) are applied. The rank of three scatter matrices for each splitting is reported.

Data Set	NLDA	iNLDA	OLDA	ULDA	ROLDA	SVM
re1	94.33 (1.72)	94.33 (1.72)	94.33 (1.72)	94.76 (1.67)	94.79 (1.64)	94.54 (1.88)
re0	74.03 (9.22)	74.15 (8.19)	79.54 (4.73)	79.72 (4.82)	85.79 (3.66)	85.87 (3.34)
tr41	97.00 (2.01)	97.00 (2.01)	97.00 (2.01)	97.14 (2.02)	97.17 (2.04)	97.14 (2.01)
ORL	97.29 (1.79)	97.29 (1.79)	97.29 (1.79)	92.75 (1.82)	97.52 (1.64)	97.55 (1.34)
AR	95.42 (1.30)	95.42 (1.30)	95.42 (1.30)	94.37 (1.46)	97.30 (1.32)	95.75 (1.43)
PIX	98.22 (1.41)	98.22 (1.41)	98.22 (1.41)	96.61 (1.92)	98.29 (1.32)	98.50 (1.24)
GCM	81.77 (3.61)	81.77 (3.61)	81.77 (3.61)	80.46 (3.71)	82.69 (3.42)	75.31 (4.45)
Colon	86.50 (5.64)	86.50 (5.64)	86.50 (5.64)	86.50 (5.64)	87.00 (6.16)	87.25 (5.25)
ALLAML4	93.54 (3.70)	93.54 (3.70)	93.54 (3.70)	93.75 (3.45)	93.75 (3.45)	93.70 (3.40)

Table 5: Comparison of classification accuracy (in percentage) for six different methods: NLDA, iNLDA, OLDA, ULDA, ROLDA, and SVM using nine high-dimensional data sets. The mean accuracy and standard deviation (in parenthesis) from 20 different runs are reported.

$|\Lambda|$. We have observed that $T(|\Lambda|)$ grows slowly as $|\Lambda|$ increases, and the ratio, $T(1024)/T(1)$, on all nine data sets ranges from 1 to 5. Thus, we can run model selection using a large candidate set of regularization values, without dramatically increasing the cost. In the following experiments, we apply model selection to ROLDA with a candidate set of size $|\Lambda| = 1024$, where

$$\lambda_j = \alpha_j/(1-\alpha_j), \tag{28}$$

with $\{\alpha_j\}_{j=1}^{|\Lambda|}$ uniformly distributed between 0 and 1. As for SVM, we employed the cross-validation to estimate the optimal parameter using a candidate set of size 50. To compare different classification algorithms, we applied the same experimental setting as in Section 7.2. The splitting into training and test sets of ratio 2:1 (for each of the *k* classes) was repeated 20 times. The final accuracy reported was the average of the 20 different runs. The standard deviation for each data set was also reported. The result on the nine high-dimensionality data sets is summarized in Table 5.

As observed in Section 7.2, OLDA has the same performance as NLDA and iNLDA in all cases except the re0 data set, while NLDA and iNLDA achieve similar performance in all cases. Overall, ROLDA and SVM are very competitive with other methods. SVM performs well in all cases except GCM. The poor performance of SVM in GCM has also been observed in (Li et al., 2004). ROLDA outperforms OLDA for re0, AR, and GCM, while it is comparable to OLDA for all other cases. This confirms the effectiveness of the regularization applied in ROLDA. Note that from Remark 1, ULDA is closely related to OLDA. However, unlike OLDA, ULDA does not apply the final orthogonalization step. Experimental result in Table 5 confirms the effectiveness of the orthogonalization step in OLDA, especially for three face image data sets and GCM.

8. Conclusions

In this paper, we present a computational and theoretical analysis of two LDA based algorithms, including null space LDA and orthogonal LDA. NLDA computes the discriminant vectors in the null space of the within-class scatter matrix, while OLDA computes a set of orthogonal discriminant vectors via the simultaneous diagonalization of the scatter matrices. They have been applied successfully in many applications, such as document classification, face recognition, and gene expression data classification.

Both NLDA and OLDA result in orthogonal transformations. However, they applied different schemes in deriving the optimal transformation. Our theoretical analysis in this paper shows that under a mild condition C1 which holds in many applications involving high-dimensional data, NLDA is equivalent to OLDA. Based on the theoretical analysis, an improved algorithm for null space LDA algorithm, called iNLDA, is proposed. We have performed extensive experimental studies on 14 data sets, including both low-dimensional and high-dimensional data. Results have shown that condition C1 holds for eight out of the nine high-dimensional data sets, while the null space of $\tilde{S}_w$ is empty for all five low-dimensional data. Thus, NLDA may not be applicable for low-dimensional data, while OLDA is still applicable in this case. Results are also consistent with our theoretical analysis. That is, for all cases when condition C1 holds, NLDA, iNLDA, and OLDA achieve the same classification performance. We also observe that for other cases with condition C1 violated, OLDA outperforms NLDA and iNLDA, due to the extra number of dimensions used in OLDA. We also compare NLDA, iNLDA, and OLDA with uncorrelated LDA (ULDA), which does not perform the final orthogonalization step. Results show that OLDA is very competitive with ULDA, which confirms the effectiveness of the orthogonalization step used in OLDA. Our empirical and theoretical results presented in this paper provide further insights into the nature of these two LDA based algorithms.

We also present the ROLDA algorithm, which extends the OLDA algorithm by applying the regularization technique. Regularization may stabilize the sample covariance matrix estimation and improve the classification performance. ROLDA involves the regularization parameter λ, which is commonly estimated via cross-validation. To speed up the cross-validation process, we decompose the computations in ROLDA into two components: the first component involves matrices of high dimensionality but independent of λ, while the second component involves matrices of low dimensionality. When searching for the optimal λ from a candidate set, we repeat the computations involved in the second component only. A comparative study on classification shows that ROLDA is very competitive with OLDA, which shows the effectiveness of the regularization applied in ROLDA.

Our extensive experimental studies have shown that condition C1 holds for most high-dimensional data sets. We plan to carry out theoretical analysis on this property in the future. Some of the theoretical results in (Hall et al., 2005) may be useful for our analysis.

The algorithms in (Yang et al., 2005; Yu and Yang, 2001) are closely related to the null space LDA algorithm discussed in this paper. The analysis presented in this paper may be useful in understanding why these algorithms perform well in many applications, especially in face recognition. We plan to explore this further in the future.

Acknowledgements

We thank the reviewers for helpful comments. Research of JY is sponsored, in part, by the Center for Evolutionary Functional Genomics of the Biodesign Institute at the Arizona State University.

References

P. N. Belhumeour, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans Pattern Analysis and Machine Intelligence*, 19 (7):711–720, 1997.

R. Bellmanna. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

M. W. Berry, S. T. Dumais, and G. W. O'Brie. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

L. F. Chen, H. Y. M. Liao, M. T. Ko, J. C. Lin, and G. J. Yu. A new LDA-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33:1713–1726, 2000.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Scienc*, 41:391–407, 1990.

L. Duchene and S. Leclerq. An optimal transformation for discriminant and principal component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(6):978–983, 1988.

R. O. Duda, P. E. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

S. Dudoit, J. Fridlyand, and T. P. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457): 77–87, 2002.

D. H. Foley and J. W. Sammon. An optimal set of discriminant vectors. *IEEE Trans Computers*, 24 (3):281–289, 1975.

J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

K. Fukunaga. *Introduction to Statistical Pattern Classification*. Academic Press, San Diego, California, USA, 1990.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

P. Hall, J. S. Marron, and A. Neeman. Geometric representation of high dimension, low sample size data. *Journal of the Royal Statistical Society series B*, 67:427–444, 2005.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer, 2001.

R. Huang, Q. Liu, H. Lu, and S. Ma. Solving the small sample size problem of LDA. In *Proc. International Conference on Pattern Recognition*, pages 29–32, 2002.

Z. Jin, J. Y. Yang, Z. S. Hu, and Z. Lou. Face recognition based on the uncorrelated discriminant transformation. *Pattern Recognition*, 34:1405–1416, 2001.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

W. J. Krzanowski, P. Jonathan, W.V McCarthy, and M. R. Thomas. Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data. *Applied Statistics*, 44:101–115, 1995.

T. Li, C. Zhang, and M. Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15): 2429–2437, 2004.

W. Liu, Y. Wang, S. Z. Li, and T. Tan. Null space approach of Fisher discriminant analysis for face recognition. In *Proc. European Conference on Computer Vision, Biometric Authentication Workshop*, 2004.

J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Face recognition using kernel direct discriminant analysis algorithms. *IEEE Trans. Neural Networks*, 14(1):117–126, 2003.

S. Raudys and R. P. W. Duin. On expected classification error of the Fisher linear classifier with pseudo-inverse covariance matrix. *Pattern Recognition Letters*, 19(5-6):385–392, 1998.

B. Schökopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

M. Skurichina and R. P. W. Duin. Stabilizing classifiers for very small sample size. In *Proc. International Conference on Pattern Recognition*, pages 891–896, 1996.

D. L. Swets and J. Y. Weng. Using discriminant eigenfeatures for image retrieval. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(8):831–836, 1996.

M. A. Turk and A. P. Pentland. Face recognition using Eigenfaces. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 586–591, 1991.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

J. Yang, A. F. Frangi, J. Y. Yang, D. Zhang, and Z. Jin. KPCA plus LDA: a complete kernel Fisher discriminant framework for feature extraction and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(2):230– 244, 2005.

J. Ye. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6:483–502, 2005.

J. Ye, R. Janardan, Q. Li, and H. Park. Feature extraction via generalized uncorrelated linear discriminant analysis. In *Proc. International Conference on Machine Learning*, pages 895–902, 2004a.

J. Ye, T. Li, T. Xiong, and R. Janardan. Using uncorrelated discriminant analysis for tissue classification with gene expression data. *IEEE/ACM Trans. Computational Biology and Bioinformatics*, 1(4):181–190, 2004b.

H. Yu and J. Yang. A direct LDA algorithm for high-dimensional data with applications to face recognition. *Pattern Recognition*, 34:2067–2070, 2001.

Journal of Machine Learning Research 7 (2006) 1205–1230 Submitted 10/05; Revised 3/06; Published 7/06

Worst-Case Analysis of Selective Sampling for Linear Classification

Nicolò Cesa-Bianchi CESA-BIANCHI@DSI.UNIMI.IT
DSI, Università di Milano
via Comelico, 39
20135 Milano, Italy

Claudio Gentile CLAUDIO.GENTILE@UNINSUBRIA.IT
DICOM, Università dell'Insubria
via Mazzini, 5
21100 Varese, Italy

Luca Zaniboni ZANIBONI@DTI.UNIMI.IT
DTI, Università di Milano
via Bramante, 65
26013 Crema, Italy

Editor: Manfred Warmuth

Abstract

A selective sampling algorithm is a learning algorithm for classification that, based on the past observed data, decides whether to ask the label of each new instance to be classified. In this paper, we introduce a general technique for turning linear-threshold classification algorithms from the general additive family into randomized selective sampling algorithms. For the most popular algorithms in this family we derive mistake bounds that hold for individual sequences of examples. These bounds show that our semi-supervised algorithms can achieve, on average, the same accuracy as that of their fully supervised counterparts, but using fewer labels. Our theoretical results are corroborated by a number of experiments on real-world textual data. The outcome of these experiments is essentially predicted by our theoretical results: Our selective sampling algorithms tend to perform as well as the algorithms receiving the true label after each classification, while observing in practice substantially fewer labels.

Keywords: selective sampling, semi-supervised learning, on-line learning, kernel algorithms, linear-threshold classifiers

1. Introduction

A selective sampling algorithm (see, e.g., Cohn et al., 1990; Cesa-Bianchi et al., 2003; Freund et al., 1997) is a learning algorithm for classification that receives a sequence of unlabelled instances and decides whether to query the label of the current instance based on the past observed data. The idea is to let the algorithm determine which labels are most useful to its inference mechanism, and thus achieve a good classification performance while using fewer labels.

Natural real-world scenarios for selective sampling are all those applications where labels are scarce or expensive to obtain. For example, collecting web pages is a fairly automated process, but assigning them a label (e.g., from a set of possible *topics*) often requires time-consuming and

costly human expertise. For this reason, it is clearly important to devise learning algorithms having the ability to exploit the label information as much as possible. An additional motivation for using selective sampling arises from the widespread use of kernel-based algorithms (Vapnik, 1998; Cristianini and Shawe-Taylor, 2001; Schölkopf and Smola, 2002). In this case, saving labels implies using fewer support vectors to represent the hypothesis, which in turn entails a more efficient use of the memory and a shorter running time in both training and test phases.

Many algorithms have been proposed in the literature to cope with the broad task of learning with partially labelled data, working under both probabilistic and worst-case assumptions for either on-line or batch settings. These range from active learning algorithms (Campbell et al., 2000; Tong and Koller, 2000), to the query-by-committee algorithm (Freund et al., 1997), to the adversarial "apple tasting" and label efficient algorithms investigated by Helmbold et al. (2000) and Helmbold and Panizza (1997), respectively. More recent work on this subject includes (Bordes et al., 2005; Dasgupta et al., 2005; Dekel et al., 2006).

In this paper we present a mistake bound analysis for selective sampling versions of Perceptron-like algorithms. In particular, we study the standard Perceptron algorithm (Rosenblatt, 1958; Block, 1962; Novikov, 1962) and the second-order Perceptron algorithm (Cesa-Bianchi et al., 2005). Then, we argue how to extend the above analysis to the general additive family of linear-threshold algorithms introduced by Grove et al. (2001) and Warmuth and Jagota (1997) (see also Cesa-Bianchi and Lugosi, 2003; Gentile, 2003; Gentile and Warmuth, 1999; Kivinen and Warmuth, 2001), and we provide details for a specific algorithm in this family, i.e., the (zero-threshold) Winnow algorithm (Littlestone, 1988, 1989; Grove et al., 2001).

Our selective sampling algorithms use a simple randomized rule to decide whether to query the label of the current instance. This rule prescribes that the label should be obtained with probability $b/(b+|\widehat{p}|)$, where $\widehat{p}$ is the (signed) margin achieved by the current linear hypothesis on the current instance, and $b > 0$ is a parameter of the algorithm acting as a scaling factor on $\widehat{p}$. Note that a label is queried with a small probability whenever the margin $\widehat{p}$ is large in magnitude. If the label is obtained, and it turns out that a mistake has been made, then the algorithm proceeds with its standard update rule. Otherwise, the algorithm's current hypothesis is left unchanged. It is important to remark that in our model we evaluate algorithms by counting their prediction mistakes also on those time steps when the true labels remain unknown. For each of the algorithms we consider a bound is proven on the expected number of mistakes made in an arbitrary data sequence, where the expectation is with respect to the randomized sampling rule.

Our analysis reveals an interesting phenomenon. In all algorithms we analyze, a proper choice of the scaling factor b in the randomized rule yields the same mistake bound as the one achieved by the original algorithm before the introduction of the selective sampling mechanism. Hence, in some sense, our technique exploits the margin information to select those labels that can be ignored without increasing (in expectation) the overall number of mistakes.

One may suspect that this gain is not real: it might very well be the case that the tuning of b preserving the original mistake bound forces the algorithm to query all but an insignificant number of labels. In the last part of the paper we present some experiments contradicting this conjecture. In particular, by running our algorithms on real-world textual data, we show that no significant decrease in the predictive performance is suffered even when b is set to values that leave a significant fraction of the labels unobserved.

The paper is organized as follows. In the remainder of this introduction we give the notation and the basic definitions used throughout the paper. In Section 2 we describe and analyze our

Perceptron-like selective sampling algorithms. In Section 3 we extend our margin-based argument to the zero-threshold Winnow algorithm. Empirical comparisons are reported in Section 4. Finally, Section 5 is devoted to conclusions and open problems.

Notation and basic definitions

An *example* is a pair (x,y), where $x \in \mathbb{R}^d$ is an *instance* vector and $y \in \{-1,+1\}$ is the associated binary label.

We consider the following selective sampling variant of the standard on-line learning model (Angluin, 1988; Littlestone, 1988). Learning proceeds in a sequence of *trials*. In the generic trial t the algorithm observes instance x_t, outputs a prediction $\widehat{y}_t \in \{-1,+1\}$ for the label y_t associated with x_t, and decides whether or not to ask the label y_t. No matter what the algorithm decides, we say that the algorithm has made a *prediction mistake* if $\widehat{y}_t \neq y_t$. We measure the performance of a linear-threshold algorithm by the total number of mistakes it makes on a sequence of examples (including the trials where the true label y_t remains unknown). The goal of the algorithm is to bound, on an *arbitrary* sequence of examples, the amount by which this total number of mistakes exceeds the performance of the best linear predictor in hindsight.

In this paper we are concerned with selective sampling versions of linear-threshold algorithms. When run on a sequence $(x_1,y_1),(x_2,y_2),\ldots$ of examples, these algorithms compute a sequence $w_0,w_1,\ldots$ of weight vectors $w_t \in \mathbb{R}^d$, where w_t can only depend on the past examples (x_1,y_1), ..., (x_t,y_t) but not on the future ones, (x_s,y_s) for $s > t$. In each trial $t = 1,2,\ldots$ the linear-threshold algorithm predicts y_t using[1] $\widehat{y}_t = \mathrm{SGN}(\widehat{p}_t)$ where $\widehat{p}_t = w_{t-1}^\top x_t$ is the margin of w_{t-1} on the instance x_t. If the label y_t is queried, then the algorithm (possibly) uses y_t to compute a new weight w_t; on the other hand, if y_t remains unknown then $w_t = w_{t-1}$.

We identify an arbitrary linear-threshold classifier with its coefficient vector $u \in \mathbb{R}^d$. For a fixed sequence $(x_1,y_1),\ldots,(x_n,y_n)$ of examples and a given margin threshold $\gamma > 0$, we measure the performance of u by its cumulative *hinge loss* (Freund and Schapire, 1999; Gentile and Warmuth, 1999)

$$L_{\gamma,n}(u) = \sum_{t=1}^{n} \ell_{\gamma,t}(u) = \sum_{t=1}^{n} (\gamma - y_t u^\top x_t)_+$$

where we used the notation $(x)_+ = \max\{0,x\}$. In words, the hinge loss, also called *soft margin* in the statistical learning literature (Vapnik, 1998; Cristianini and Shawe-Taylor, 2001; Schölkopf and Smola, 2002), measures the extent to which the hyperplane u separates the sequence of examples with margin at least γ.

We represent the algorithm's decision of querying the label at time t through the value of a Bernoulli random variable Z_t, whose parameter is determined by the specific selection rule used by the algorithm under consideration. Though we make no assumptions on the source generating the sequence $(x_1,y_1),(x_2,y_2),\ldots$, we require that each example (x_t,y_t) be generated before the value of Z_t is drawn. In other words, the source cannot use the knowledge of Z_t to determine x_t and y_t. We use $\mathbb{E}_{t-1}[\cdot]$ to denote the conditional expectation $\mathbb{E}[\cdot \,|\, Z_1,\ldots,Z_{t-1}]$ and M_t to denote the indicator function of the event $\widehat{y}_t \neq y_t$, where $\widehat{y}_t$ is the prediction at time t of the algorithm under consideration.

1. Here and throughout SGN denotes the signum function $\mathrm{SGN}(x) = 1$ if $x > 0$ and $\mathrm{SGN}(x) = -1$, otherwise.

Selective sampling Perceptron.
Parameters: $b > 0$.
Initialization: $w_0 = (0, \ldots, 0)^\top$.
For each trial $t = 1, 2, \ldots$

(1) observe an instance vector $x_t \in \mathbb{R}^d$, and set $\widehat{p}_t = w_{t-1}^\top x_t$;

(2) predict with $\widehat{y}_t = \mathrm{SGN}(\widehat{p}_t)$;

(3) draw a Bernoulli random variable $Z_t \in \{0,1\}$ of parameter $\dfrac{b}{b+|\widehat{p}_t|}$;

(4) **if** $Z_t = 1$ **then** query label $y_t \in \{-1,+1\}$ and perform the standard Perceptron update: $w_t = w_{t-1} + M_t y_t x_t$;

(5) **else** ($Z_t = 0$) set $w_t = w_{t-1}$.

Figure 1: A selective sampling version of the classical Perceptron algorithm.

Finally, whenever the distribution laws of $Z_1, Z_2, \ldots$ and $M_1, M_2, \ldots$ are clear from the context, we use the abbreviation

$$\overline{L}_{\gamma,n}(u) = \mathbb{E}\left[\sum_{t=1}^{n} M_t Z_t \ell_{\gamma,t}(u)\right] .$$

Note that $\overline{L}_{\gamma,n}(u) \le L_{\gamma,n}(u)$ trivially holds for all choices of γ, n, and u.

2. Selective Sampling Algorithms and Their Analysis

In this section we describe and analyze three algorithms: a selective sampling version of the classical Perceptron algorithm (Rosenblatt, 1958; Block, 1962; Novikov, 1962), a variant of the same algorithm with a dynamically tuned parameter, and a selective sampling version of the second-order Perceptron algorithm (Cesa-Bianchi et al., 2005). It is worth pointing out that, like any Perceptron-like update rule, each of the algorithms presented in this section can be efficiently run in any given reproducing kernel Hilbert space once the update rule is expressed in an equivalent dual-variable form (see, e.g., Vapnik, 1998; Cristianini and Shawe-Taylor, 2001; Schölkopf and Smola, 2002). Note that, in the case of kernel-based algorithms, label efficiency provides the additional benefit of a more compact representation of the trained classifiers. The experiments reported in Section 4 were indeed obtained using a dual-variable implementation of our algorithms.

2.1 Selective Sampling Perceptron

Our selective sampling variant of the classical Perceptron algorithm is described in Figure 1. The algorithm maintains a vector $w \in \mathbb{R}^d$ (whose initial value is zero). In each trial t the algorithm observes an instance vector $x_t \in \mathbb{R}^d$ and predicts the binary label y_t through the sign of the margin

value $\widehat{p}_t = w_{t-1}^\top x_t$. Then the algorithm decides whether to query the label y_t through the randomized rule described in the introduction: a coin with bias $b/(b+|\widehat{p}_t|)$ is flipped; if the coin turns up heads ($Z_t = 1$ in Figure 1), then the label y_t is queried. If a prediction mistake is observed ($\widehat{y}_t \neq y_t$), then the algorithm updates vector w_t according to the usual Perceptron additive rule. On the other hand, if either the coin turns up tails or $\widehat{y}_t = y_t$ ($M_t = 0$ in Figure 1), then no update takes place.

The following theorem shows that our selective sampling Perceptron can achieve, in expectation, the same mistake bound as the standard Perceptron's, but using fewer labels.

Theorem 1 *If the algorithm of Figure 1 is run with input parameter $b > 0$ on a sequence (x_1, y_1), $(x_2, y_2), \ldots \in \mathbb{R}^d \times \{-1, +1\}$ of examples, then for all $n \geq 1$, all $u \in \mathbb{R}^d$, and all $\gamma > 0$,*

$$\mathbb{E}\left[\sum_{t=1}^{n} M_t\right] \leq \left(1 + \frac{X^2}{2b}\right) \frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{\|u\|^2 \left(2b + X^2\right)^2}{8\, b\, \gamma^2}$$

where $X = \max_{t=1,\ldots,n} \|x_t\|$. Furthermore, the expected number of labels queried by the algorithm equals $\sum_{t=1}^{n} \mathbb{E}\left[\frac{b}{b+|\widehat{p}_t|}\right]$.

The above bound depends on the choice of parameter b. In general, b might be viewed as a noise parameter ruling the extent to which a linear threshold model fits the data at hand. In principle, the optimal tuning of b is easily computed. Choosing

$$b = \frac{X^2}{2}\sqrt{1 + \frac{4\gamma^2}{||u||^2 X^2}\,\frac{\overline{L}_{\gamma,n}(u)}{\gamma}}$$

in Theorem 1 gives the following bound on the expected number of mistakes

$$\frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{\|u\|^2 X^2}{2\gamma^2} + \frac{\|u\|\, X}{\gamma}\sqrt{\frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{\|u\|^2 X^2}{4\gamma^2}}\,. \tag{1}$$

This is an expectation version of the mistake bound for the standard Perceptron algorithm (Freund and Schapire, 1999; Gentile, 2003; Gentile and Warmuth, 1999). Note that in the special case when the data are linearly separable with margin γ^* the optimal tuning simplifies to $b = X^2/2$ and yields the familiar Perceptron bound $\left(\|u\| X\right)^2/(\gamma^*)^2$. Hence, in the separable case, we obtain the somewhat counterintuitive result that the standard Perceptron bound is achieved by an algorithm whose label rate does not (directly) depend on how big the separation margin is.

As it turns out, (1) might be even sharper than its deterministic counterpart since, as already noted, $\overline{L}_{\gamma,n}(u)$ can be much smaller than $L_{\gamma,n}(u)$. However, since b is an input parameter of the selective sampling algorithm, the above setting implies that, at the beginning of the prediction process, the algorithm needs some extra information on the sequence of examples. In addition, unlike the bound of Theorem 1, which holds simultaneously for all γ and u, this refined bound can only be obtained for fixed choices of these quantities. Finally, observe that letting $b \to \infty$ in Figure 1 yields the standard Perceptron algorithm but, as a shortcoming, the corresponding bound in Theorem 1 gets vacuous. This is due to the fact that our simple proof produces a mistake bound where the constant ruling the (natural) trade-off between hinge loss term and margin term is directly related to the label sampling rate.

All of the above shortcomings will be fixed in Section 2.2, where we present an adaptive parameter version of the algorithm in Figure 1. Via a more involved analysis, we show that it is still possible to achieve a bound having the same form as (1) with no prior information.

That said, we are ready to prove Theorem 1.

Proof of Theorem 1. The proof extends the standard proof of the Perceptron mistake bound (see, e.g., Duda et al., 2000, Chap. 5) which is based on estimating the influence of an update on the distance $\|u-w_{t-1}\|^2$ between the current weight vector w_{t-1} and an arbitrary "target" hyperplane u. Our analysis uses a tighter estimate on this influence, and then uses a probabilistic analysis to turn this increased tightness into an expected saving on the number of observed labels. Since this probabilistic analysis only involves the terms that are brought about by the improved estimate, we are still able to recover (in expectation) the original Perceptron bound.

Fix an arbitrary sequence $(x_1,y_1),\ldots,(x_n,y_n)\in\mathbb{R}^d\times\{-1,+1\}$ of examples. Let t be an update trial, i.e., a trial such that $M_t Z_t = 1$. We can write

$$\begin{aligned}
\gamma-\ell_{\gamma,t}(u) &= \gamma-(\gamma-y_t\,u^\top x_t)_+ \\
&\le y_t\,u^\top x_t \\
&= y_t(u-w_{t-1}+w_{t-1})^\top x_t \\
&= y_t\,w_{t-1}^\top x_t+\frac{1}{2}\|u-w_{t-1}\|^2-\frac{1}{2}\|u-w_t\|^2+\frac{1}{2}\|w_{t-1}-w_t\|^2 \\
&= y_t\,\widehat{p}_t+\frac{1}{2}\|u-w_{t-1}\|^2-\frac{1}{2}\|u-w_t\|^2+\frac{1}{2}\|w_{t-1}-w_t\|^2\ .
\end{aligned}$$

Since the above inequality holds for any $\gamma>0$ and any $u\in\mathbb{R}^d$, we can replace γ by $\alpha\gamma$ and u by αu, where α is a constant to be optimized.

Rearranging, and using $y_t\,\widehat{p}_t\le 0$ implied by $M_t=1$, yields

$$\alpha\gamma+|\widehat{p}_t|\le\alpha\ell_{\gamma,t}(u)+\frac{1}{2}\|\alpha u-w_{t-1}\|^2-\frac{1}{2}\|\alpha u-w_t\|^2+\frac{1}{2}\|w_{t-1}-w_t\|^2\ .$$

Note that, instead of discarding the term $|\widehat{p}_t|$, as in the original Perceptron proof, we keep it around. This yields a stronger inequality which, as we will see, is the key to achieving our final result.

If t is such that $M_t Z_t = 0$ then no update occurs and $w_t = w_{t-1}$. Hence we conclude that, for any trial t,

$$\begin{aligned}
M_t Z_t(\alpha\gamma+|\widehat{p}_t|) \le{}& M_t Z_t\,\alpha\,\ell_{\gamma,t}(u) \\
&+\frac{1}{2}\|\alpha u-w_{t-1}\|^2-\frac{1}{2}\|\alpha u-w_t\|^2+\frac{M_t Z_t}{2}\|w_{t-1}-w_t\|^2\ . \qquad (2)
\end{aligned}$$

We now sum the above over t, use $\|w_{t-1}-w_t\|^2\le X^2$ and recall that $w_0=0$. We get

$$\sum_{t=1}^n M_t Z_t\left(\alpha\gamma+|\widehat{p}_t|-\frac{X^2}{2}\right)\le\alpha\sum_{t=1}^n M_t Z_t\,\ell_{\gamma,t}(u)+\frac{\alpha^2}{2}\|u\|^2\ .$$

Now choose $\alpha=(2b+X^2)/(2\gamma)$, where $b>0$ is the algorithm's parameter. The above then becomes

$$\sum_{t=1}^n M_t Z_t\left(b+|\widehat{p}_t|\right)\le\frac{2b+X^2}{2\gamma}\sum_{t=1}^n M_t Z_t\,\ell_{\gamma,t}(u)+\frac{\|u\|^2\left(2b+X^2\right)^2}{8\gamma^2}\ . \qquad (3)$$

A similar inequality is also obtained in the analysis of the standard Perceptron algorithm. Here, however, we have added the random variable Z_t, associated with the selective sampling, and kept the term $|\widehat{p}_t|$. Note that this term also appears in the conditional expectation of Z_t, since we have defined $\mathbb{E}_{t-1} Z_t$ as $b/(b+|\widehat{p}_t|)$. This fact is exploited now, when we take expectations on both sides of (3). On the left-hand side we obtain

$$\mathbb{E}\left[\sum_{t=1}^{n} M_t Z_t \left(b+|\widehat{p}_t|\right)\right] = \mathbb{E}\left[\sum_{t=1}^{n} M_t \left(b+|\widehat{p}_t|\right) \mathbb{E}_{t-1} Z_t\right] = \mathbb{E}\left[\sum_{t=1}^{n} b M_t\right] ,$$

where the first equality is proven by observing that M_t and $\widehat{p}_t$ are determined by $Z_1, \ldots, Z_{t-1}$ (that is, they are both measurable with respect to the σ-algebra generated by $Z_1, \ldots, Z_{t-1}$). Dividing by b we obtain the claimed inequality on the expected number of mistakes.

The value of $\mathbb{E}\left[\sum_{t=1}^{n} Z_t\right]$ (the expected number of queried labels) trivially follows from

$$\mathbb{E}\left[\sum_{t=1}^{n} Z_t\right] = \mathbb{E}\left[\sum_{t=1}^{n} \mathbb{E}_{t-1} Z_t\right] .$$

This concludes the proof. ■

2.2 Selective Sampling Perceptron: Adaptive Version

In this section we show how to learn the best trade-off parameter b in an on-line fashion. Our goal is to devise a time-changing expression for this parameter that achieves a bound on the expected number of mistakes having the same form as (1)—i.e., with constant 1 in front of the cumulative hinge loss term—but relying on no prior knowledge whatsoever on the sequence of examples.

We follow the "self-confident" approach introduced by Auer et al. (2002) and Gentile (2001) though, as pointed out later, our self-confidence tuning here is technically different, since it does not rely on projections to control the norm of the weight (as in, e.g., Herbster and Warmuth, 2001; Auer et al., 2002; Gentile, 2001, 2003).

Our adaptive version of the selective sampling Perceptron algorithm is described in Figure 2. The algorithm still has a parameter $\beta > 0$ but, as we will see, any constant value for β leads to bounds of the form (1). Thus β has far less influence on the final bound than the b parameter in Figure 1.

The adaptive algorithm is essentially the same as the one in Figure 1, but for maintaining two further variables, X_t and K_t. At the *end* of trial t, variable X_t stores the maximal norm of the instance vectors involved in updates up to and including time t, while K_t just counts the number of such updates. Observe that b_t increases with (the square root of) this number, thereby implementing the easy intuition that the more updates are made by the algorithm the harder the problem looks, and the more labels are needed on average. However the reader should not conclude from this observation that the label rate $b_{t-1}/(b_{t-1}+|\widehat{p}_t|)$ converges to 1 as $t \to \infty$, since b_t does not scale with time t but with the number of *updates* made up to time t, which can be far smaller than t. At the same time, the margin $|\widehat{p}_t|$ might have an erratic behavior whose oscillations can also grow with the number of updates.

We have the following result.

Selective sampling Perceptron with adaptive parameter.
Parameters: $\beta > 0$.
Initialization: $w_0 = (0,\ldots,0)^\top$, $X_0 = 0$, $K_0 = 0$.
For each trial $t = 1,2,\ldots$

(1) observe an instance vector $x_t \in \mathbb{R}^d$, and set $\widehat{p}_t = w_{t-1}^\top x_t$;

(2) predict with $\widehat{y}_t = \mathrm{SGN}(\widehat{p}_t)$;

(3) set $X' = \max\{X_{t-1}, \|x_t\|\}$;

(4) draw a Bernoulli random variable $Z_t \in \{0,1\}$ of parameter

$$\frac{b_{t-1}}{b_{t-1} + |\widehat{p}_t|} \qquad \text{where} \qquad b_{t-1} = \beta\,(X')^2\,\sqrt{1+K_{t-1}}\,;$$

(5) **if** $Z_t = 1$ **then**

(a) query label $y_t \in \{-1,+1\}$,

(b) if $\widehat{y}_t \neq y_t$ (i.e., $M_t = 1$) then update

$$\begin{aligned} w_t &= w_{t-1} + y_t\,x_t \\ K_t &= K_{t-1} + 1 \\ X_t &= X'; \end{aligned}$$

(6) **else** ($Z_t = 0$) set $w_t = w_{t-1}$, $K_t = K_{t-1}$, $X_t = X_{t-1}$.

Figure 2: Adaptive parameter version of the selective sampling Perceptron algorithm.

Theorem 2 *If the algorithm of Figure 2 is run with input parameter $\beta > 0$ on a sequence (x_1,y_1), $(x_2,y_2)\ldots \in \mathbb{R}^d \times \{-1,+1\}$ of examples, then for all $n \geq 1$, all $u \in \mathbb{R}^d$, and all $\gamma > 0$,*

$$\mathbb{E}\left[\sum_{t=1}^{n} M_t\right] \leq \frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{R}{2\beta} + \frac{B^2}{2} + B\sqrt{\frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{R}{2\beta} + \frac{B^2}{4}}$$

where

$$B = R + \frac{1+3R/2}{\beta} \qquad \text{and} \qquad R = \frac{\|u\|\left(\max_{t=1,\ldots,n}\|x_t\|\right)}{\gamma}\,.$$

Moreover, the expected number of labels queried by the algorithm equals $\sum_{t=1}^{n}\mathbb{E}\left[\frac{b_{t-1}}{b_{t-1}+|\widehat{p}_t|}\right]$.

Before delving into the proof, it is worth observing the role of parameter β. As we have already said, if we set β to any constant value (no matter how small), we obtain a bound of the form (1). On the other hand, for $\beta \to \infty$ the algorithm reduces to the classical Perceptron algorithm, and the

bound (unlike the one in Theorem 1) becomes the Perceptron bound, as given by Gentile (2003). Clearly, the larger is β the more labels are queried on average over the trials. Thus β has also an indirect influence on the hinge loss term $\overline{L}_{\gamma,n}(u)$. In particular, we might expect that a small value of β makes the number of updates shrink (note that in the limit when $\beta \to 0$ this number goes to 0).

Proof of Theorem 2. Fix an arbitrary sequence $(x_1,y_1),\ldots,(x_n,y_n) \in \mathbb{R}^d \times \{-1,+1\}$ of examples and let $X = \max_{t=1,\ldots,n} \|x_t\|$. The proof is a more involved version of the proof of Theorem 1. We start from the one-trial equation (2) established there, where we replace the (constant) stretching factor α by the time-varying factor c_{t-1}/γ, where $c_{t-1} \geq 0$ will be set later and $\gamma > 0$ is the free margin parameter of the hinge loss. This yields

$$\begin{aligned} M_t Z_t (c_{t-1} + |\widehat{p}_t|) \leq M_t Z_t \frac{c_{t-1}}{\gamma} \ell_{\gamma,t}(u) \\ + \frac{1}{2}\left\| \frac{c_{t-1}}{\gamma} u - w_{t-1} \right\|^2 - \frac{1}{2}\left\| \frac{c_{t-1}}{\gamma} u - w_t \right\|^2 + \frac{M_t Z_t}{2} \|w_{t-1} - w_t\|^2 \ . \end{aligned}$$

From the update rule in Figure 2 we have $(M_t Z_t/2)\|w_{t-1} - w_t\|^2 \leq (M_t Z_t/2)\|x_t\|^2$. We rearrange and divide by b_{t-1}. This yields

$$\begin{aligned} M_t Z_t \left(\frac{c_{t-1} + |\widehat{p}_t| - \|x_t\|^2/2}{b_{t-1}} \right) \leq M_t Z_t \frac{c_{t-1}}{b_{t-1}} \frac{\ell_{\gamma,t}(u)}{\gamma} \\ + \frac{1}{2\,b_{t-1}} \left(\left\| \frac{c_{t-1}}{\gamma} u - w_{t-1} \right\|^2 - \left\| \frac{c_{t-1}}{\gamma} u - w_t \right\|^2 \right) \ . \end{aligned} \tag{4}$$

We now transform the difference of squared norms in (4) into a pair of telescoping differences,

$$\begin{aligned} \frac{1}{2\,b_{t-1}} \left(\left\| \frac{c_{t-1}}{\gamma} u - w_{t-1} \right\|^2 - \left\| \frac{c_{t-1}}{\gamma} u - w_t \right\|^2 \right) \\ = \frac{1}{2\,b_{t-1}} \left\| \frac{c_{t-1}}{\gamma} u - w_{t-1} \right\|^2 - \frac{1}{2\,b_t} \left\| \frac{c_t}{\gamma} u - w_t \right\|^2 \\ + \frac{1}{2\,b_t} \left\| \frac{c_t}{\gamma} u - w_t \right\|^2 - \frac{1}{2\,b_{t-1}} \left\| \frac{c_{t-1}}{\gamma} u - w_t \right\|^2 \ . \end{aligned} \tag{5}$$

If we set

$$c_{t-1} = \frac{1}{2}\big(\max\{X_{t-1}, \|x_t\|\}\big)^2 + b_{t-1}$$

we can expand the difference of norms (5) as follows

$$\begin{aligned} (5) &= \frac{\|u\|^2}{2\gamma^2} \left(\frac{c_t^2}{b_t} - \frac{c_{t-1}^2}{b_{t-1}} \right) + \frac{u^\top w_t}{\gamma} \left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t} \right) + \frac{\|w_t\|^2}{2} \left(\frac{1}{b_t} - \frac{1}{b_{t-1}} \right) \\ &\leq \frac{\|u\|^2}{2\gamma^2} \left(\frac{c_t^2}{b_t} - \frac{c_{t-1}^2}{b_{t-1}} \right) + \frac{\|u\|\,\|w_t\|}{\gamma} \left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t} \right) \end{aligned} \tag{6}$$

where in the last step we used $b_t \geq b_{t-1}$ and the inequality

$$\frac{c_{t-1}}{b_{t-1}} \geq \frac{c_t}{b_t}$$

which follows from $c_{t-1}/b_{t-1} = 1/\left(2\beta\sqrt{1+K_{t-1}}\right)+1$.

Recall now the standard way of bounding the norm of a Perceptron weight vector in terms of the number of updates,

$$\begin{aligned}\|w_t\|^2 &= \|w_{t-1}\|^2 + M_t Z_t y_t w_{t-1}^\top x_t + M_t Z_t \|x_t\|^2 \\ &\le \|w_{t-1}\|^2 + M_t Z_t \|x_t\|^2 \\ &\le \|w_{t-1}\|^2 + M_t Z_t X^2\end{aligned}$$

which, combined with $w_0 = 0$, implies

$$\|w_t\| \le X\sqrt{K_t} \qquad \text{for any } t. \tag{7}$$

Applying inequality (7) to (6) yields

$$(5) \le \frac{\|u\|^2}{2\gamma^2}\left(\frac{c_t^2}{b_t} - \frac{c_{t-1}^2}{b_{t-1}}\right) + \frac{\|u\|\,X\sqrt{K_t}}{\gamma}\left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t}\right). \tag{8}$$

We continue by bounding from above the last term in (8). If t is such that $M_t Z_t = 1$ we have $K_t = K_{t-1}+1$. Thus we can write

$$\begin{aligned}\sqrt{K_t}\left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t}\right) &= \frac{\sqrt{K_t}}{2\beta}\left(\frac{1}{\sqrt{1+K_{t-1}}} - \frac{1}{\sqrt{1+K_t}}\right) \\ &= \frac{\sqrt{K_t}}{2\beta}\left(\frac{1}{\sqrt{K_t}} - \frac{1}{\sqrt{1+K_t}}\right) \\ &= \frac{1}{2\beta}\,\frac{\sqrt{1+K_t}-\sqrt{K_t}}{\sqrt{1+K_t}} \\ &\le \frac{1}{4\beta}\,\frac{1}{\sqrt{K_t}\sqrt{1+K_t}} \\ &\quad \text{(using } \sqrt{1+x}-\sqrt{x} \le \tfrac{1}{2\sqrt{x}}\text{)} \\ &\le \frac{1}{4\beta}\,\frac{1}{K_t}\,.\end{aligned}$$

On the other hand, if $M_t Z_t = 0$ we have $b_t = b_{t-1}$ and $c_t = c_{t-1}$. Hence, for any trial t we obtain

$$\sqrt{K_t}\left(\frac{c_{t-1}}{b_{t-1}} - \frac{c_t}{b_t}\right) \le \frac{M_t Z_t}{4\beta}\,\frac{1}{K_t}\,.$$

Putting together as in (4) and using $c_{t-1} - \|x_t\|^2/2 \ge b_{t-1}$ on the left-hand side yields

$$\begin{aligned}M_t Z_t\left(\frac{b_{t-1}+|\widehat{p}_t|}{b_{t-1}}\right) \le\; & M_t Z_t\,\frac{c_{t-1}}{b_{t-1}}\,\frac{\ell_{\gamma,t}(u)}{\gamma} \\ & + \frac{1}{2b_{t-1}}\left\|\frac{c_{t-1}}{\gamma}u - w_{t-1}\right\|^2 - \frac{1}{2b_t}\left\|\frac{c_t}{\gamma}u - w_t\right\|^2 \\ & + \frac{\|u\|^2}{2\gamma^2}\left(\frac{c_t^2}{b_t} - \frac{c_{t-1}^2}{b_{t-1}}\right) + \frac{\|u\|\,X}{\gamma}\,\frac{M_t Z_t}{4\beta}\,\frac{1}{K_t}\end{aligned}$$

holding for any trial t, any $u \in \mathbb{R}^d$, and any $\gamma > 0$.

Now, as in the proof of Theorem 1, we sum over $t = 1, \ldots, n$, use $w_0 = 0$, and simplify

$$\sum_{t=1}^{n} M_t Z_t \left(\frac{b_{t-1} + |\widehat{p}_t|}{b_{t-1}} \right) \leq \frac{1}{\gamma} \sum_{t=1}^{n} M_t Z_t \frac{c_{t-1}}{b_{t-1}} \ell_{\gamma,t}(u) \tag{9}$$

$$+ \frac{c_n^2}{b_n} \frac{\|u\|^2}{2\gamma^2} - \frac{1}{2\,b_n} \left\| \frac{c_n}{\gamma} u - w_n \right\|^2 \tag{10}$$

$$+ \frac{1}{4\beta} \frac{\|u\| X}{\gamma} \sum_{t=1}^{n} \frac{M_t Z_t}{K_t} \,.$$

We now proceed by bounding separately the terms in the right-hand side of the above inequality. For (9) we get

$$\begin{aligned} \frac{1}{\gamma} \frac{c_{t-1}}{b_{t-1}} \ell_{\gamma,t}(u) &= \frac{1}{\gamma} \left(\frac{1}{2\beta\sqrt{1+K_{t-1}}} + 1 \right) \ell_{\gamma,t}(u) \\ &\leq \frac{1}{\gamma} \frac{1}{2\beta\sqrt{1+K_{t-1}}} \big(\gamma + \|u\| X\big) + \frac{\ell_{\gamma,t}(u)}{\gamma} \\ &\quad \text{(since } \ell_{\gamma,t}(u) \leq \gamma + \|u\| X\text{)} \\ &= \frac{1}{2\beta} \left(1 + \frac{\|u\| X}{\gamma} \right) \frac{1}{\sqrt{1+K_{t-1}}} + \frac{\ell_{\gamma,t}(u)}{\gamma} \,. \end{aligned}$$

For (10) we obtain

$$\begin{aligned} \frac{c_n^2}{b_n} \frac{\|u\|^2}{2\gamma^2} - \frac{1}{2b_n} \left\| \frac{c_n}{\gamma} u - w_n \right\|^2 &= \frac{c_n}{b_n} \frac{u^\top w_n}{\gamma} - \frac{\|w_n\|^2}{2b_n} \\ &\leq \frac{c_n}{b_n} \frac{u^\top w_n}{\gamma} \\ &\leq \frac{c_n}{b_n} \frac{\|u\| \, \|w_n\|}{\gamma} \\ &\leq \left(1 + \frac{1}{2\beta\sqrt{1+K_n}} \right) \frac{\|u\| \, X \sqrt{K_n}}{\gamma} \end{aligned}$$

where in the last step we used (7). Using these expressions to bound the left-hand side of (9) yields

$$\sum_{t=1}^{n} M_t Z_t \left(\frac{b_{t-1} + |\widehat{p}_t|}{b_{t-1}} \right)$$

$$\leq \frac{1}{\gamma} \sum_{t=1}^{n} M_t Z_t \, \ell_{\gamma,t}(u) + \frac{1}{2\beta} \left(1 + \frac{\|u\| X}{\gamma} \right) \sum_{t=1}^{n} \frac{M_t Z_t}{\sqrt{1+K_{t-1}}} \tag{11}$$

$$+ \left(1 + \frac{1}{2\beta\sqrt{1+K_n}} \right) \frac{\|u\| X \sqrt{K_n}}{\gamma} + \frac{1}{4\beta} \frac{\|u\| X}{\gamma} \sum_{t=1}^{n} \frac{M_t Z_t}{K_t} \,. \tag{12}$$

Next, we focus on the second sum in (11) and the sum in (12). Since $M_t Z_t = 1$ implies $K_t = K_{t-1} + 1$ we can write

$$\sum_{t=1}^{n} \frac{M_t Z_t}{\sqrt{1+K_{t-1}}} = \sum_{t : M_t Z_t = 1} \frac{1}{\sqrt{K_t}} = \sum_{t=1}^{K_n} \frac{1}{\sqrt{t}} \leq 2\sqrt{K_n} \,.$$

Similarly for the other sum, but using a more crude bound,

$$\sum_{t=1}^{n} \frac{M_t Z_t}{K_t} = \sum_{t\,:\,M_t Z_t=1} \frac{1}{K_t} \le \sum_{t\,:\,M_t Z_t=1} \frac{1}{\sqrt{K_t}} \le 2\sqrt{K_n}\,.$$

Recalling the short-hand $R = (\|u\|X)/\gamma$, we apply these bounds to (11) and (12). After a simple overapproximation this gives

$$\sum_{t=1}^{n} M_t Z_t \left(\frac{b_{t-1}+|\widehat{p}_t|}{b_{t-1}}\right) \le \frac{1}{\gamma}\sum_{t=1}^{n} M_t Z_t \,\ell_{\gamma,t}(u) + \sqrt{K_n}\left(R + \frac{1+3R/2}{\beta}\right) + \frac{R}{2\beta}\,.$$

We are now ready to take expectations on both sides. As in the proof of Theorem 1, since $\mathbb{E}_{t-1} Z_t = \frac{b_{t-1}}{b_{t-1}+|\widehat{p}_t|}$ and both M_t and b_{t-1} are measurable with respect to the σ-algebra generated by $Z_1,\ldots,Z_{t-1}$, we obtain

$$\mathbb{E}\left[\sum_{t=1}^{n} M_t Z_t \left(\frac{b_{t-1}+|\widehat{p}_t|}{b_{t-1}}\right)\right] = \mathbb{E}\left[\sum_{t=1}^{n} M_t\right]\,.$$

In taking the expectation on the right-hand side, we first bound $K_n = \sum_{t=1}^{n} M_t Z_t$ as $K_n \le \sum_{t=1}^{n} M_t$, then exploit the concavity of the square root. This results in

$$\sum_{t=1}^{n} \mathbb{E} M_t \le \frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \left(R + \frac{1+3R/2}{\beta}\right)\sqrt{\sum_{t=1}^{n} \mathbb{E} M_t} + \frac{R}{2\beta}\,.$$

Solving the above inequality for $\sum_{t=1}^{n} \mathbb{E} M_t$ gives the stated bound on the expected number of mistakes.

Finally, as in the proof of Theorem 1, the expected number of labels queried by the algorithm trivially follows from

$$\mathbb{E}\left[\sum_{t=1}^{n} Z_t\right] = \mathbb{E}\left[\sum_{t=1}^{n} \mathbb{E}_{t-1} Z_t\right]$$

concluding the proof. ■

The proof of Theorem 2 is reminiscent of the analysis of the "self-confident" dynamical tuning used in Auer et al. (2002) and Gentile (2001). In those papers, however, the variable learning rate was combined with a re-normalization step of the weight. Here we use a different technique based on a time-changing stretching factor $\alpha_{t-1} = c_{t-1}/\gamma$ for the comparison vector u. This alternative approach is made possible by the boundedness of the hinge loss terms, as shown by the inequality $\ell_{\gamma,t}(u) \le \gamma + \|u\|X$.

2.3 Selective Sampling Second-Order Perceptron

We now consider a selective sampling version of the second-order Perceptron algorithm introduced by Cesa-Bianchi et al. (2005). The second-order Perceptron algorithm might be seen as running the standard (first-order) Perceptron algorithm as a subroutine. Let v_{t-1} denote the weight vector computed by the standard Perceptron algorithm. In trial t, instead of using the sign of $v_{t-1}^{\top} x_t$ to predict the current instance x_t, the second-order algorithm predicts through the sign of the margin

$$\widehat{p}_t = \left(M^{-1/2} v_{t-1}\right)^{\top}\left(M^{-1/2} x_t\right) = v_{t-1}^{\top} M^{-1} x_t\,.$$

Here $M = I + \sum_s x_s x_s^\top + x_t x_t^\top$ is a (full-rank) positive definite matrix, where I is the $d \times d$ identity matrix, and the sum $\sum_s x_s x_s^\top$ runs over the mistaken trials s up to time $t-1$. If, when using the above prediction rule, the algorithm makes a mistake in trial t, then v_{t-1} is updated according to the standard Perceptron rule and t is included in the set of mistaken trials. Hence the second-order algorithm differs from the standard Perceptron algorithm in that, before each prediction, a linear transformation $M^{-1/2}$ is applied to both the current Perceptron weight v_{t-1} and the current instance x_t. This linear transformation depends on the correlation matrix defined over mistaken instances, including the current one. As explained in Cesa-Bianchi et al. (2005), this linear transformation has the effect of reducing the number of mistakes whenever the instance correlation matrix $\sum_s x_s x_s^\top + x_t x_t^\top$ has a spectral structure that causes an eigenvector with small eigenvalue to correlate well with a good linear approximator u of the entire data sequence. In such situations, the mistake bound of the second-order Perceptron algorithm can be shown to be significantly better than the one for the first-order algorithm.

In what follows, we use A_{t-1} to denote $I + \sum_s x_s x_s^\top$ where the sum ranges over the mistaken trials between trial 1 and trial $t-1$. We derive a selective sampling version of the second-order algorithm in much the same way as we did for the standard Perceptron algorithm: The selective sampling second-order Perceptron algorithm predicts and then decides whether to ask for the label y_t using the same randomized rule as the one in Figure 1. In Figure 3 we provide a pseudo-code description and introduce the notation used in the analysis.

The analysis follows the same pattern as the proof of Theorem 1. A key step is a one-trial progress equation developed by Forster (1999) for a regression framework. See also Azoury and Warmuth (2001). As before, the comparison between the second-order Perceptron's bound and the one contained in Theorem 3 reveals that the selective sampling algorithm can achieve, in expectation, the same mistake bound using fewer labels.

Theorem 3 *If the algorithm of Figure 3 is run with parameter $b > 0$ on a sequence $(x_1, y_1), (x_2, y_2), \ldots \in \mathbb{R}^d \times \{-1, +1\}$ of examples, then for all $n \geq 1$, all $u \in \mathbb{R}^d$, and all $\gamma > 0$,*

$$\mathbb{E}\left[\sum_{t=1}^{n} M_t\right] \leq \frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{b}{2\gamma^2} u^\top \mathbb{E}[A_n] u + \frac{1}{2b} \sum_{i=1}^{d} \mathbb{E} \ln(1 + \lambda_i)$$

where $\lambda_1, \ldots, \lambda_d$ are the eigenvalues of the (random) correlation matrix $\sum_{t=1}^{n} M_t Z_t x_t x_t^\top$ and $A_n = I + \sum_{t=1}^{n} M_t Z_t x_t x_t^\top$ (thus $1 + \lambda_i$ is the i-th eigenvalue of A_n). Moreover, the expected number of labels queried by the algorithm equals $\sum_{t=1}^{n} \mathbb{E}\left[\frac{b}{b + |\widehat{p}_t|}\right]$.

Again, the above bound depends on the algorithm's parameter b. Setting

$$b = \gamma \sqrt{\frac{\sum_{i=1}^{d} \mathbb{E} \ln (1 + \lambda_i)}{u^\top \mathbb{E}\,[A_n]\, u}}$$

in Theorem 3 we are led to the bound

$$\mathbb{E}\left[\sum_{t=1}^{n} M_t\right] \leq \frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{1}{\gamma} \sqrt{(u^\top \mathbb{E}\,[A_n]\, u) \sum_{i=1}^{d} \mathbb{E} \ln (1 + \lambda_i)}\,. \tag{13}$$

This is an expectation version of the mistake bound for the (deterministic) second-order Perceptron algorithm, as proven by Cesa-Bianchi et al. (2005). As for the first-order algorithms, this

Selective sampling second-order Perceptron.
Parameter: $b > 0$.
Initialization: $A_0 = I$, $v_0 = (0,\ldots,0)^\top$.
For each trial $t = 1,2,\ldots$

(1) observe an instance vector $x_t \in \mathbb{R}^d$, and set $\widehat{p}_t = v_{t-1}^\top (A_{t-1} + x_t x_t^\top)^{-1} x_t$;

(2) predict with $\widehat{y}_t = \mathrm{SGN}(\widehat{p}_t)$;

(3) draw a Bernoulli random variable $Z_t \in \{0,1\}$ of parameter $\dfrac{b}{b+|\widehat{p}_t|}$;

(4) **if** $Z_t = 1$ **then** query label $y_t \in \{-1,+1\}$ and perform the standard second-order Perceptron update:

$$\begin{aligned} v_t &= v_{t-1} + M_t y_t x_t \ , \\ A_t &= A_{t-1} + M_t x_t x_t^\top \ ; \end{aligned}$$

(5) **else** ($Z_t = 0$) set $v_t = v_{t-1}$ and $A_t = A_{t-1}$.

Figure 3: A selective sampling version of the second-order Perceptron algorithm.

bound might be sharper than its deterministic counterpart, since the magnitude of the three quantities $\overline{L}_{\gamma,n}(u)$, $u^\top \mathbb{E}[A_n] u$, and $\sum_{i=1}^d \mathbb{E}\ln(1+\lambda_i)$ is ruled by the size of the random set of updates $\{t : M_t Z_t = 1\}$, which is typically smaller than the set of mistaken trials of the deterministic algorithm.

However, as for the algorithm in Figure 1, this parameter tuning turns out to be unrealistic, since it requires preliminary information on the structure of the sequence of examples. Unlike the first-order algorithm, we have been unable to devise a meaningful adaptive parameter version for the algorithm in Figure 3.

Proof of Theorem 3. The proof proceeds along the same lines as the proof of Theorem 1, thus we only emphasize the main differences. In addition to the notation given there, we define Φ_t to be the (random) function

$$\Phi_t(u) = \frac{1}{2}\|u\|^2 + \sum_{s=1}^{t} \frac{M_s Z_s}{2} (y_s - u^\top x_s)^2 \ .$$

The quantity $\Phi_t(u)$, which is the regularized cumulative square loss of u on the past mistaken trials, plays a key role in the proof. Indeed, we now show that the algorithm incurs on each mistaken trial a square loss $\left(y_t - \widehat{p}_t\right)^2$ bounded by the difference $\inf_v \Phi_{t+1}(v) - \inf_v \Phi_t(v)$ plus a quadratic term involving A_t^{-1}. When we sum over mistaken trials, the difference telescopes and the sum of quadratic terms can be bounded using known results. Finally, the margin we use in the probabilistic analysis is obtained as cross-term when the square loss is expanded.

When trial t is such that $M_t Z_t = 1$ we can exploit a result proven by Forster (1999) for linear regression (proof of Theorem 3 therein), where it is essentially shown that choosing $\widehat{p}_t = v_{t-1}^\top (A_{t-1} + x_t x_t^\top)^{-1} x_t$ (as in Figure 3) yields

$$\frac{1}{2}\left(\widehat{p}_t - y_t\right)^2 = \inf_v \Phi_{t+1}(v) - \inf_v \Phi_t(v) + \frac{1}{2} x_t^\top A_t^{-1} x_t - \frac{1}{2}\left(x_t^\top A_{t-1}^{-1} x_t\right) \widehat{p}_t^2 \,.$$

On the other hand, if trial t is such that $M_t Z_t = 0$ we have $\inf_{v \in \mathbb{R}^d} \Phi_{t+1}(v) = \inf_{v \in \mathbb{R}^d} \Phi_t(v)$. Hence the equality

$$\frac{M_t Z_t}{2}\left(\widehat{p}_t - y_t\right)^2 = \inf_v \Phi_{t+1}(v) - \inf_v \Phi_t(v) + \frac{M_t Z_t}{2} x_t^\top A_t^{-1} x_t - \frac{M_t Z_t}{2}\left(x_t^\top A_{t-1}^{-1} x_t\right) \widehat{p}_t^2$$

holds for all trials t. We drop the term $-M_t Z_t \left(x_t^\top A_{t-1}^{-1} x_t\right) \widehat{p}_t^2/2$, which is nonpositive (since A_{t-1} is positive definite), and sum over $t = 1, \ldots, n$. Observing that $\inf_v \Phi_1(v) = 0$, we obtain

$$\begin{aligned}
\frac{1}{2}\sum_{t=1}^n M_t Z_t \left(\widehat{p}_t - y_t\right)^2 &\leq \inf_v \Phi_{n+1}(v) - \inf_v \Phi_1(v) + \frac{1}{2}\sum_{t=1}^n M_t Z_t x_t^\top A_t^{-1} x_t \\
&\leq \Phi_{n+1}(u) + \frac{1}{2}\sum_{t=1}^n M_t Z_t x_t^\top A_t^{-1} x_t \\
&\leq \frac{1}{2}\|u\|^2 + \frac{1}{2}\sum_{t=1}^n M_t Z_t \left(u^\top x_t - y_t\right)^2 + \frac{1}{2}\sum_{t=1}^n M_t Z_t x_t^\top A_t^{-1} x_t
\end{aligned}$$

holding for any $u \in \mathbb{R}^d$.

Expanding the squares and performing trivial simplifications we arrive at the following inequality

$$\begin{aligned}
&\frac{1}{2}\sum_{t=1}^n M_t Z_t \left(\widehat{p}_t^2 - 2 y_t \widehat{p}_t\right) \\
&\quad\leq \frac{1}{2}\left[\|u\|^2 + \sum_{t=1}^n M_t Z_t \left(u^\top x_t\right)^2\right] - \sum_{t=1}^n M_t Z_t y_t u^\top x_t + \frac{1}{2}\sum_{t=1}^n M_t Z_t x_t^\top A_t^{-1} x_t \,. \qquad (14)
\end{aligned}$$

We focus on the right-hand side of (14). We rewrite the first term and bound from above the last term. For the first term we have

$$\frac{1}{2}\left[\|u\|^2 + \sum_{t=1}^n M_t Z_t \left(u^\top x_t\right)^2\right] = \frac{1}{2} u^\top \left(I + \sum_{t=1}^n x_t x_t^\top M_t Z_t\right) u = \frac{1}{2} u^\top A_n u \,. \qquad (15)$$

For the third term, we use a property of the inverse matrices A_t^{-1} (see, e.g., Lai and Wei, 1982; Azoury and Warmuth, 2001; Forster, 1999; Cesa-Bianchi et al., 2005),

$$\begin{aligned}
\frac{1}{2}\sum_{t=1}^{n} M_t Z_t x_t^\top A_t^{-1} x_t &= \frac{1}{2}\sum_{t=1}^{n}\left(1-\frac{|A_{t-1}|}{|A_t|}\right) \\
&\le \frac{1}{2}\sum_{t=1}^{n}\ln\frac{|A_t|}{|A_{t-1}|} \\
&= \frac{1}{2}\ln\frac{|A_n|}{|A_0|} \\
&= \frac{1}{2}\ln|A_n| \\
&= \frac{1}{2}\sum_{i=1}^{d}\ln(1+\lambda_i)
\end{aligned}$$

where we recall that $1+\lambda_i$ is the i-th eigenvalue of A_n.

Replacing back, observing that $-y_t\widehat{p}_t \le 0$ whenever $M_t = 1$, dropping the term involving $\widehat{p}_t^2$, and rearranging yields

$$\sum_{t=1}^{n} M_t Z_t\left(|\widehat{p}_t| + y_t u^\top x_t\right) \le \frac{1}{2}u^\top A_n u + \frac{1}{2}\sum_{i=1}^{d}\ln(1+\lambda_i)\ .$$

At this point, as in the proof of Theorem 1, we introduce hinge loss terms and stretch the comparison vector u to $\frac{b}{\gamma}u$, where b is the algorithm's parameter. We obtain

$$\sum_{t=1}^{n} M_t Z_t\left(|\widehat{p}_t| + b\right) \le \frac{b}{\gamma}\sum_{t=1}^{n} M_t Z_t\,\ell_{\gamma,t}(u) + \frac{b^2}{2\gamma^2}u^\top A_n u + \frac{1}{2}\sum_{i=1}^{d}\ln(1+\lambda_i)\ .$$

We take expectations on both sides. Recalling that $\mathbb{E}_{t-1} Z_t = b/(b+|\widehat{p}_t|)$, and proceeding similarly to the proof of Theorem 1 we get the claimed bounds on $\sum_{t=1}^{n}\mathbb{E}\,M_t$ and $\sum_{t=1}^{n}\mathbb{E}\,Z_t$. ■

3. Selective Sampling Winnow

The techniques used to prove Theorem 1 can be readily extended to analyze selective sampling versions of algorithms in the general additive family of Grove et al. (2001), Warmuth and Jagota (1997), and Kivinen and Warmuth (2001). The algorithms in this family—which includes Winnow (Littlestone, 1988), the p-norm Perceptron (Grove et al., 2001; Gentile, 2001), and others—are parametrized by a strictly convex and differentiable *potential function* $\Psi : \mathbb{R}^d \to \mathbb{R}$ obeying some additional regularity properties. We now show a concrete example by analyzing the selective sampling version of the Winnow algorithm (Littlestone, 1988), a member of the general additive family based on the exponential potential $\Psi(u) = e^{u_1} + \cdots + e^{u_d}$.

In its basic version, Winnow uses weights that belong to the probability simplex in $\mathbb{R}^d$. The update rule for the weights is multiplicative, and is followed by a normalization step which projects the updated weight vector back to the simplex. Introducing the intermediate weight w'_t, we define

Selective sampling Winnow.
Parameters: $\eta, b > 0$.
Initialization: $w_0 = (1/d, \ldots, 1/d)^\top$.
For each trial $t = 1, 2, \ldots$

(1) observe an instance vector $x_t \in \mathbb{R}^d$, and set $\widehat{p}_t = w_{t-1}^\top x_t$;

(2) predict with $\widehat{y}_t = \mathrm{SGN}(\widehat{p}_t)$;

(3) draw a Bernoulli random variable $Z_t \in \{0,1\}$ of parameter $\dfrac{b}{b+|\widehat{p}_t|}$;

(4) **if** $Z_t = 1$ **then** query label $y_t \in \{-1,+1\}$ and perform the standard Winnow update:

$$\begin{aligned} w'_{i,t} &= w_{i,t-1}\, e^{M_t \eta y_t x_{i,t}}\,, \\ w_{i,t} &= w'_{i,t}/(w'_{1,t} + \cdots + w'_{d,t}) \qquad i = 1, \ldots, d; \end{aligned}$$

(5) **else** ($Z_t = 0$) set $w_t = w_{t-1}$.

Figure 4: A selective sampling version of the Winnow algorithm.

the update rule as follows:

$$\begin{aligned} w'_{i,t} &= w_{i,t-1}\, e^{\eta y_t x_{i,t}} \\ w_{i,t} &= \frac{w'_{i,t}}{\sum_{j=1}^d w'_{j,t}} \qquad \text{for } i = 1, \ldots, d. \end{aligned}$$

The theory behind the analysis of general additive family of algorithms shows that, notwithstanding their apparent diversity, Winnow and Perceptron are actually instances of the same additive algorithm.

To obtain a selective sampling version of Winnow we proceed exactly as we did in the previous cases: we query the label y_t with probability $b/(b+|\widehat{p}_t|)$, where $|\widehat{p}_t|$ is the margin computed by the algorithm. The complete pseudo-code is described in Figure 4.

The mistake bound we prove for selective sampling Winnow is somewhat atypical since, unlike the Perceptron-like algorithms analyzed so far, the choice of the learning rate η given in this theorem is the same as the one suggested by the original Winnow analysis (see, e.g., Littlestone, 1989; Grove et al., 2001). Furthermore, since a meaningful bound in Winnow requires η be chosen in terms of γ, it turns out that in the selective sampling version there is no additional tuning to perform, and we are able to obtain the same mistake bound as the original version. Thus, unlike the other cases, the selective sampling mechanism does not weaken in any respect the original mistake bound, apart from turning a deterministic bound into an expected one.

Theorem 4 *If the algorithm of Figure 4 is run with parameters*

$$\eta = \frac{2(1-\alpha)\gamma}{X_\infty^2} \qquad \text{and} \qquad b = \alpha\gamma \qquad \text{for some } \alpha \in (0,1)$$

on a sequence $(x_1,y_1),\ldots,(x_n,y_n) \in \mathbb{R}^d \times \{-1,+1\}$ *of examples such that* $\|x_t\|_\infty \le X_\infty$ *for all* $t = 1,\ldots,n$*, then for all* $u \in \mathbb{R}^d$ *in the probability simplex,*

$$\mathbb{E}\left[\sum_{t=1}^n M_t\right] \le \frac{1}{\alpha}\,\frac{\overline{L}_{\gamma,n}(u)}{\gamma} + \frac{1}{2\alpha(1-\alpha)}\,\frac{X_\infty^2 \ln d}{\gamma^2} \ .$$

As before, the expected number of labels queried by the algorithm equals $\sum_{t=1}^n \mathbb{E}\left[\frac{b}{b+|\widehat{p}_t|}\right]$.

Proof Similarly to the proof of Theorem 1, we estimate the influence of an update on the distance between the current weight w_{t-1} and an arbitrary "target" hyperplane u, where in this case both vectors live in the probability simplex. Unlike the Perceptron analysis, based on the squared Euclidean distance, the analysis of Winnow uses the Kullback-Leibler divergence, or relative entropy, $\mathrm{KL}(\cdot,\cdot)$ to measure the progress of w_{t-1} towards u. The relative entropy of any two vectors u,v belonging to the probability simplex on $\mathbb{R}^d$ is defined by

$$\mathrm{KL}(u,v) = \sum_{i=1}^d u_i \ln \frac{u_i}{v_i} \ .$$

Fix an arbitrary sequence $(x_1,y_1),\ldots,(x_n,y_n) \in \mathbb{R}^d \times \{-1,+1\}$ of examples. As in the proof of Theorem 1, we have that $M_t Z_t = 1$ implies

$$\begin{aligned}
\eta\big(\gamma - \ell_{\gamma,t}(u)\big) &= \eta\big(\gamma - (\gamma - y_t, u^\top x_t)_+\big) \\
&\le \eta\, y_t\, u^\top x_t \\
&= \eta\, y_t (u - w_{t-1} + w_{t-1})^\top x_t \\
&= \eta\, y_t (u - w_{t-1})^\top x_t + \eta\, y_t\, w_{t-1}^\top x_t \ .
\end{aligned}$$

Besides, exploiting a simple identity (as in the proof of Theorem 11.3 in Cesa-Bianchi and Lugosi, 2006, Chap. 5), we can rewrite the term $\eta\, y_t (u - w_{t-1})^\top x_t$ as

$$\eta\, y_t (u - w_{t-1})^\top x_t = \mathrm{KL}(u, w_{t-1}) - \mathrm{KL}(u, w_t) + \ln\left(\sum_{j=1}^d w_{j,t-1} e^{\eta y_t v_j}\right)$$

where $v_j = x_j - w_{t-1}^\top x_t$. This equation is similar to the one obtained in the analysis of the selective sampling Perceptron algorithm, but for the relative entropy replacing the squared Euclidean distance. Note, however, that the last term in the right-hand side of the above equation is not a relative entropy. To bound this last term, we consider the random variable X taking value $x_{i,t} \in [-X_\infty, X_\infty]$ with probability $w_{i,t-1}$. Then, from the Hoeffding inequality (Hoeffding, 1963) applied to X,

$$\ln\left(\sum_{j=1}^d w_{j,t-1} e^{\eta y_t v_j}\right) = \ln \mathbb{E}\left[e^{\eta y_t (X - \mathbb{E}X)}\right] \le \frac{\eta^2}{2} X_\infty^2 \ .$$

We plug back, rearrange and note that $w_t = w_{t-1}$ whenever $M_t Z_t = 0$. This gets

$$M_t Z_t \eta \left(\gamma + |\widehat{p}_t| - \frac{\eta}{2} X_\infty^2\right) \leq M_t Z_t \eta \, \ell_{\gamma,t}(u) + \mathrm{KL}(u, w_{t-1}) - \mathrm{KL}(u, w_t) \ ,$$

holding for any t. Summing over $t = 1, \ldots, n$ and dividing by η yields

$$\sum_{t=1}^{n} M_t Z_t \left(\gamma + |\widehat{p}_t| - \frac{\eta}{2} X_\infty^2\right) \leq \sum_{t=1}^{n} M_t Z_t \, \ell_{\gamma,t}(u) + \frac{\mathrm{KL}(u, w_0)}{\eta} - \frac{\mathrm{KL}(u, w_n)}{\eta} \ .$$

We drop the last term (which is nonpositive), and use $\mathrm{KL}(u, w_0) \leq \ln d$ holding for any u in the probability simplex whenever $w_0 = (1/d, \ldots, 1/d)$. Then the above reduces to

$$\sum_{t=1}^{n} M_t Z_t \left(\gamma + |\widehat{p}_t| - \frac{\eta}{2} X_\infty^2\right) \leq \sum_{t=1}^{n} M_t Z_t \, \ell_{\gamma,t}(u) + \frac{\ln d}{\eta} \ .$$

Substituting our choice for η and b yields

$$\sum_{t=1}^{n} M_t Z_t \left(b + |\widehat{p}_t|\right) \leq \sum_{t=1}^{n} M_t Z_t \, \ell_{\gamma,t}(u) + \frac{X_\infty^2 \ln d}{2(1-\alpha)\gamma} \ .$$

To conclude, it suffices to exploit $\mathbb{E}_{t-1} Z_t = b/(b + |\widehat{p}_t|)$ and proceed as in the proof of the previous theorems. ■

4. Experiments

To investigate the empirical behavior of our algorithms we carried out a series of experiments on the first (in chronological order) 40,000 newswire stories from the Reuters Corpus Volume 1 (Reuters, 2000). Each story in this dataset is labelled with one or more elements from a set of 101 categories. In our experiments, we associated a binary classification task with each one of the 50 most frequent categories in the dataset, ignoring the remaining 51 (this was done mainly to reduce the effect of unbalanced datasets). All results presented in this section refer to the average performance over these 50 binary classification tasks. Though all of our algorithms are randomized, we did not compute averages over multiple runs of the same experiment, since we empirically observed that the variances of our statistics are quite small for the sample size taken into consideration.

To evaluate the algorithms we used the F-measure (harmonic average between precision and recall) since this is the most widespread performance index in text categorization experiments. Replacing F-measure with classification accuracy yields results that are qualitatively similar to the ones shown here.

We focused on the following three algorithms: the selective sampling Perceptron algorithm of Figure 1 (here abbreviated as SEL-P), its adaptive version of Figure 2 (abbreviated as SEL-ADA), and the selective sampling second-order Perceptron algorithm of Figure 3 (abbreviated as SEL-2ND).

In Figure 5 we check whether our margin-based sampling technique achieves a better performance than the baseline sampling strategy of querying each label with constant probability. In particular, we fixed 7 different sampling rates (from 29.2% to 71.8%) and run SEL-P each time with the parameter b chosen so as to obtain the desired sampling rate. Then we compared the achieved

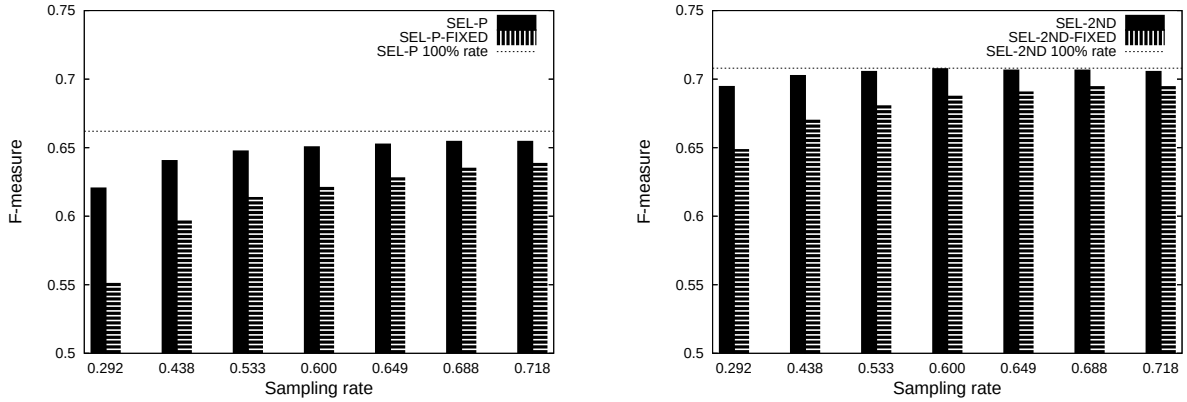


Figure 5: Comparison between margin-based sampling and random sampling with pre-specified sampling rate for the Perceptron algorithm (left) and the second-order Perceptron algorithm (right). The dotted lines show the performance obtained by querying all labels.

performance to the performance obtained by sampling each label with constant probability, i.e., the case when the Bernoulli random variables Z_t in step (3) of Figure 1 have constant parameter equal to the desired sampling rate. We call this variant SEL-P-FIXED. The same experiment was repeated using SEL-2ND and its fixed probability variant SEL-2ND-FIXED.

The following table shows the values of parameter b leading to the fixed sampling rates for both experiments.

SAMPLING RATE	b (SEL-P)	b (SEL-2ND)
0.292	0.250	0.040
0.438	0.500	0.085
0.533	0.750	0.125
0.600	1.000	0.168
0.649	1.250	0.210
0.688	1.500	0.236
0.718	1.750	0.240

Note that in both cases the margin-based sampling technique is clearly dominating. Also, as expected, the difference between the two techniques tends to shrink as the sampling rate gets larger. In Figure 6 we illustrate the sensitivity of performance and sampling rate to different choices of the input parameter b for the two algorithms SEL-P and SEL-2ND. This experiment supports Theorems 1 and 3 in two ways: First, it shows that the choice of b achieving a performance comparable to the one obtained by sampling all labels can save a significant fraction of labels; second, this choice is not unique. Indeed, in a sizeable interval of values for parameter b, the sampling rate decreases significantly with b while the performance level is essentially constant. In Figure 7 we directly compare the performance of SEL-P, SEL-2ND, and SEL-ADA for different values of their average sampling rate (obtained, as before, via suitable choices of their input parameters b and β). This experiment confirms that SEL-2ND is the algorithm offering the best trade-off between performance and sampling rate. On the other hand, the fact that SEL-ADA performs slightly worse than SEL-P, together with the results of Figure 6, appears to indicate that our adaptive choice of b can only be motivated on theoretical grounds.

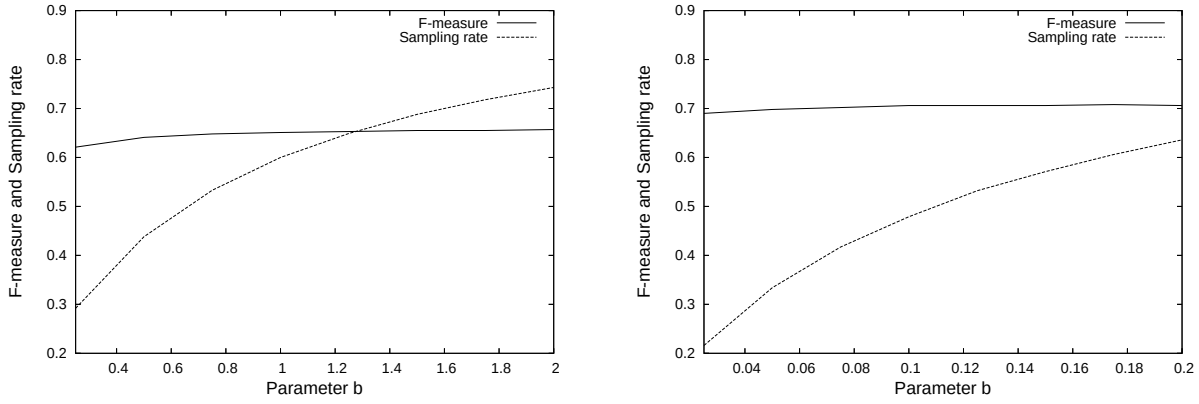


Figure 6: Dependence of performance and sampling rate on the b parameter for the Perceptron algorithm (left) and the second-order Perceptron algorithm (right).

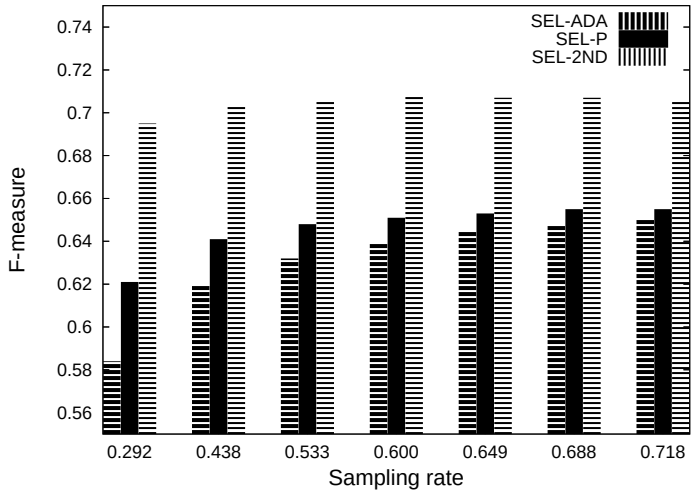


Figure 7: Performance level of SEL-P, SEL-2ND, and SEL-ADA at different sampling rates.

In the last experiment we fixed a target value (0.65) for the F-measure averaged over all 50 categories and we tuned all algorithms to achieve that performance after training on the entire sequence of 40,000 examples. Then, we compared the sampling rates that each algorithm needed to attain the target performance. To get a more accurate picture of the behavior of each algorithm, each time a block of 4,000 training examples was completed, we plotted the average F-measure and sampling rate achieved over that block. The results are reported in Figure 8. Note that SEL-P uses an average sampling rate of about 60%, while SEL-ADA needs a larger (and growing with time) sampling rate of about 74%. On the other hand, SEL-2ND uses only about 9% of the labels. Note also that the sampling rate of SEL-P and SEL-2ND decreases with time, thus indicating that in both cases the margin tends to grow in magnitude. The small sampling rate exhibited by SEL-2ND compared to SEL-P

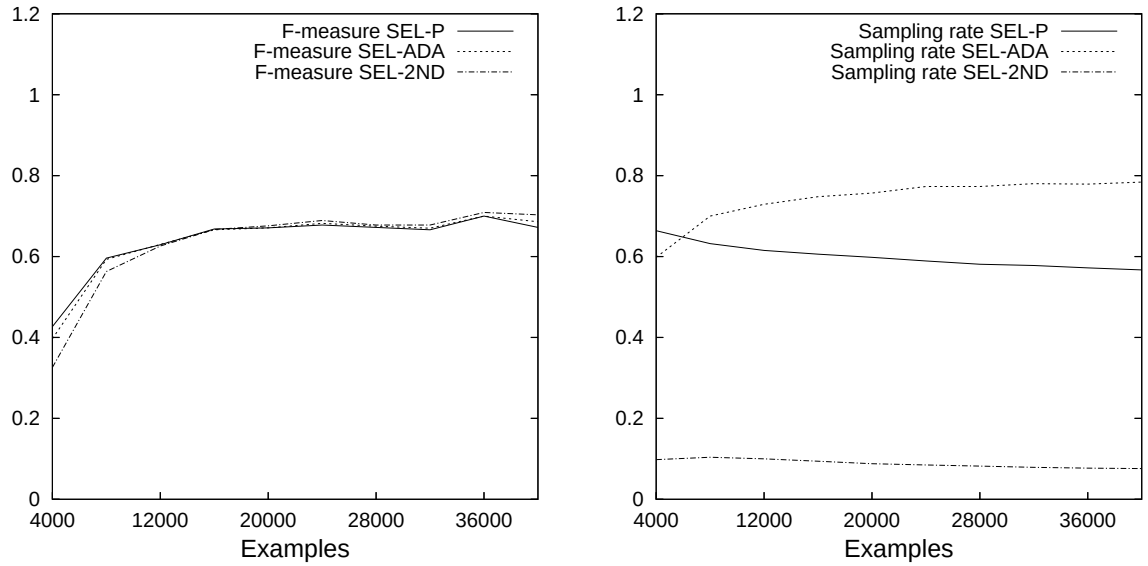


Figure 8: The right plot shows the sampling rates required by different algorithms to achieve a given target performance value (shown in the left plot).

(and SEL-ADA) might be an indication that the second-order Perceptron tends to achieve a larger margin than the standard Perceptron, but we do not have a clear explanation for this phenomenon.

5. Conclusions and Open Problems

We have introduced a general technique for turning linear-threshold algorithms from the general additive family into selective sampling algorithms. We have analyzed these algorithms in a worst-case on-line learning setting, providing bounds on the expected number of mistakes. Our theoretical investigation naturally arises from the traditional way margin-based algorithms are analyzed in the mistake bound model of on-line learning (Littlestone, 1988; Grove et al., 2001; Gentile and Warmuth, 1999; Freund and Schapire, 1999; Gentile, 2003; Cesa-Bianchi et al., 2005). This investigation suggests that our semi-supervised algorithms can achieve, on average, the same accuracy as that of their fully supervised counterparts, but allowing a substantial saving of labels. When applied to (kernel-based) Perceptron-like algorithms, label saving directly implies higher sparsity for the computed classifier which, in turn, yields a running time saving in both training and test phases.

Our theoretical results are corroborated by an empirical comparison on textual data. In these experiments we have shown that proper choices of the scaling parameter b yield a significant reduction in the rate of queried labels without causing an excessive degradation of the classification performance. In addition, we have also shown that by fixing ahead of time the total number of label observations, the margin-driven way of distributing these observations over the training set is largely more effective than a random one.

The choice of the scaling parameter b might affect performance in a significant way. Thus we have also provided a theoretical analysis for an adaptive parameter version of the (first-order) selective sampling Perceptron algorithm. This analysis shows that it is still possible to obtain, with

no prior information, a bound on the expected number of mistakes having the same form as the one achieved by choosing the "best" b in hindsight. Now, it is intuitively clear that the number of prediction mistakes and the number of queried labels can be somehow traded-off against each other. Within this trade-off, the above "best" choice is only aimed at minimizing mistakes, rather than queried labels. In fact, the practical utility of this adaptive algorithm seems, at present, fairly limited.

There are many ways this work could be extended. Perhaps the most important is being able to quantify the expected number of requested labels as a function of the problem parameters (margin of the data and so on). It is worth observing that for the adaptive version of the selective sampling Perceptron (Figure 2) we can easily derive a *lower* bound on the label sampling rate. Assume for simplicity that $\|x_t\| = 1$ for all t. Then we can write

$$\begin{aligned}
\frac{b_{t-1}}{b_{t-1}+|\widehat{p}_t|} &= \frac{\beta\sqrt{1+K_{t-1}}}{\beta\sqrt{1+K_{t-1}}+|w_{t-1}^{\top}x_t|} \\
&\geq \frac{\beta\sqrt{1+K_{t-1}}}{\beta\sqrt{1+K_{t-1}}+\|w_{t-1}\|} \\
&\geq \frac{\beta\sqrt{1+K_{t-1}}}{\beta\sqrt{1+K_{t-1}}+\sqrt{K_{t-1}}} \qquad \text{(using Inequality (7))} \\
&\geq \frac{\beta}{\beta+1}
\end{aligned}$$

holding for any trial t. Is it possible to obtain a meaningful *upper* bound? At first glance, this requires a lower bound on the margin $|\widehat{p}_t|$. But since there are no guarantees on the margin the algorithm achieves (even in the separable case), this route does not look profitable. Would such an argument work for on-line large margin algorithms, such as those by Li and Long (2002) and Gentile (2001)?

As a related issue, our theorems do not make any explicit statement about the number of weight updates (i.e., support vectors) computed by our selective sampling algorithms. We would like to see a theoretical argument that enables us to combine the bound on the number of mistakes with a bound on the number of labels, resulting in an informative upper bound on the number of updates.

Finally, the adaptive parameter version of Figure 2 centers on inequalities such as (7) to determine the current label request rate. It seems these inequalities are too coarse to make the algorithm effective in practice. Our experiments basically show that this algorithm tends to query more labels than needed. It turns out there are many ways one can modify this algorithm to make it less "cautious", though this gives rise to algorithms which seem to escape a crisp mathematical analysis. We would like to devise an adaptive parameter version of the selective sampling Perceptron algorithm that both lends itself to formal analysis and is competitive in practice.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments that greatly helped to improve the presentation of this paper. In particular, we thank one of them for finding a mistake in our initial version of the proof of Theorem 4. We would also like to thank the action editor for his timely work.

The authors gratefully acknowledge partial support by the PASCAL Network of Excellence under EC grant no. 506778. This publication only reflects the authors' views.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Journal of Computer and System Sciences*, 64:48–75, 2002.

K. S. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential familiy of distributions. *Machine Learning*, 43(3):211–246, 2001.

H. D. Block. The Perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34: 123–135, 1962.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning reserarch*, 6:1579–1619, 2005.

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 111–11. Morgan Kaufman, 2000.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. Learning probabilistic linear-threshold classifiers via selective sampling. In *Proceedings of the 16th Annual Conference on Learning Theory, LNAI 2777*, pages 373–386. Springer, 2003.

N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order Perceptron algorithm. *SIAM Journal on Computing*, 43(3):640–668, 2005.

N. Cesa-Bianchi and G. Lugosi. Potential-based algorithms in on-line prediction and game theory. *Machine Learning*, 51(3):239–261, 2003.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

R. Cohn, L. Atlas, and R. Ladner. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems 2*. MIT Press, 1990.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2001.

S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of Perceptron-based active learning. In *Proceedings of the 18th Annual Conference on Learning Theory, LNAI 2777*, pages 249–263. Springer, 2005.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: a kernel-based Perceptron on a fixed budget. In *Advances in Neural Information Processing Systems 18*, pages 259–266. MIT Press, 2006.

R. Duda and P. Hart, and D. Stork. *Pattern classification, second edition*. Wiley Interscience, 2000.

J. Forster. On relative loss bounds in generalized linear regression. In *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory, LNCS 1684*, pages 269–280. Springer, 1999.

Y. Freund and R. Schapire. Large margin classification using the Perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

Y. Freund, S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2/3):133–168, 1997.

C. Gentile. A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research*, 2:213–242, 2001.

C. Gentile. The robustness of the *p*-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.

C. Gentile and M. Warmuth. Linear hinge loss and average margin. In *Advances in Neural Information Processing Systems 10*, pages 225–231. MIT Press, 1999.

A. J. Grove, N. Littlestone, and D. Schuurmans. General convergence results for linear discriminant updates. *Machine Learning*, 43(3):173–210, 2001.

D. P. Helmbold, N. Littlestone, and P. M. Long. Apple tasting. *Information and Computation*, 161 (2):85–139, 2000.

D. P. Helmbold and S. Panizza. Some label efficient learning results. In *Proceedings of the 10th Annual Conference on Computational Learning Theory*, pages 218–230. ACM Press, 1997.

M. Herbster and M. K. Warmuth. Tracking the Best Linear Predictor. *Journal of Machine Learning Research*, 1: 281–309, 2001.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

T. L. Lai and C. Z. Wei. Least squares estimates in stochastic regression models with applications to identification and control of dynamic systems. *The Annals of Statistics*, 10(1):154–166, 1982.

Y. Li and P. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46:361–387, 2002.

N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, University of California Santa Cruz, 1989.

A. B. J. Novikov. On convergence proofs on Perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata, vol. XII*, pages 615–622, 1962.

Reuters. Reuters corpus vol. 1, 2000.
URL about.reuters.com/researchandstandards/corpus/.

F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the 17th International Conference on Machine Learning*, pages 999–1006. Morgan Kaufmann, 2000.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

M. K. Warmuth and A. K. Jagota. Continuous and discrete-time nonlinear gradient descent: Relative loss bounds and convergence. In *Electronic proceedings of the 5th International Symposium on Artificial Intelligence and Mathematics*, 1997.

Journal of Machine Learning Research 7 (2006) 1231–1264 Submitted 10/05; Revised 5/06; Published 7/06

Nonparametric Quantile Estimation

Ichiro Takeuchi TAKEUCHI@PA.INFO.MIE-U.AC.JP
Division of Computer Science
Graduate School of Engineering, Mie University
1577, Kurimamachiya-cho, Tsu 514-8507, Japan

Quoc V. Le QUOC.LE@ANU.EDU.AU
Timothy D. Sears TIM.SEARS@ANU.EDU.AU
Alexander J. Smola ALEX.SMOLA@NICTA.COM.AU
RSISE, Australian National University and
Statistical Machine Learning Program, National ICT Australia
0200, ACT, Australia

Editor: Chris Williams

Abstract

In regression, the desired estimate of $y|x$ is not always given by a conditional mean, although this is most common. Sometimes one wants to obtain a good estimate that satisfies the property that a proportion, τ, of $y|x$, will be below the estimate. For $\tau = 0.5$ this is an estimate of the *median*. What might be called median regression, is subsumed under the term *quantile regression*. We present a nonparametric version of a quantile estimator, which can be obtained by solving a simple quadratic programming problem and provide uniform convergence statements and bounds on the quantile property of our estimator. Experimental results show the feasibility of the approach and competitiveness of our method with existing ones. We discuss several types of extensions including an approach to solve the *quantile crossing* problems, as well as a method to incorporate prior qualitative knowledge such as monotonicity constraints.

Keywords: support vector machines, kernel methods, quantile estimation, nonparametric techniques, estimation with constraints

1. Introduction

Regression estimation is typically concerned with finding a real-valued function f such that its values $f(x)$ correspond to the conditional mean of y, or closely related quantities. Many methods have been developed for this purpose, e.g. least mean square (LMS) regression, robust regression (Huber, 1981), or ε-insensitive regression (Vapnik, 1995; Vapnik et al., 1997). Regularized variants include Wahba (1990), penalized by a Reproducing Kernel Hilbert Space (RKHS) norm, and Hoerl and Kennard (1970), regularized via ridge regression.

1.1 Motivation

While these estimates of the mean serve their purpose, there exists a large area of problems where we are more interested in estimating a quantile. That is, we might wish to know other features of the the distribution of the random variable $y|x$:

- A device manufacturer may wish to know what are the 10% and 90% quantiles for some feature of the production process, so as to tailor the process to cover 80% of the devices produced.

- For risk management and regulatory reporting purposes, a bank may need to estimate a lower bound on the changes in the value of its portfolio which will hold with high probability.

- A pediatrician requires a growth chart for children given their age and perhaps even medical background, to help determine whether medical interventions are required, e.g. while monitoring the progress of a premature infant.

These problems are addressed by a technique called Quantile Regression (QR) or Quantile Estimation championed by Koenker (see Koenker, 2005, for a description, practical guide, and extensive list of references). These methods have been deployed in econometrics, social sciences, ecology, etc. The purpose of our paper is:

- To bring the technique of quantile regression to the attention of the machine learning community and show its relation to ν-Support Vector Regression (Schölkopf et al., 2000).

- To demonstrate a nonparametric version of QR which outperforms the currently available nonlinear QR regression formations (Koenker, 2005). See Section 5 for details.

- To derive small sample size results for the algorithms. Most statements in the statistical literature for QR methods are of asymptotic nature (Koenker, 2005). Empirical process results permit us to define two quality criteria and show tail bounds for both of them in the finite-sample-size case.

- To extend the technique to permit commonly desired constraints to be incorporated. As examples we show how to enforce non-crossing constraints and a monotonicity constraint. These constraints allow us to incorporate prior knowlege on the data.

1.2 Notation and Basic Definitions

In the following we denote by $\mathcal{X}, \mathcal{Y}$ the domains of x and y respectively. $X = \{x_1, \ldots, x_m\}$ denotes the training set with corresponding targets $Y = \{y_1, \ldots, y_m\}$, both drawn independently and identically distributed (iid) from some distribution $p(x,y)$. With some abuse of notation y also denotes the vector of all y_i in matrix and vector expressions, whenever the distinction is obvious.

Unless specified otherwise $\mathcal{H}$ denotes a Reproducing Kernel Hilbert Space (RKHS) on $\mathcal{X}$, k is the corresponding kernel function, and $K \in \mathbb{R}^{m \times m}$ is the kernel matrix obtained via $K_{ij} = k(x_i, x_j)$. θ denotes a vector in *feature space* and $\phi(x)$ is the corresponding feature map of x. That is, $k(x,x') = \langle \phi(x), \phi(x') \rangle$. Finally, $\alpha \in \mathbb{R}^m$ is the vector of Lagrange multipliers.

Definition 1 (Quantile) *Denote by $y \in \mathbb{R}$ a random variable and let $\tau \in (0,1)$. Then the τ-quantile of y, denoted by μ_τ is given by the infimum over μ for which $\Pr\{y \leq \mu\} = \tau$. Likewise, the conditional quantile $\mu_\tau(x)$ for a pair of random variables $(x,y) \in \mathcal{X} \times \mathbb{R}$ is defined as the function $\mu_\tau : \mathcal{X} \to \mathbb{R}$ for which pointwise μ_τ is the infimum over μ for which $\Pr\{y \leq \mu | x\} = \tau$.*

1.3 Examples

To illustrate regression analyses with conditional quantile functions, we provide two simple examples here.

1.3.1 ARTIFICIAL DATA

The above definition of conditional quantiles may be best illustrated by a simple example. Consider a situation where the relationship between x and y is represented as

$$y(x) = f(x) + \xi, \text{ where } \xi \sim \mathcal{N}\left(0, \sigma(x)^2\right). \tag{1}$$

Here, note that, the amount of noise ξ is a function of x. Since ξ is symmetric with mean and median 0 we have $\mu_{0.5}(x) = f(x)$. Moreover, we can compute the τ-th quantiles by solving $\Pr\{y \leq \mu | x\} = \tau$ explicitly. Since ξ is normally distributed, we know that the τ-th quantile of ξ is given by $\sigma(x)\Phi^{-1}(\tau)$, where Φ is the cumulative distribution function of the normal distribution with unit variance. This means that

$$\mu_\tau(x) = f(x) + \sigma(x)\Phi^{-1}(\tau).$$

Figure 1 shows the case where x is uniformly drawn from $[-1,1]$ and y is obtained based on (1) with $f(x) = \text{sinc}(x)$ and $\sigma(x) = 0.1\exp(1-x)$. The black circles are 500 data examples and the five curves are $\tau = 0.10, 0.25, 0.50, 0.75$ and 0.90 conditional quantile functions. The probability densities $p(y|x=-0.5)$ and $p(y|x=+0.5)$ are superimposed. The τ-th conditional quantile function is obtained by connecting the τ-th quantile of the conditional distribution $p(y|x)$ for all $x \in \mathcal{X}$. We see that $\tau = 0.5$ case provides the central tendency of the data distribution and $\tau = 0.1$ and 0.9 cases track the lower and upper envelope of the data points, respectively. The error bars of many regression estimates can be viewed as crude quantile regressions. Quantile regression on the other hand tries to estimate such quantities directly.

1.3.2 REAL DATA

The next example is based on actual measurements of bone density (BMD) in adolescents. The data was originally reported in Bachrach et al. (1999) and is also analyzed in Hastie et al. (2001).[1] Figure 2 (a) shows a regression analysis with conditional mean and figure 2 (b) shows that with a set of conditional quantiles for the variable BMD. The response in the vertical axis is relative change in spinal BMD and the covariate in the horizontal axis is the age of the adolescents. The conditional mean analysis (a) provides only the central tendency of the conditional distribution, while apparently the entire distribution of BMD changes according to age. The conditional quantile analysis (b) gives us more detailed description of these changes. For example, we can see that the variance of the BMD changes with the age (heteroscedastic) and that the conditional distribution is slightly positively skewed.

2. Quantile Estimation

Given the definition of $\mu_\tau(x)$ and knowledge of support vector machines we might be tempted to use version of the ε-insensitive tube regression to estimate $\mu_\tau(x)$. More specifically one might try to

1. The data is also available from the website http://www-stat.stanford.edu/ElemStatlearn.

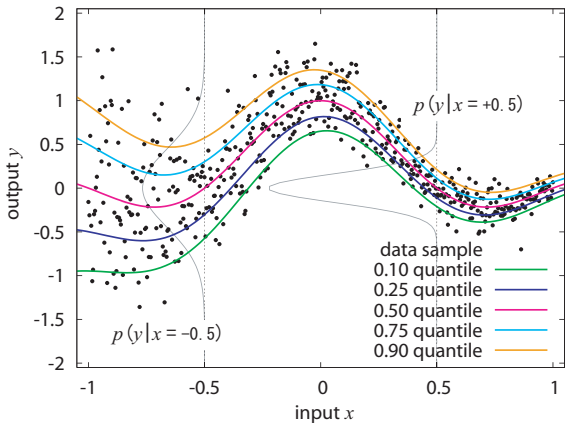


Figure 1: Illustration of conditional quantile functions of a simple artificial system in (1) with $f(x) = \mathrm{sinc}(x)$ and $\sigma(x) = 0.1\exp(1-x)$. The black circles are 500 data examples and the five curves are $\tau = 0.10, 0.25, 0.50, 0.75$ and 0.90 conditional quantile functions. The probability densities $p(y|x=-0.5)$ and $p(y|x=+0.5)$ are superimposed. In this paper, we are concerned with the problem of estimating these conditional quantile functions from training data.

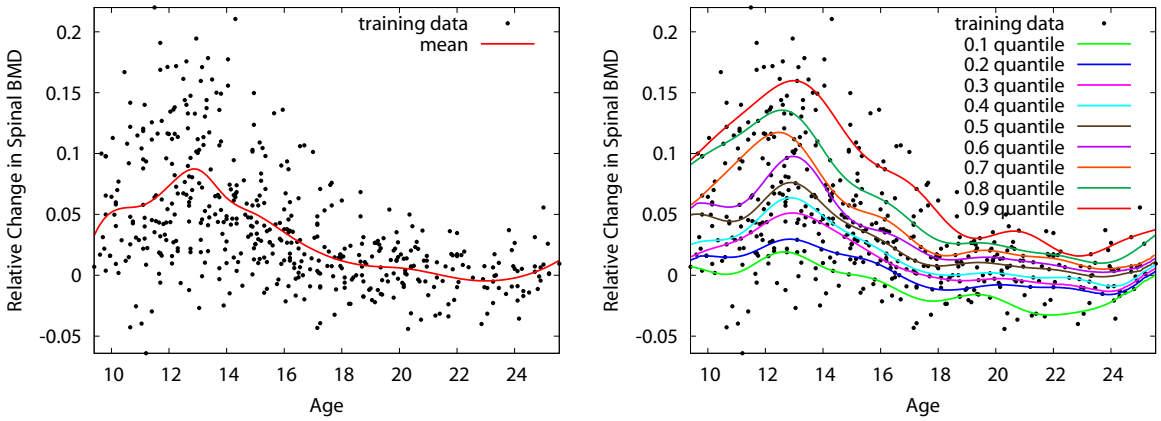
estimate quantiles nonparametrically using an extension of the ν-trick, as outlined in Schölkopf et al. (2000). However this approach carries the disadvantage of requiring us to estimate both an upper and lower quantile *simultaneously*.[2] While this can be achieved by quadratic programming, in doing so we estimate "too many" parameters simultaneously. More to the point, if we are interested in finding an upper bound on y which holds with 0.95 probability we may not want to use information about the 0.05 probability bound in the estimation. Following Vapnik's paradigm of estimating only the relevant parameters directly (Vapnik, 1982) we attack the problem by estimating each quantile separately. For completeness and comparison, we provide a detailed description of a symmetric quantile regression in Appendix A.

2.1 Loss Function

The basic strategy behind quantile estimation arises from the observation that minimizing the ℓ_1-loss function for a location estimator yields the median. Observe that to minimize $\sum_{i=1}^m |y_i - \mu|$ by choice of μ, an equal number of terms $y_i - \mu$ have to lie on either side of zero in order for the derivative wrt. μ to vanish. Koenker and Bassett (1978) generalizes this idea to obtain a regression estimate for any quantile by tilting the loss function in a suitable fashion. More specifically one may show that the following "pinball" loss leads to estimates of the τ-quantile:

Lemma 2 (Quantile Estimator) *Let $Y = \{y_1, \ldots, y_m\} \subset \mathbb{R}$ and let $\tau \in (0,1)$ then the minimizer μ_τ of $\sum_{i=1}^m l_\tau(y_i - \mu)$ with respect to μ satisfies:*

2. Schölkopf et al. (2000) does, in fact, suggests that a choice of different upper bounds on the dual problem would lead to estimators which weigh errors for positive and negative excess differently, that is, which would lead to quantile regression estimators.



(a) Conditional mean analysis (b) Conditional quantile analysis

Figure 2: An illustration of (a) conditional mean analysis and (b) conditional quantile analysis for a data set on bone mineral density (BMD) in adolescents. In (a) the conditional mean curve is estimated by regression spline with least square criterion. In (b) the nine curves are the estimated conditional quantile curves at orders $0.1, 0.2, \ldots, 0.9$. The set of conditional quantile curves provides more informative description of the relationship among variables such as non-constant variance or non-normality of the noise (error) distribution. In this paper, we are concerned with the problem of estimating these conditional quantiles.

$$l_\tau(\xi) = \begin{cases} \tau\xi & \text{if } \xi \geq 0 \\ (\tau-1)\xi & \text{if } \xi < 0 \end{cases} \qquad (2)$$

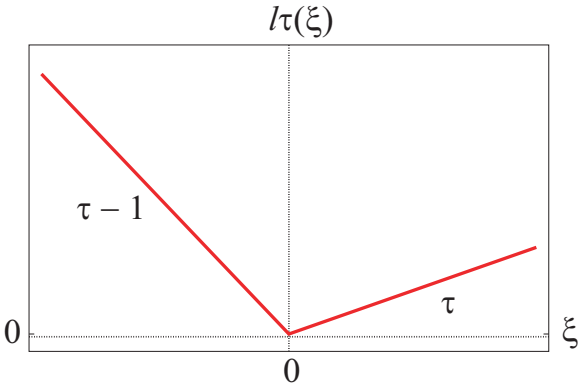


Figure 3: Pinball loss function for quantile estimation.

1. *The number of terms, m_-, with $y_i < \mu_\tau$ is bounded from above by τm.*
2. *The number of terms, m_+, with $y_i > \mu_\tau$ is bounded from above by $(1-\tau)m$.*
3. *For $m \to \infty$, the fraction $\frac{m_-}{m}$, converges to τ if $\Pr(y)$ does not contain discrete components.*

Proof Assume that we are at an optimal solution. Then, increasing the minimizer μ by $\delta\mu$ changes the objective by $[(1-m_+)(1-\tau) - m_+\tau]\,\delta\mu$. Likewise, decreasing the minimizer μ by $\delta\mu$ changes the objective by $[-m_-(1-\tau) + (1-m_-)\tau]\,\delta\mu$. Requiring that both terms are nonnegative at opti-

mality in conjunction with the fact that $m_- + m_+ \leq m$ proves the first two claims. To see the last claim, simply note that the event $y_i = y_j$ for $i \neq j$ has probability measure zero for distributions not containing discrete components. Taking the limit $m \to \infty$ shows the claim. ■

The idea is to use the same loss function for functions, $f(x)$, rather than just constants in order to obtain quantile estimates conditional on x. Koenker (2005) uses this approach to obtain linear estimates and certain nonlinear spline models. In the following we will use kernels for the same purpose.

2.2 Optimization Problem

Based on $l_\tau(\xi)$ we define the expected quantile risk as

$$R[f] := \mathbf{E}_{p(x,y)}\left[l_\tau(y - f(x))\right]. \tag{3}$$

By the same reasoning as in Lemma 2 it follows that for $f : \mathcal{X} \to \mathbb{R}$ the minimizer of $R[f]$ is the quantile $\mu_\tau(x)$. Since $p(x,y)$ is unknown and we only have X, Y at our disposal we resort to minimizing the empirical risk plus a regularizer:

$$R_{\mathrm{reg}}[f] := \frac{1}{m}\sum_{i=1}^{m} l_\tau(y_i - f(x_i)) + \frac{\lambda}{2}\|g\|^2_{\mathcal{H}} \text{ where } f = g + b \text{ and } b \in \mathbb{R}. \tag{4}$$

Here $\|\cdot\|_{\mathcal{H}}$ is RKHS norm and we require $g \in \mathcal{H}$. Notice that we do not regularize the constant offset, b, in the optimization problem. This ensures that the minimizer of (4) will satisfy the quantile property:

Lemma 3 (Empirical Conditional Quantile Estimator) *Assuming that f contains a scalar unregularized term, the minimizer of (4) satisfies:*

1. *The number of terms m_- with $y_i < f(x_i)$ is bounded from above by τm.*
2. *The number of terms m_+ with $y_i > f(x_i)$ is bounded from above by $(1-\tau)m$.*
3. *If (x,y) is drawn iid from a distribution $\Pr(x,y)$, with $\Pr(y|x)$ continuous and the expectation of the modulus of absolute continuity of its density satisfying $\lim_{\delta\to 0}\mathbf{E}\left[\varepsilon(\delta)\right] = 0$. With probability 1, asymptotically, $\frac{m_-}{m}$ equals τ.*

Proof For the two claims, denote by f^* the minimum of $R_{\mathrm{reg}}[f]$ with $f^* = g^* + b^*$. Then $R_{\mathrm{reg}}[g^* + b]$ has to be minimal for $b = b^*$. With respect to b, however, minimizing R_{reg} amounts to finding the τ quantile in terms of $y_i - g(x_i)$. Application of Lemma 2 proves the first two parts of the claim.

For the second part, an analogous reasoning to Schölkopf et al. (2000, Proposition 1) applies. In a nutshell, one uses the fact that the measure of the δ-neighborhood of $f(x)$ converges to 0 for $\delta \to 0$. Moreover, for kernel functions the entropy numbers are well behaved (Williamson et al., 2001). The application of the union bound over a cover of such function classes completes the proof. Details are omitted, as the proof is identical to that of Schölkopf et al. (2000). ■

Later, in Section 4 we discuss finite sample size results regarding the convergence of $\frac{m_-}{m} \to \tau$ and related quantities. These statements will make use of scale sensitive loss functions. Before we do that, let us consider the practical problem of minimizing the regularized risk functional.

2.3 Dual Optimization Problem

Here we compute the dual optimization problem to (4) for efficient numerical implementation. Using the connection between RKHS and feature spaces we write $f(x) = \langle \phi(x), w \rangle + b$ and we obtain the following equivalent to minimizing $R_{\text{reg}}[f]$.

$$\underset{w,b,\xi_i^{(*)}}{\text{minimize}} \quad C\sum_{i=1}^{m} \tau\xi_i + (1-\tau)\xi_i^* + \frac{1}{2}\|w\|^2 \tag{5a}$$

$$\text{subject to} \quad y_i - \langle \phi(x_i), w \rangle - b \leq \xi_i \text{ and } \langle \phi(x_i), w \rangle + b - y_i \leq \xi_i^* \text{ where } \xi_i, \xi_i^* \geq 0 \tag{5b}$$

Here we used $C := 1/(\lambda m)$. The dual of this problem can be computed straightforwardly using Lagrange multipliers. The dual constraints for ξ and ξ^* can be combined into one variable. This yields the following dual optimization problem

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2}\alpha^\top K\alpha - \alpha^\top \vec{y} \text{ subject to } C(\tau-1) \leq \alpha_i \leq C\tau \text{ for all } 1 \leq i \leq m \text{ and } \vec{1}^\top\alpha = 0. \tag{6}$$

We recover f via the familiar kernel expansion

$$w = \sum_i \alpha_i \phi(x_i) \text{ or equivalently } f(x) = \sum_i \alpha_i k(x_i, x) + b. \tag{7}$$

Note that the constant b is the dual variable to the constraint $\mathbf{1}^\top\alpha = \mathbf{0}$. Alternatively, b can be obtained by using the fact that $f(x_i) = y_i$ for $\alpha_i \notin \{C(\tau-1), C\tau\}$. The latter holds as a consequence of the KKT-conditions on the primal optimization problem of minimizing $R_{\text{reg}}[f]$.

Note that the optimization problem is very similar to that of an ε-SV regression estimator (Vapnik et al., 1997). The key difference between the two estimation problems is that in ε-SVR we have an additional $\varepsilon\|\alpha\|_1$ penalty in the objective function. This ensures that observations with deviations from the estimate, i.e. with $|y_i - f(x_i)| < \varepsilon$ do not appear in the support vector expansion. Moreover the upper and lower constraints on the Lagrange multipliers α_i are matched. This means that we balance excess in both directions. The latter is useful for a regression estimator. In our case, however, we obtain an estimate which penalizes loss unevenly, depending on whether $f(x)$ exceeds y or vice versa. This is exactly what we want from a quantile estimator: by this procedure errors in one direction have a larger influence than those in the converse direction, which leads to the shifted estimate we expect from QR. A practical advantage of (6) is that it can be solved directly with standard quadratic programming code rather than using pivoting, as is needed in SVM regression (Vapnik et al., 1997).

A practical estimate does require a procedure for setting the regularization parameter. Figure 4 shows how QR responds to changing the regularization parameter. All three estimates in Figure 4 attempt to compute the median, subject to different smoothness constraints. While they all satisfy the quantile property having half the points on either side of the regression, some estimates appear track the observations better. This issue is addressed in Section 5 where we compute quantile regression estimates on a range of data sets.

3. Extensions and Modifications

Our optimization framework lends itself naturally to a series of extensions and modifications of the regularized risk minimization framework for quantile regression. In the following we discuss some extensions and modifications.

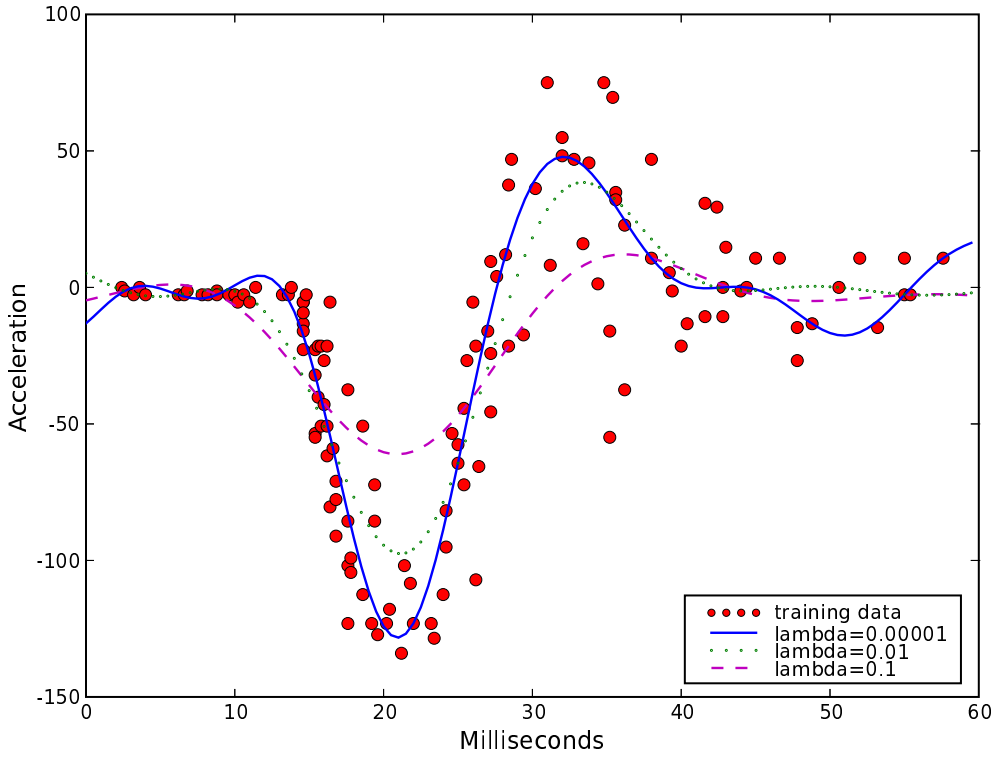


Figure 4: The data set measures acceleration in the head of a crash test dummy v. time in tests of motorcycle crashes. Three regularized versions of the median regression estimate ($\tau = 0.5$). While all three variants satisfy the quantile property, the degree of smoothness is controlled by the regularization constant λ. All three estimates compare favorably to a similar graph of nonlinear QR estimates reported by Koenker (2005).

3.1 Non-Crossing Constraints

When we want to estimate several conditional quantiles (e.g. $\tau = 0.1, 0.2, \ldots, 0.9$), two or more estimated conditional quantile functions can cross or overlap. This embarrassing phenomenon called *quantile crossings* occurs because each conditional quantile function is independently estimated (Koenker, 2005; He, 1997). Figure 5(a) shows BMD data presented in 1.3.2 and $\tau = 0.1, 0, 2, \ldots, 0.9$ conditional quantile functions estimated by the kernel-based estimator described in the previous section. Both of the input and the output variables are standardized in $[0,1]$. We note quantile crossings at several places, especially at the outside of the training data range ($x < 0$ and $1 < x$). In this subsection, we address this problem by introducing *non-crossing constraints*.[3] Figure 5(b) shows a family of conditional quantile functions estimated with the non-crossing constraints.

Suppose that we want to estimate n conditional quantiles at $0 < \tau_1 < \tau_2 < \ldots < \tau_n < 1$. We enforce *non-crossing* constraints at l points $\{x_j\}_{j=1}^{l}$ in the input domain $\mathcal{X}$. Let us write the model for the τ_h-th conditional quantile function as $f_h(x) = \langle \phi(x), w_h \rangle + b_h$ for $h = 1, 2, \ldots, n$. In $\mathcal{H}$ the non-crossing constraints are represented as linear constraints

$$\left\langle \phi(x_j), \omega_h \right\rangle + b_h \leq \left\langle \phi(x_j), \omega_{h+1} \right\rangle + b_{h+1}, \text{ for all } 1 \leq h \leq n-1,\ 1 \leq j \leq l. \tag{8}$$

Solving (5) or (6) for $1 \leq h \leq n$ with non-crossing constraints (8) allows us to estimate n conditional quantile functions not crossing at l points $x_1, \ldots, x_l \in \mathcal{X}$. The primal optimization problem is given by

$$\underset{w_h, b_h, \xi_{hi}^{(*)}}{\text{minimize}} \quad \sum_{h=1}^{n} \left[C \sum_{i=1}^{m} \tau_h \xi_{hi} + (1-\tau_h)\xi_{hi}^{*} + \frac{1}{2}\|w_h\|^2 \right] \tag{9a}$$

$$\text{subject to } \ y_i - \langle \phi(x_i), w_h \rangle - b_h = \xi_{hi} - \xi_{hi}^{*} \text{ where } \xi_{hi}, \xi_{hi}^{*} \geq 0, \quad \text{for all } 1 \leq h \leq n,\ 1 \leq i \leq m. \tag{9b}$$

$$\left\{ \left\langle \phi(x_j), \omega_{h+1} \right\rangle + b_{h+1} \right\} - \left\{ \left\langle \phi(x_j), \omega_h \right\rangle + b_h \right\} \geq 0, \quad \text{for all } 1 \leq h \leq n-1,\ 1 \leq j \leq l. \tag{9c}$$

Using Lagrange multipliers, we can obtain the dual optimization problem:

$$\underset{\alpha_h, \theta_h}{\text{minimize}} \quad \sum_{h=1}^{n} \left[\frac{1}{2}\alpha_h^{\top} K \alpha_h + \alpha_h^{\top} \tilde{K}(\theta_{h-1} - \theta_h) + \frac{1}{2}(\theta_{h-1} - \theta_h)^T \bar{K}(\theta_{h-1} - \theta_h) - \alpha_h^{\top} \vec{y} \right] \tag{10a}$$

$$\text{subject to } \ C(\tau_h - 1) \leq \alpha_{hi} \leq C\tau_h, \text{ for all } 1 \leq h \leq n, 1 \leq i \leq m, \tag{10b}$$

$$\theta_{hj} \geq 0, \text{ for all } 1 \leq h \leq n, 1 \leq j \leq l, \ \vec{1}^{\top} \alpha_h = 0, \text{ for all } 1 \leq h \leq n, \tag{10c}$$

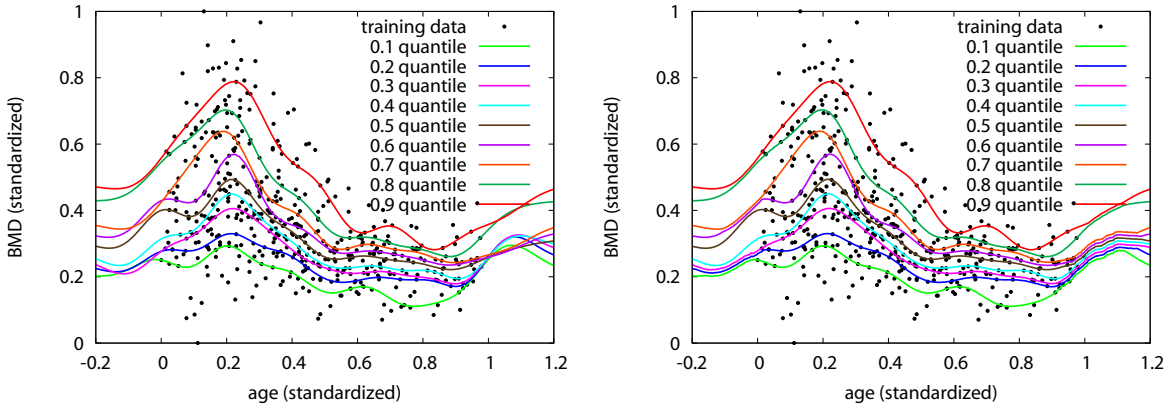
where θ_{hj} is the Lagrange multiplier of (9c) for all $1 \leq h \leq n$, $1 \leq j \leq l$, $\tilde{K}$ is $m \times l$ matrix with its (i, j)-th entry $k(x_i, x_j)$, $\bar{K}$ is $l \times l$ matrix with its (j_1, j_2)-th entry $k(x_{j1}, x_{j2})$ and θ_h is l-vector with its j-th entry θ_{hj} for all $1 \leq h \leq n$. For notational convenience we define $\theta_{0j} = \theta_{nj} = 0$ for all $1 \leq j \leq l$. The model for conditional quantile τ_h-th quantile function is now represented as

$$f_h(x) = \sum_{i=1}^{m} \alpha_{hi} k(x, x_i) + \sum_{j=1}^{l} (\theta_{h-1i} - \theta_{hi}) k(x, x_j) + b_h. \tag{11}$$

3. A part of the contents in this subsection was presented by one of the authors (Takeuchi and Furuhashi, 2004).

In section 5.2.1 we empirically investigate the effect of non-crossing constraints on the generalization performances.

It is worth noting that, after enforcing the non-crossing constraints, the quantile property as in Lemma 3 may not be guaranteed. This is because the method both tries to optimize for the quantile property and the non-crossing property (in relation to other quantiles). Hence, the final outcome may not empirically satisfy the quantile property. Yet, the non-crossing constraints are very nice because they ensure the semantics of the quantile definition: lower quantile level should not cross the higher quantile level.



(a) Without *non-crossing* constraints

(b) With *non-crossing* constraints

Figure 5: An example of *quantile crossing* problem in BMD data set presented in Section 1. Both of the input and the output variable are standardized in $[0,1]$. In (a) the set of conditional quantiles at $0.1, 0.2, \ldots, 0.9$ are estimated by the kernel-based estimator presented in the previous section. Quantile crossings are found at several points, especially at the outside of the training data range ($x < 0$ and $1 < x$). The plotted curves in (b) are the conditional quantile functions obtained with *non-crossing* constraints explained in Section 3.1. There are no *quantile crossing* even at the outside of the training data range.

3.2 Monotonicity and Growth Curves

Consider the situation of a health statistics office which wants to produce growth curves. That is, it wants to generate estimates of y being the height of a child given parameters x such as age, ethnic background, gender, parent's height, etc. Such curves can be used to assess whether a child's growth is abnormal.

A naive approach is to apply QR directly to the problem of estimating $y|x$. Note, however, that we have additional information about the biological process at hand: the height of every individual child is a *monotonically increasing* function of age. Without observing large amounts of data, there is no guarantee that the estimates $f(x)$, will also be monotonic functions of age. Figure 6 is an example of quantile regression with monotonicity constraints. The data set is taken from Mammen et al. (2001). Fuel efficiency (in miles per gallon) is studied as a function of engine output.

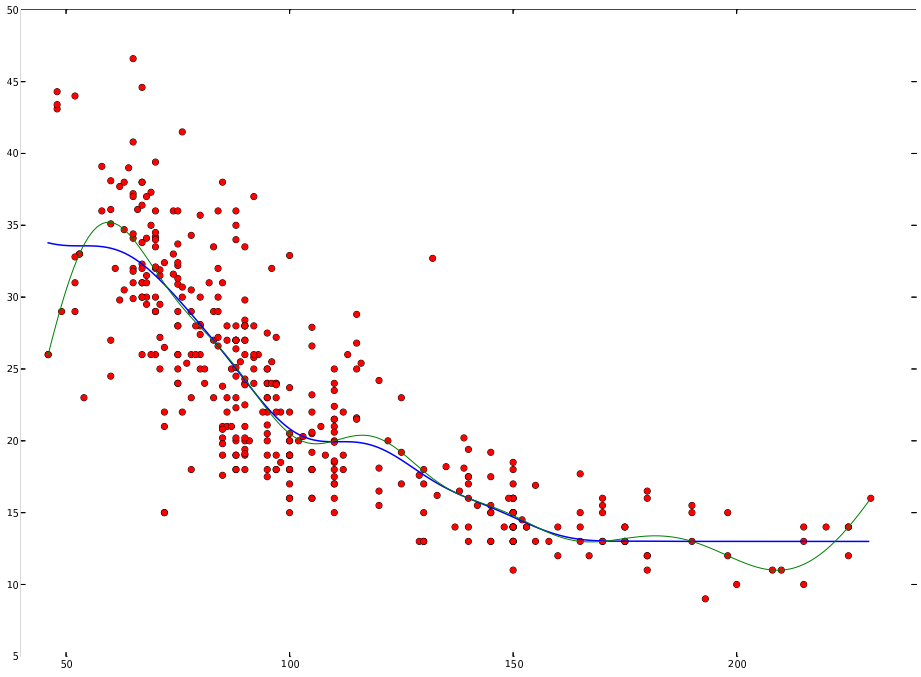


Figure 6: Example plots from quantile regression with and without monotonicity constraints. The thin line represents the nonparametric quantile regression without monotonicity constraints whereas the thick line represents the nonparamtric quantile regression with monotonicity constraints.

To address this problem we adopt an approach similar to (Vapnik et al., 1997; Smola and Schölkopf, 1998) and impose constraints on the derivatives of f directly. While this only ensures that f is monotonic on the observed data X, we could always add more locations x_i' for the express purpose of enforcing monotonicity.

Formally, we require that for a differential operator D, such as $D = \partial_{x_{\text{age}}}$ the estimate $Df(x) \geq 0$ for all $x \in X$. Using the linearity of inner products we have

$$Df(x) = D(\langle \phi(x), w \rangle + b) = \langle D\phi(x), w \rangle = \langle \psi(x), w \rangle \text{ where } \psi(x) := D\phi(x). \tag{12}$$

Note that accordingly inner products between ψ and ϕ can be obtained via $\langle \psi(x), \phi(x') \rangle = D_1 k(x,x')$ and $\langle \psi(x), \psi(x') \rangle = D_1 D_2 k(x,x')$, where D_1 and D_2 denote the action of D on the first and second argument of k respectively. Consequently the optimization problem (5) acquires an additional set of

constraints and we need to solve

$$\begin{aligned}\underset{w,b,\xi_i}{\text{minimize}} \;\; & C\sum_{i=1}^{m}\tau\xi_i + (1-\tau)\xi_i^* + \frac{1}{2}\|w\|^2 \\ \text{subject to} \;\; & y_i - \langle\phi(x_i), w\rangle - b \leq \xi_i, \;\; \langle\phi(x_i), w\rangle + b - y_i \leq \xi_i^*, \\ & \langle\psi(x_i), w\rangle \geq 0, \;\; \xi_i, \xi_i^* \geq 0.\end{aligned}$$

Since the additional constraint does not depend on b it is easy to see that the quantile property still holds. The dual optimization problem yields

$$\underset{\alpha,\beta}{\text{minimize}} \;\; \frac{1}{2}\begin{bmatrix}\alpha\\ \beta\end{bmatrix}^\top \begin{bmatrix}K & D_1K\\ D_2K & D_1D_2K\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} - \alpha^\top\vec{y} \tag{13a}$$

$$\text{subject to} \;\; C(\tau-1) \leq \alpha_i \leq C\tau \text{ and } 0 \leq \beta_i \text{ for all } 1 \leq i \leq m \text{ and } \vec{1}^\top\alpha = 0. \tag{13b}$$

Here D_1K is a shorthand for the matrix of entries $D_1k(x_i, x_j)$ and D_2K, D_1D_2K are defined analogously. Here $w = \sum_i \alpha_i\phi(x_i) + \beta_i\psi(x_i)$ or equivalently $f(x) = \sum_i \alpha_i k(x_i, x) + \beta_i D_1k(x_i, x) + b$.

Example Assume that $x \in \mathbb{R}^n$ and that x_1 is the coordinate with respect to which we wish to enforce monotonicity. Moreover, assume that we use a Gaussian RBF kernel, that is

$$k(x, x') = \exp\left(-\frac{1}{2\sigma^2}\left\|x - x'\right\|^2\right). \tag{14}$$

In this case $D_1 = \partial_1$ with respect to x and $D_2 = \partial_1$ with respect to x'. Consequently we have

$$D_1k(x, x') = \frac{x_1' - x_1}{\sigma^2}k(x, x'); D_2k(x, x') = \frac{x_1 - x_1'}{\sigma^2}k(x, x') \tag{15a}$$

$$D_1D_2k(x, x') = \left[\sigma^{-2} - \frac{(x_1 - x_1')^2}{\sigma^4}\right]k(x, x'). \tag{15b}$$

Plugging the values of (15) into (13) yields the quadratic program. Note also that both $k(x, x')$ and $D_1k(x, x')$ in (15a), are used in the function expansion.

If x_1 were drawn from a discrete (yet ordered) domain we could replace D_1, D_2 with a finite difference operator. This is still a linear operation on k and consequently the optimization problem remains unchanged besides a different functional form for D_1k.

An alternative to the above approach is not to modify the optimization problem but to ensure the constraints by modifying the function in the hypothesis space which is much simpler to implement as in Le et al. (2006).

3.3 Other Function Classes

Semiparametric Estimates RKHS expansions may not be the only function classes desired for quantile regression. For instance, in the social sciences a semiparametric model may be more desirable, as it allows for interpretation of the linear coefficients (Gu and Wahba, 1993; Smola et al., 1999; Bickel et al., 1994). In this case we add a set of parametric functions f_i and solve

$$\text{minimize } \frac{1}{m}\sum_{i=1}^{m} l_\tau(y_i - f(x_i)) + \frac{\lambda}{2}\|g\|_{\mathcal{H}}^2 \text{ where } f(x) = g(x) + \sum_{i=1}^{n}\beta_i f_i(x) + b. \tag{16}$$

For instance, the function class f_i could be linear coordinate functions, that is, $f_i(x) = x_i$. The main difference to (6) is that the resulting optimization problem exhibits a larger number of equality constraint. We obtain (6) with the additional constraints

$$\sum_{j=1}^{m} \alpha_j f_i(x_j) = 0 \text{ for all } i. \tag{17}$$

Linear Programming Regularization Convex function classes with ℓ_1 penalties can be obtained by imposing an $\|\alpha\|_1$ penalty instead of the $\|g\|_{\mathcal{H}}^2$ penalty in the optimization problem. The advantage of this setting is that minimizing

$$\text{minimize } \frac{1}{m}\sum_{i=1}^{m} l_\tau(y_i - f(x_i)) + \lambda \sum_{j=1}^{n} |\alpha_i| \text{ where } f(x) = \sum_{i=1}^{n} \alpha_i f_i(x) + b. \tag{18}$$

is a *linear program* which can be solved efficiently by existing codes for large scale problems. In the context of (18) the functions f_i constitute the generators of the convex function class. This approach is similar to Koenker et al. (1994) and Bosch et al. (1995). The former discuss ℓ_1 regularization of expansion coefficients whereas the latter discuss an explicit second order smoothing spline method for the purpose of quantile regression. Most of the discussion in the present paper can be adapted to this case without much modification. For details on how to achieve this see Schölkopf and Smola (2002). Note that smoothing splines are a special instance of kernel expansions where one assumes explicit knowledge of the basis functions.

Relevance Vector Regularization and Sparse Coding Finally, for sparse expansions one can use more aggressive penalties on linear function expansions than those given in (18). For instance, we could use a staged regularization as in the RVM (Tipping, 2001), where a quadratic penalty on each coefficient is exerted with a secondary regularization on the penalty itself. This corresponds to a Student-t penalty on α.

Likewise we could use a mix between an ℓ_1 and ℓ_0 regularizer as used in Fung et al. (2002) and apply successive linear approximation. In short, there exists a large number of regularizers, and (non)parametric families which can be used. In this sense the RKHS parameterization is but one possible choice. Even so, we show in Section 5 that QR using the RKHS penalty yields excellent performance in experiments.

Neural Networks, Generalized Models Our method does not depend on the how the function class is represented (not only the Kernelized version), in fact, one can use Neural Networks or Generalized Models for estimation as long as the loss function is kept the same. This is the main reason why this paper is called *Non-parametric quantile estimation*.

4. Theoretical Analysis

In this section we state some performance bounds for our estimator.

4.1 Performance Indicators

We first need to discuss how to evaluate the performance of the estimate f versus the true conditional quantile $\mu_\tau(x)$. Two criteria are important for a good quantile estimator f_τ:

- f_τ needs to satisfy the quantile property as well as possible. That is, we want that

$$\Pr_{X,Y}\{|\Pr\{y < f_\tau(x)\} - \tau| \geq \varepsilon\} \leq \delta. \tag{19}$$

  In other words, we want that the probability that $y < f_\tau(x)$ does not deviate from τ by more than ε with high probability, when viewed over all draws (X,Y) of training data. Note however, that (19) does not imply having a conditional quantile estimator at all. For instance, the constant function based on the unconditional quantile estimator with respect to Y performs extremely well under this criterion. Hence we need a second quantity to assess how closely $f_\tau(x)$ tracks $\mu_\tau(x)$.

- Since μ_τ itself is not available, we take recourse to (3) and the fact that μ_τ is the minimizer of the expected risk $R[f]$. While this will not allow us to compare μ_τ and f_τ directly, we can at least compare it by assessing how close to the minimum $R[f_\tau^*]$ the estimate $R[f_\tau]$ is. Here f_τ^* is the minimizer of $R[f]$ with respect to the chosen function class. Hence we will strive to bound

$$\Pr_{X,Y}\{R[f_\tau] - R[f_\tau^*] > \varepsilon\} \leq \delta. \tag{20}$$

These statements will be given in terms of the Rademacher complexity of the function class of the estimator as well as some properties of the loss function used in select it. The technique itself is standard and we believe that the bounds can be tightened considerably by the use of *localized* Rademacher averages (Mendelson, 2003), or similar tools for empirical processes. However, for the sake of simplicity, we use the tools from Bartlett and Mendelson (2002), as the key point of the derivation is to describe a new setting rather than a new technique.

4.2 Bounding $R[f_\tau^*]$

Definition 4 (Rademacher Complexity) *Let $X := \{x_1,\ldots,x_m\}$ be drawn iid from $p(x)$ and let $\mathcal{F}$ be a class of functions mapping from (X) to $\mathbb{R}$. Let σ_i be independent uniform $\{\pm 1\}$-valued random variables. Then the Rademacher complexity $\mathcal{R}_m$ and its empirical variant $\hat{\mathcal{R}}_m$ are defined as follows:*

$$\hat{\mathcal{R}}_m(\mathcal{F}) := \mathbf{E}_\sigma\left[\sup_{f\in\mathcal{F}}\left|\frac{2}{m}\sum_1^n \sigma_i f(x_i)\right| \quad \middle| X\right] \text{ and } \mathcal{R}_m(\mathcal{F}) := \mathbf{E}_X\left[\hat{\mathcal{R}}_m(\mathcal{F})\right]. \tag{21}$$

Conveniently, if Φ is a Lipschitz continuous function with Lipschitz constant L, one can show (Bartlett and Mendelson, 2002) that

$$\mathcal{R}_m(\Phi\circ\mathcal{F}) \leq 2L\mathcal{R}_m(\mathcal{F}) \text{ where } \Phi\circ\mathcal{F} := \{g|g = \phi\circ f \text{ and } f\in\mathcal{F}\}. \tag{22}$$

An analogous result exists for empirical quantities bounding $\hat{\mathcal{R}}_m(\Phi\circ\mathcal{F}) \leq 2L\hat{\mathcal{R}}_m(\mathcal{F})$. The combination of (22) with Bartlett and Mendelson (2002, Theorem 8) yields:

Theorem 5 (Concentration for Lipschitz Continuous Functions) *For any Lipschitz continuous function Φ with Lipschitz constant L and a function class $\mathcal{F}$ of real-valued functions on $\mathcal{X}$ and probability measure on $\mathcal{X}$ the following bound holds with probability $1-\delta$ for all draws of X from $\mathcal{X}$:*

$$\sup_{f\in\mathcal{F}}\left|\mathbf{E}_x[\Phi(f(x))] - \frac{1}{m}\sum_{i=1}^m \Phi(f(x_i))\right| \leq 2L\mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{8\log 2/\delta}{m}}. \tag{23}$$

We can immediately specialize the theorem to the following statement about the loss for QR:

Theorem 6 *Denote by f_τ^* the minimizer of the $R[f]$ with respect to $f \in \mathcal{F}$. Moreover assume that all $f \in \mathcal{F}$ are uniformly bounded by some constant B. With the conditions listed above for any sample size m and $0 < \delta < 1$, every quantile regression estimate f_τ satisfies with probability at least $(1-\delta)$*

$$R[f_\tau] - R[f_\tau^*] \leq 2\max L\mathcal{R}_m(\mathcal{F}) + (4+LB)\sqrt{\frac{\log 2/\delta}{2m}} \text{ where } L = \{\tau, 1-\tau\}. \tag{24}$$

Proof We use the standard bounding trick that

$$\begin{aligned} R[f_\tau] - R[f_\tau^*] &\leq \left|R[f_\tau] - R_{\text{emp}}[f_\tau]\right| + R_{\text{emp}}[f_\tau^*] - R[f_\tau^*] && (25)\\ &\leq \sup_{f\in\mathcal{F}} \left|R[f] - R_{\text{emp}}[f]\right| + R_{\text{emp}}[f_\tau^*] - R[f_\tau^*] && (26)\end{aligned}$$

where (25) follows from $R_{\text{emp}}[f_\tau] \leq R_{\text{emp}}[f_\tau^*]$. The first term can be bounded directly by Theorem 5. For the second part we use Hoeffding's bound (Hoeffding, 1963) which states that the deviation between a bounded random variable and its expectation is bounded by $B\sqrt{\frac{\log 1/\delta}{2m}}$ with probability δ. Applying a union bound argument for the two terms with probabilities $2\delta/3$ and $\delta/3$ yields the confidence-dependent term. Finally, using the fact that l_τ is Lipschitz continuous with $L = \max(\tau, 1-\tau)$ completes the proof. ■

Example Assume that $\mathcal{H}$ is an RKHS with radial basis function kernel k for which $k(x,x) = 1$. Moreover assume that for all $f \in \mathcal{F}$ we have $\|f\|_{\mathcal{H}} \leq C$. In this case it follows from Mendelson (2003) that $\mathcal{R}_m(\mathcal{F}) \leq \frac{2C}{\sqrt{m}}$. This means that the bounds of Theorem 6 translate into a rate of convergence of

$$R[f_\tau] - R[f_\tau^*] = O(m^{-\frac{1}{2}}). \tag{27}$$

This is as good as it gets for nonlocalized estimates. Since we do not expect $R[f]$ to vanish except for pathological applications where quantile regression is inappropriate (that is, cases where we have a deterministic dependency between y and x), the use of localized estimates (Bartlett et al., 2002) provides only limited returns. We believe, however, that the constants in the bounds could benefit from considerable improvement.

4.3 Bounds on the Quantile Property

The theorem of the previous section gave us some idea about how far the sample average quantile loss is from its true value under p. We now proceed to stating bounds to which degree f_τ satisfies the quantile property, i.e. (19).

In this view (19) is concerned with the deviation $\mathbf{E}\left[\chi_{(-\infty,0]}(y - f_\tau(x))\right] - \tau$. Unfortunately $\chi_{(-\infty,0]} \circ \mathcal{F}$ is not scale dependent. In other words, small changes in $f_\tau(x)$ around the point $y = f_\tau(x)$ can have large impact on (19). One solution for this problem is to use an artificial margin ε and ramp functions $r_\varepsilon^+, r_\varepsilon^-$ as defined in (28) and Figure 7. These functions are Lipschitz continuous with constant $L = 1/\varepsilon$. This leads to:

$$r_\varepsilon^+(\xi) := \min\{1, \max\{0, 1-\xi/\varepsilon\}\} \quad (28a)$$
$$r_\varepsilon^-(\xi) := \min\{1, \max\{0, -\xi/\varepsilon\}\} \quad (28b)$$

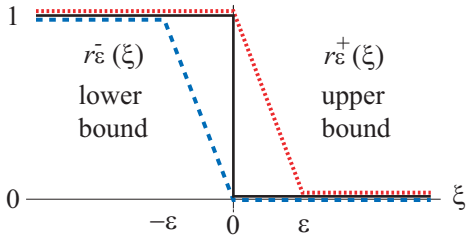


Figure 7: Ramp functions bracketing the characteristic function via $r_\varepsilon^+ \geq \chi_{(-\infty,0]} \geq r_\varepsilon^-$.

Theorem 7 *Under the assumptions of Theorem 6 the expected quantile is bounded with probability* $1-\delta$ *each from above and below by*

$$\frac{1}{m}\sum_{i=1}^m r_\varepsilon^-(y_i - f(x_i)) - \Delta \leq \mathbf{E}\left[\chi_{(-\infty,0]}(y - f_\tau(x))\right] \leq \frac{1}{m}\sum_{i=1}^m r_\varepsilon^+(y_i - f(x_i)) + \Delta, \quad (29)$$

where the statistical confidence term is given by $\Delta = \frac{2}{\varepsilon}\mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{-8\log\delta}{m}}$.

Proof The claim follows directly from Theorem 5 and the Lipschitz continuity of r_ε^+ and r_ε^-. Note that r_ε^+ and r_ε^- minorize and majorize $\xi_{(-\infty,0]}$, which bounds the expectations. Next use a Rademacher bound on the class of loss functions induced by $r_\varepsilon^+ \circ \mathcal{F}$ and $r_\varepsilon^- \circ \mathcal{F}$ and note that the ramp loss has Lipschitz constant $L = 1/\varepsilon$. Finally apply the union bound on upper and lower deviations. ■

Note that Theorem 7 allows for some flexibility: we can decide to use a very conservative bound in terms of ε, i.e. a large value of ε to reap the benefits of having a ramp function with small L. This leads to a lower bound on the Rademacher average of the induced function class. Likewise, a small ε amounts to a potentially tight approximation of the empirical quantile, while risking loose statistical confidence terms.

5. Experiments

The present section mirrors the theoretical analysis of the previous section.

5.1 Experiments with Standard Nonparametric Quantile Regression

We check the performance of various quntile estimators with respect to two criteria:

- Expected risk with respect to the ℓ_τ loss function. Since computing the true conditional quantile is impossible and all approximations of the latter rely on intermediate density estimation, this is the only objective criterion we could find. We denote this loss measure as *pinball loss*.

- Simultaneously we need to ensure that the estimate satisfies the quantile property, that is, we want to ensure that the estimator we obtained does indeed produce numbers $f_\tau(x)$ which exceed y with probability close to τ. The quantile property was measured by *ramp loss*.[4]

5.1.1 MODELS

We compare the following four models:

- An unconditional quantile estimator. Given the simplicity of the function class (constants!) this model should tend to underperform all other estimates in terms of minimizing the empirical risk. By the same token, it should perform best in terms of preserving the quantile property. This appears as `uncond`.

- Linear QR as described in Koenker and Bassett (1978). This uses a linear unregularized model to minimize l_τ. In experiments, we used the `rq` routine available in the *R* package[5] called `quantreg`. This appears as `linear`.

- Nonparametric QR as described by Koenker et al. (1994). This uses a spline model for each coordinate individually, with linear effect. The fitting routine used was `rqss`, also available in `quantreg`.[6] The regularization parameter in this model was chosen by 10-fold cross-validation within the training sample. This appears as `rqss`.

- Nonparametric quantile regression as described in Section 2. We used Gaussian RBF kernels with automatic kernel width (ω^2) and regularization (C) adjustment by 10-fold cross-validation within training sample.[7] This appears as `npqr`.

As we increase the complexity of the function class (from constant to linear to nonparametric) we expect that (subject to good capacity control) the expected risk will decrease. Simultaneously we expect that the quantile property becomes less and less maintained, as the function class grows. This is exactly what one would expect from Theorems 6 and 7. As the experiments show, performance of the `npqr` method is comparable or significantly better than other models. In particular it preserves the quantile property well.

Notes on Gaussian RBF kernel parameter selection trick The parameter σ in the Gaussian kernel could be chosen by the following trick. We fist subsample the training data (if the training data set is not large, use the whole training data), then compute the distance between the points and find the distances at 0.9 and 0.1 quantile of all the distances, the average distance of these two distances is set to be the initial σ_0. This is to guarantee that the kernel parameter is neither too big or too small. Other values of σ to be selected in the experiments (via cross-validation) are $[10^{-4}\sigma_0, \ldots, \sigma_0, \ldots, 10^3\sigma_0, 10^4\sigma_0]$. In general, depending on the problems, one may set the search space to be finer (the distance between two consecutive items in the list is smaller) or coarser (the distance between two consecutive items in the list is larger), or even a higher value for maximum item in the list, and a smaller value for minimum item in the list, etc.

4. In the experiments we set $\varepsilon = 0$ in (28) for simplicity. Thus, it might be appropriate to call it as *step loss* rather than ramp loss. However, we keep to use the term "ramp loss" throughout this paper.
5. See http://cran.r-project.org/.
6. Additional code containing bugfixes and other operations necessary to carry out our experiments is available at http://users.rsise.anu.edu.au/∼timsears.
7. Code will be available as part of the CREST toolbox for research purposes.

5.1.2 DATA SETS

We chose 20 regression data sets from the following R packages: `mlbench, quantreg, alr3` and `MASS`. The first library contains data sets from the UCI repository. The last two were made available as illustrations for regression textbooks. The data sets are all documented and available in *R*. Data sets were chosen not to have any missing variables, to have suitable datatypes, and to be of a size where all models would run on them.[8] In most cases either there was an obvious variable of interest, which was selected as the *y*-variable, or else we chose a continuous variable arbitrarily. The sample sizes vary from $m = 38$ (CobarOre) to $m = 1375$ (heights), and the number of regressors vary from $d = 1$ (5 sets) and $d = 12$ (BostonHousing). Some of the data sets contain categorical variables. We omitted variables which were effectively record identifiers, or obviously produced very small groupings of records. Finally, we *standardized* all data sets coordinatwise to have zero mean and unit variance before running the algorithms. This had a side benefit of putting the pinball loss on similar scale for comparison purposes.

Data Set	Sample Size	No. Regressors (x)	Y Var.	Dropped Vars.
caution	100	2	y	-
ftcollinssnow	93	1	Late	YR1
highway	39	11	Rate	-
heights	1375	1	Dheight	-
sniffer	125	4	Y	-
snowgeese	45	4	photo	-
ufc	372	4	Height	-
birthwt	189	7	bwt	ftv, low
crabs	200	6	CW	index
GAGurine	314	1	GAG	-
geyser	299	1	waiting	-
gilgais	365	8	e80	-
topo	52	2	z	-
BostonHousing	506	13	medv	-
CobarOre	38	2	z	-
engel	235	1	y	-
mcycle	133	1	accel	-
BigMac2003	69	9	BigMac	City
UN3	125	6	Purban	Locality
cpus	209	7	estperf	name

Table 1: Data Set facts

8. The last requirement, using `rqss` proved to be challenging. The underlying spline routines do not allow extrapolation beyond the previously seen range of a coordinate, only permitting interpolation. This does not prevent fitting, but does randomly prevent forecasting on unseen examples, which was part of our performance metric.

5.1.3 RESULTS

We tested the performance of the **4** models. For each model we used 10-fold cross-validation to assess the confidence of our results. As mentioned above, a regularization parameter in `rqss` and ω^2 and C in `npqr` were automatically chosen by 10-fold cross-validation **within** the training sample, i.e. we used *nested* cross-validation. To compare across all four models we measured both *pinball loss* and *ramp loss*. The 20 data sets and three different quantile levels ($\tau \in \{0.1, 0.5, 0.9\}$) yield 60 trials for each model. The full results are shown in Appendix B. In summary, we conclude as follows:

- In terms of *pinball loss*, the performance of our `npqr` were comparable or better than other three models.

  `npqr` performed significantly better than other three models in 14 of the 60 trials, while `rqss` performed significantly better than other three models in only one of the 60 trials. In the rest of 45 trials, no single model performed significantly better then the others. All these statements are based on the two-sided paired-sample t-test with significance level 0.05. We got similar but a bit less conservative results by (nonparametric) Wilcoxon signed rank test.

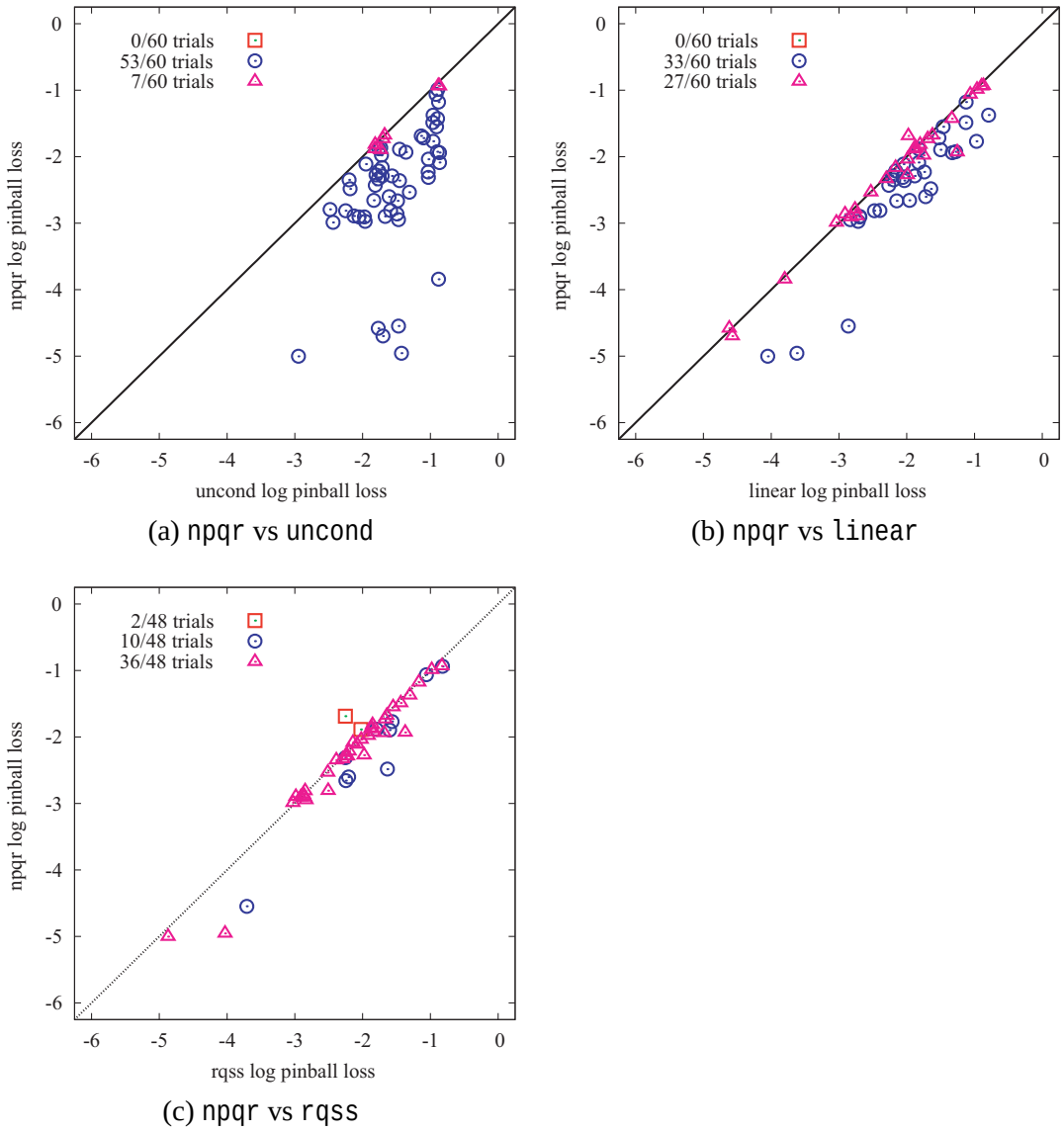
  Figure 8 depicts the comparison of `npqr` performance with each of `uncond`, `linear` and `rqss` models. Each of three plots contain 60 points corresponding to 60 trials (3 different τs times 20 data sets).[9] The vertical axis indicates the log pinball losses of `npqr` and the horizontal axis indicates those of the alternative. The points under (over) the 45 degree line means that the `npqr` was better (worse) than the alternative. Circles (squares) indicate that `npqr` was significantly better (worse) than the alternative at 0.05 significance level in paired-sample t-test, while triangles indicate no significant difference.

- In terms of *ramp loss* (quantile property), the performance of our `npqr` were comparable to other three models for intermediate quantile ($\tau = 0.5$). All four models produced ramp losses close to the desired quantile, although flexible nonparametric models `rqss` and `npqr` were noisier in this regard. When $\tau = 0.5$, the number of $f_\tau(x)$ which exceed y did NOT deviate significantly from the binomial distribution $B(\text{ sample size }, \tau)$ in all 20 data sets.

  On the other hand, for extreme quantiles ($\tau = 0.1$ and 0.9), `rqss` and `npqr` showed a small but significant bias towards the median in a few trials. We conjecture that this bias is related to the problem of *data piling* (Hall et al., 2005). See section 6 for the discussion.

  Note that the quantile property, as such, is not informative measure for *conditional* quantile estimation. It merely measures *unconditional* quantile estimation performances. For example, `uncond`, the constant function based on the unconditional quantile estimator with respect to Y (straightforwardly obtained by sorting $\{y_i\}_{i=1}^m$ without using $\{x_i\}_{i=1}^m$ at all), performed best under this criterion. It is clear that the less flexible model would have the better quantile property, but it does not necessarily mean that those less flexible ones are better for conditional quantile functions.

9. In the comparison between `npqr` and `rqss`, 48 trials were examined since in the other 12 trials `rqss` was unable to produce estimates, due to its construction of the function system.



(a) `npqr` vs `uncond` (b) `npqr` vs `linear`

(c) `npqr` vs `rqss`

Figure 8: Log-log plots of out-of-sample performances. The plots show `npqr` versus (a) `uncond`, (b) `linear` and (c) `rqss`; combining the average pinball losses of all 60 trials (3 quantiles times 20 data sets). The points under (over) the 45 degree line means that the `npqr` was better (worse) than the alternative. Circle (squares) indicate that `npqr` was significantly better (worse) than the alternative at 0.05 significance level in paired-sample t-test, while triangles indicate no significant difference.

5.2 Experiments on Nonparametric Quantile Regression with Additional Constraints

We empirically investigate the performances of nonparametric quantile regression estimator with the additional constraints described in section 3. Imposing constraints is one way to introduce the prior knowledge on the data set being analyzed. Although additional constraints always increase training errors, we will see that these constraints can sometimes reduce test errors. The full results are shown in Appendix B.

5.2.1 NON-CROSSING CONSTRAINTS

First we look at the effect of non-crossing constraints on the generalization performances. We used the same 20 data sets mentioned in the previous subsection. We denote the `npqrs` trained with non-crossing constraints as `noncross` and `npqr` indicates standard one here. We made comparisons between `npqr` and `noncross` with $\tau \in \{0.1, 0.5, 0.9\}$. The results for `noncross` with $\tau = 0.1$ were obtained by training a pair of non-crossing models with $\tau = 0.1$ and 0.2. The results with $\tau = 0.5$ were obtained by training three non-crossing models with $\tau = 0.4$, 0.5 and 0.6. The results with $\tau = 0.9$ were obtained by training a pair of non-crossing models with $\tau = 0.8$ and 0.9. In this experiment, we simply impose non-crossing constraints only at a single test point to be evaluated. The kernel width and smoothing parameter were always set to be the selected ones in the above standard `npqr` experiments. The confidences were assessed by 10-fold cross-validation in the same way as the previous section. The complete results are found in the tables in Appendix B. The performances of `npqr` and `noncross` are quite similar since `npqr` itself could produce *almost* non-crossing estimates and the constraints only make a *small* adjustments only when there happen to be the violations.

5.2.2 MONOTONICITY CONSTRAINTS

We compare two models:

- Nonparametric QR as described in Section 2 (`npqr`).
- Nonparametric QR with monotonicity constraints as described in Section 3.2 (`npqrm`).

We use two data sets:

- The *cars* data set as described in Mammen et al. (2001). Fuel efficiency (in miles per gallon) is studied as a function of engine output.
- The *onions* data set as described in Ruppert and Carroll (2003). log(Yield) is studied as a function of density, we use only the measurements taken at Purnong Landing.

We tested the performance of the two methods on 3 different quantiles ($\tau \in \{0.1, 0.5, 0.9\}$). In the experiments with *cars*, we noticed that the data is not truly monotonic. This is because, smaller engines may correspond to cheap cars and thus may not be very efficient. Monotonic models (`npqrm`) tend to do worse than standard models (`npqr`) for lower quantiles. With higher quantiles, `npqrm` tends to do better than the standard `npqr`. For the `onions` data set, as the data is truly monotonic, the `npqrm` does better than the standard `npqr` in terms of the pinball loss.

6. Discussion and Extensions

Frequently in the literature of regression, including quantile regression, we encounter the term "exploratory data analysis". This is meant to describe a phase before the user has settled on a "model", after which some statistical tests are performed, justifying the choice of the model. Quantile regression, which allows the user to highlight many aspects of the distribution, is indeed a useful tool for this type of analysis. We also note that no attempts at statistical modeling beyond automatic parameter choice via cross-validation, were made to tune the results. So the effort here stays true to that spirit, yet may provide useful estimates immediately.

In the Machine Learning literature the emphasis is more on short circuiting the modeling process. Here the two approaches are complementary. While not completely model-free, the experience of building the models in this paper shows how easy it is to estimate the quantities of interest in QR, with little of the angst of model selection, thanks to regularization. It is interesting to consider whether kernel methods, with regularization, can blur the distinction between model building and data exploration in statistics.

In summary, we have presented a Quadratic Programming method for estimating quantiles which bests the state of the art in statistics. It is easy to implement, comes with uniform convergence results and experimental evidence for its soundness. We also introduce non-crossing and monotonicity constraints as extensions to avoid some undesirable behaviors in some circumstances.

Overly Optimistic Estimates for Ramp Loss The experiments show us that the there is a bias towards the median in terms of the ramp loss. For example, if we run a quantile estimator with $\tau = 0.05$, then we will not necessarily get the empirical quantile is also at 0.05 but more likely to be at 0.08 or higher. Likewise, the empirical quantile will be 0.93 or lower if the estimator is run at 0.9. This affects all estimators, using the pinball loss as the loss function, not just the kernel version.

This is because the algorithm tends to aggressively push a number of points to the kink in the training set, these points may then be miscounted (see Lemma 3). The main reason behind it is that the extreme quantiles tend to be less smooth, the regularizer will therefore makes sure we get a simpler model by biasing towards the median (which is usually simpler). However, in the test set it is very unlikely to get the points lying exactly at the kink. Figure 9 shows us there is a linear relationship between the fraction of points at and below the kink (for low quantiles) and below the kink (for higher quantiles) with the empirical ramp loss.

Accordingly, in order to get a better performance in terms of the ramp loss, we just estimate the quantiles, and if they turn out to be too optimistic on the training set, we use a slightly lower (for $\tau < 0.5$) or higher (for $\tau > 0.5$) value of τ until we have exactly the right quantity.

The fact that there is a number of points sitting exactly on the kink (quantile regression - this paper), the edge of the tube (ν-SVR - see Schölkopf et al., 2000), or the supporting hyperplane (single-class problems and novelty detection - see Schölkopf et al., 1999) might affect the overall accuracy control in the test set. This issue deserves further scrutiny.

Estimation with constraints We introduce non-crossing and monotonicity constraints in the context of nonparametric quantile regression. However, as discussed in Mammen et al. (2001), other constraints can also be applied very similiarly to the constraints described in this paper but might be in different estimation contexts. Here are some variations (we just give directions for the first two, the rest can be applied in the same manner)

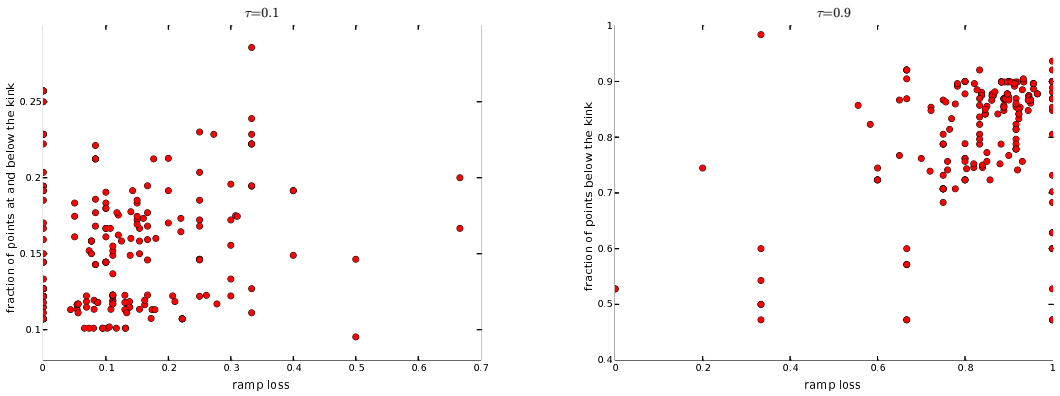


Figure 9: Illustration of the relationship between quantile in training and ramp loss.

- *Bivariate extreme-value distributions*. Hall and Tajvidi (2000) propose methods to estimate the dependence function of a bivariate extreme-value distribution. They require to estimate a **convex** function f such that $f(0) = f(1) = 1$ and $f(x) \geq \max(x, 1-x)$ for $x \in [0,1]$. We can also apply this approach to our method as to the monotonicity constraint, all we have to do is to ensure $\langle \phi(0), w \rangle + b = \langle \phi(1), w \rangle + b = 1$, $\langle \phi''(x), w \rangle \geq 0$ and $\langle \phi(x), w \rangle + b \geq \max(x, 1-x)$ for $x \in [0,1]$.

- *Positivity constraints*. The regression function is positive. In this case, we must ensure $\langle \phi(x), w \rangle + b > 0$, $\forall x$.

- *Boundary conditions*. The regression function is defined in $[a,b]$ and assumed to be v at the boundary point a or b.

- *Additive models with monotone components*. The regression function $f : \mathbb{R}^n \to \mathbb{R}$ is of additive form $f(x_1, ..., x_n) = f_1(x_1) + ... + f_n(x_n)$ where each additive component f_i is monotonic.

- *Observed deriatives*. Assume that m samples are observed corresponding with m regression functions. Now, the constraint is that f_j coincides with the derivative of f_{j-1} (same notation with last point) (Cox, 1988).

Future Work Quantile regression has been mainly used as a data analysis tool to assess the influence of individual variables. This is an area where we expect that nonparametric estimates will lead to better performance.

Being able to estimate an upper bound on a random variable $y|x$ which hold with probability τ is useful when it comes to determining the so-called Value at Risk of a portfolio. Note, however, that in this situation we want to be able to estimate the regression quantile for a large set of different portfolios. For example, an investor may try to optimize their portfolio allocation to maximize return while keeping risk within a constant bound. Such uniform statements will need further analysis if we are to perform nonparametric estimates. We need more efficient optimization algorithm for non-crossing constraints since we have to work with $o(nm)$ dual variables. Simple SVM (Vishwanathan et al., 2003) would be a promising candidate for this purpose.

Acknowledgments

National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. This work was supported by grants of the ARC, by the Pascal Network of Excellence and by Japanese Grants-in-Aid for Scientific Research 16700258. We thank Roger Koenker for providing us with the latest version of the *R* package `quantreg`, and for technical advice. We thank Shahar Mendelson and Bob Williamson for useful discussions and suggestions. We also thank the anonymous reviewers for valuable feedback.

Appendix A. Nonparametric ν-Support Vector Regression

In this section we explore an alternative to the quantile regression framework proposed in Section 2. It derives from Schölkopf et al. (2000). There the authors suggest a method for adapting SV regression and classification estimates such that automatically only a quantile ν lies beyond the desired confidence region. In particular, if $p(y|x)$ can be modeled by additive noise of equal degree (i.e. $y = f(x) + \xi$ where ξ is a random variable independent of x) Schölkopf et al. (2000) show that the ν-SV regression estimate does converge to a quantile estimate.

A.1 Heteroscedastic Regression

Whenever the above assumption on $p(y|x)$ is violated ν-SVR will not perform as desired. This problem can be amended as follows: one needs to turn the margin $\varepsilon(x)$ into a nonparametric estimate itself. This means that we solve the following optimization problem.

$$\begin{aligned}
\underset{\theta_1,\theta_2,b,\varepsilon}{\text{minimize}} \quad & \frac{\lambda_1}{2}\|\theta_1\|^2 + \frac{\lambda_2}{2}\|\theta_2\|^2 + \sum_{i=1}^{m}(\xi_i + \xi_i^*) - \nu m\varepsilon && \text{(30a)}\\
\text{subject to} \quad & \langle\phi_1(x_i),\theta_1\rangle + b - y_i \le \varepsilon + \langle\phi_2(x_i),\theta_2\rangle + \xi_i && \text{(30b)}\\
& y_i - \langle\phi_1(x_i),\theta_1\rangle - b \le \varepsilon + \langle\phi_2(x_i),\theta_2\rangle + \xi_i^* && \text{(30c)}\\
& \xi_i, \xi_i^* \ge 0 && \text{(30d)}
\end{aligned}$$

Here ϕ_1, ϕ_2 are feature maps, θ_1, θ_2 are corresponding parameters, ξ_i, ξ_i^* are slack variables and b, ε are scalars. The key difference to the heteroscedastic estimation problem described in Schölkopf et al. (2000) is that in the latter the authors assume that the specific form of the noise is *known*. In (30) instead, we make no such assumption and instead we estimate $\varepsilon(x)$ as $\langle\phi_2(x),\theta_2\rangle + \varepsilon$.

One may check that the dual of (30) is obtained by

$$\begin{aligned}
\underset{\alpha,\alpha^*}{\text{minimize}} \quad & \frac{1}{2\lambda_1}(\alpha-\alpha^*)^\top K_1(\alpha-\alpha^*) + \frac{1}{2\lambda_2}(\alpha+\alpha^*)^\top K_1(\alpha+\alpha^*) + (\alpha-\alpha^*)^\top y && \text{(31a)}\\
\text{subject to} \quad & \vec{1}^\top(\alpha-\alpha^*) = 0 && \text{(31b)}\\
& \vec{1}^\top(\alpha+\alpha^*) = Cm\nu && \text{(31c)}\\
& 0 \le \alpha_i, \alpha_i^* \le 1 \text{ for all } 1 \le i \le m && \text{(31d)}
\end{aligned}$$

Here K_1, K_2 are kernel matrices where $[K_i]_{jl} = k_i(x_j, x_l)$ and $\vec{1}$ denotes the vector of ones. Moreover, we have the usual kernel expansion, this time for the regression $f(x)$ and the margin $\varepsilon(x)$ via

$$f(x) = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)\,k_1(x_i,x) + b \text{ and } \varepsilon(x) = \sum_{i=1}^{m}(\alpha_i + \alpha_i^*)\,k_2(x_i,x) + \varepsilon. \tag{32}$$

The scalars b and ε can be computed conveniently as dual variables of (31) when solving the problem with an interior point code (see Schölkopf and Smola, 2002, for more details).

A.2 The ν-Property

As in the parametric case also (30) has the ν-property. However, it is worth noting that the solution $\varepsilon(x)$ need not be positive throughout unless we change the optimization problem slightly by imposing a nonnegativity constraint on ε. The following theorem makes this reasoning more precise:

Theorem 8 *The minimizer of (30) satisfies*

1. *The fraction of points for which $|y_i - f(x_i)| < \varepsilon(x_i)$ is bounded by $1-\nu$.*
2. *The fraction of constraints (30b) and (30c) with $\xi_i > 0$ or $\xi_i^* > 0$ is bounded from above by ν.*
3. *If (x,y) is drawn iid from a distribution $\Pr(x,y)$, with $\Pr(y|x)$ continuous and the expectation of the modulus of absolute continuity of its density satisfying $\lim_{\delta\to 0}\mathbf{E}[\varepsilon(\delta)] = 0$. With probability 1, asymptotically, the fraction of points satisfying $|y_i - f(x_i)| = \varepsilon(x_i)$ converges to 0.*

Moreover, imposing $\varepsilon \geq 0$ is equivalent to relaxing (31c) to $\vec{1}^\top(\alpha - \alpha^*) \leq Cm\nu$. If in addition K_2 has only nonnegative entries then also $\varepsilon(x) \geq 0$ for all x_i.

Proof The proof is essentially similar to that of Lemma 3 and Schölkopf et al. (2000). However note that the flexibility in ε and potential $\varepsilon(x) < 0$ lead to additional complications. However, if both f and $\varepsilon(x)$ have well behaved entropy numbers, then also $f \pm \varepsilon$ are well behaved.

To see the last set of claims note that the constraint $\vec{1}^\top(\alpha - \alpha^*) \leq Cm\nu$ is obtained again directly from dualization via the condition $\varepsilon \geq 0$. Since $\alpha_i, \alpha_i^* \geq 0$ for all i it follows that $\varepsilon(x)$ contains only nonnegative coefficients, which proves the last part of the claim. ■

Note that in principle we could enforce $\varepsilon(x_i) \geq 0$ for all x_i. This way, however, we would lose the ν-property and add even more complication to the optimization problem. A third set of Lagrange multipliers would have to be added to the optimization problem.

A.3 An Example

The above derivation begs the question why one should not use (31) instead of (6) for the purpose of quantile regression. After all, both estimators yield an estimate for the upper and lower quantiles.

Firstly, the combined approach is numerically more costly as it requires optimization over twice the number of parameters, albeit at the distinct advantage of a sparse solution, whereas (6) always leads to a dense solution.

The key difference, however, is that (31) is prone to producing estimates where the margin $\varepsilon(x) < 0$. While such a solution is clearly unreasonable, it occurs whenever the margin is rather small and the overall tradeoff of simple f vs. simple ε yields an advantage by keeping f simple. With enough data this effect vanishes, however, it occurs quite frequently, even with supposedly distant quantiles, as can be seen in Figure 10.

In addition, the latter suffers from the assumption that the error be symmetrically distributed. In other words, if we are just interested in obtaining the 0.95 quantile estimate we end up estimating

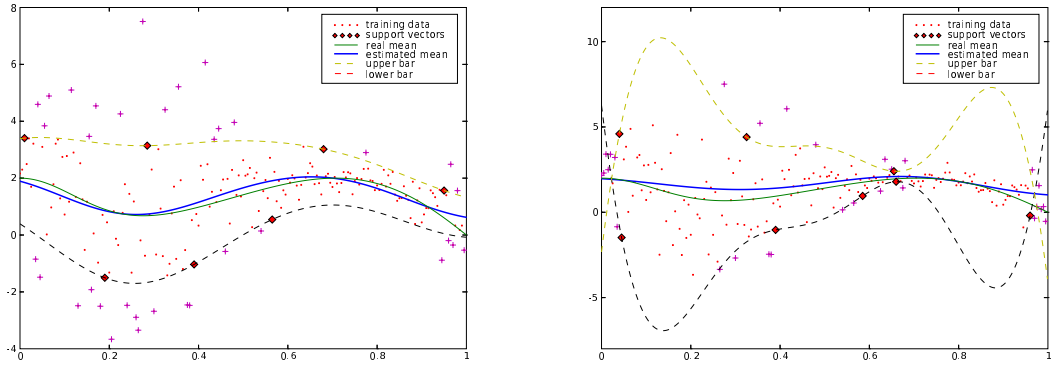


Figure 10: Illustration of the heteroscedastic SVM regression on artificial data set generated from (1) with $f(x) = \sin \pi x$ and $\sigma(x) = \exp(\sin 2\pi x)$. On the left, $\lambda_1 = 1$, $\lambda_2 = 10$ and $\nu = 0.2$, the algorithm successfully regresses the data. On the right, $\lambda_1 = 1$, $\lambda_2 = 0.1$ and $\nu = 0.2$, the algorithm fails to regress the data as ε becomes negative.

the 0.05 quantile on the way. In addition to that, we make the assumption that the additive noise is symmetric.

We produced this derivation and experiments mainly to make the point that the adaptive margin approach of Schölkopf et al. (2000) is insufficient to address the problems posed by quantile regression. We found empirically that it is much easier to adjust QR instead of the symmetric variant.

In summary, the symmetric approach is probably useful only for parametric estimates where the number of parameters is small and where the expansion coefficients ensure that $\varepsilon(x) \geq 0$ for all x.

Appendix B. Experimental Results

In this appendix, we show the detail results on the experiments.

B.1 Standard Nonparametric Quantile Regression

Here we assemble six tables to display the comparisons among four models, `uncond`, `linear`, `rqss` and `npqr`. Each table represents *pinball loss* or *ramp loss* for each of $\tau = 0.1$, 0.5 and 0.9 cases.

	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
Pinball Loss	Table 2	Table 4	Table 6
Ramp Loss	Table 3	Table 5	Table 7

Tables 2, 4, and 6 show the average pinball loss for each data set. A lower figure is preferred in each case. The bold figures indicate the best (smallest) performances. The circles '$\circ$' indicate that the difference from the second best model were statistically significant at 0.05 level with two-sided paired-sample t-test. NA denotes cases where rqss (Koenker et al., 1994) was unable to produce estimates, due to its construction of the function system.

Tables 3, 5 and 7, show the ramp loss, a measure for quantile property. In each table a figure close to the intended quantile (10, 50 or 90) is preferred. The figures in round brackets denote the

p-values under the null-hypothesis that the ramp loss, i.e. the number of test points (x,y) such that $y < f_\tau(x)$, is a sample from binomial distribution $B(\text{ sample size}, \tau)$. The bold figures indicate the best (closest to the intended quantile τ) performances. The bullets '$\bullet$' indicate that the ramp loss deviate significantly from binomial distribution $B(\text{ sample size}, \tau)$.

B.2 Nonparametric Quantile Regression with Constraints

Next, we show the results on constrained nonparameteric quantile regression.

B.2.1 NON-CROSSING CONSTRAINTS

Table 8 shows the average pinball loss comparison between the nonparametric quantile regression without (`npqr`) and with (`noncross`) non-crossing constraints. The bold figures indicate the better (smaller) performances The circles '$\circ$' indicate that the difference were statistically significant at 0.05 level with two-sided paired-sample t-test.

Table 9 shows the ramp loss, a measure for quantile property, of `npqr` and `noncross`. The figures in round brackets denote the p-values under the null-hypothesis that the ramp loss, i.e. the number of test points (x,y) such that $y < f_\tau(x)$, is a sample from binomial distribution $B(\text{ sample size}, \tau)$. The bold figures indicate the better (closeer to the intended quantile τ) performances. The bullets '$\bullet$' indicate that the ramp loss deviated significantly from binomial distribution $B(\text{ sample size}, \tau)$.

B.2.2 MONOTONICITY CONSTRAINTS

We tested on the cars and the onions data set for monotonicity with respect to engine size and diameter respectively. Note that on the engines data set the monotonicity constraint is not perfectly satisfied. Table 10 shows the average pinball loss comparison between the nonparametric quantile regression without (`npqr`) and with (`npqrm`) monotonicity constraints. See above for the notation of the table. Table 11 shows the ramp loss, a measure for quantile property, of `npqr` and `npqrm`. See above for the notation of the table.

data set	uncond	linear	rqss	npqr
caution	11.09 ± 0.95	11.18 ± 1.04	**9.18 ± 0.93**	9.56 ± 0.92
ftcollinssnow	16.28 ± 1.18	16.48 ± 1.19	**15.68 ± 1.33**	16.24 ± 1.17
highway	11.27 ± 1.48	19.32 ± 5.11	19.51 ± 4.44	∘ **8.34 ± 1.18**
heights	17.20 ± 0.44	15.28 ± 0.39	15.27 ± 0.40	**15.26 ± 0.39**
sniffer	13.92 ± 0.99	6.78 ± 0.68	**5.44 ± 0.58**	5.48 ± 0.64
snowgeese	8.74 ± 1.44	**4.79 ± 0.89**	4.85 ± 0.90	5.03 ± 0.87
ufc	17.06 ± 0.72	10.02 ± 0.42	10.11 ± 0.44	**9.70 ± 0.42**
birthwt	18.29 ± 1.39	18.44 ± 1.24	18.85 ± 1.28	**17.68 ± 1.16**
crabs	18.27 ± 0.97	1.03 ± 0.08	NA	**0.91 ± 0.07**
GAGurine	10.53 ± 0.55	8.39 ± 0.41	**5.79 ± 0.43**	6.00 ± 0.63
geyser	17.15 ± 0.52	11.50 ± 0.49	11.10 ± 0.49	**10.91 ± 0.49**
gilgais	12.84 ± 0.49	5.93 ± 0.40	5.75 ± 0.44	**5.46 ± 0.35**
topo	20.41 ± 2.45	9.12 ± 1.32	8.15 ± 1.30	**6.03 ± 0.91**
BostonHousing	14.05 ± 0.56	6.60 ± 0.34	NA	∘ **5.10 ± 0.42**
CobarOre	17.88 ± 2.28	17.36 ± 1.97	14.71 ± 2.20	**13.80 ± 2.70**
engel	11.92 ± 0.65	6.49 ± 0.79	5.68 ± 0.45	**5.55 ± 0.37**
mcycle	19.99 ± 0.86	17.87 ± 0.98	10.98 ± 0.66	∘ **7.39 ± 0.90**
BigMac2003	8.37 ± 1.17	6.31 ± 0.95	NA	**6.13 ± 0.96**
UN3	18.02 ± 1.06	**11.47 ± 0.97**	NA	**11.47 ± 1.02**
cpus	5.25 ± 0.69	1.74 ± 0.34	0.77 ± 0.18	**0.67 ± 0.23**

Table 2: Method Comparison: Pinball Loss ($\times 100$, $\tau = 0.1$)

data set	uncond	linear	rqss	npqr
caution	**11.00** (**0.59**)	12.00 (0.40)	• 16.00 (0.04)	12.00 (0.40)
ftcollinssnow	**10.00** (**0.91**)	11.10 (0.65)	12.20 (0.44)	12.20 (0.44)
highway	**10.80** (**0.70**)	• 20.00 (0.03)	• 26.70 (0.00)	• 20.00 (0.03)
heights	9.60 (0.66)	**10.00** (**0.92**)	**10.00** (**0.92**)	**10.00** (**0.92**)
sniffer	7.80 (0.57)	13.70 (0.15)	**12.00** (**0.37**)	• 15.90 (0.02)
snowgeese	12.50 (0.32)	**9.70** (**0.95**)	**9.70** (**0.95**)	13.60 (0.32)
ufc	9.70 (0.92)	**9.90** (**0.94**)	11.80 (0.21)	10.50 (0.68)
birthwt	**10.00** (**0.86**)	12.00 (0.27)	12.60 (0.18)	11.60 (0.38)
crabs	**10.00** (**0.88**)	12.00 (0.29)	NA	13.30 (0.09)
GAGurine	10.40 (0.68)	**9.90** (**0.96**)	10.70 (0.55)	12.10 (0.19)
geyser	**9.70** (**0.96**)	11.20 (0.48)	10.70 (0.60)	12.20 (0.21)
gilgais	9.50 (0.88)	**10.40** (**0.71**)	• 13.50 (0.03)	12.40 (0.12)
topo	**8.90** (**0.84**)	13.40 (0.29)	16.00 (0.14)	• 19.40 (0.03)
BostonHousing	**9.70** (**0.89**)	11.50 (0.24)	NA	• 15.00 (0.00)
CobarOre	**8.50** (**0.93**)	12.70 (0.35)	16.10 (0.16)	16.10 (0.16)
engel	**10.20** (**0.81**)	9.40 (0.85)	**10.20** (**0.81**)	12.20 (0.20)
mcycle	**10.00** (**0.92**)	11.50 (0.51)	11.40 (0.51)	12.00 (0.35)
BigMac2003	**9.00** (**0.92**)	• 18.00 (0.04)	NA	14.30 (0.16)
UN3	9.50 (0.97)	12.00 (0.37)	NA	**10.30** (**0.74**)
cpus	**9.40** (**0.95**)	12.20 (0.29)	• 15.30 (0.01)	• 19.10 (0.00)

Table 3: Method Comparison: Ramp Loss ($\times 100, \tau = 0.1$)

data set	uncond	linear	rqss	npqr
caution	38.13 ± 3.44	32.40 ± 2.91	23.76 ± 2.74	**22.56 ± 2.68**
ftcollinssnow	42.10 ± 2.95	40.82 ± 2.95	44.07 ± 3.24	**39.08 ± 3.09**
highway	38.35 ± 6.34	45.39 ± 7.04	27.17 ± 3.26	**25.33 ± 3.62**
heights	40.08 ± 0.81	**34.50 ± 0.72**	34.66 ± 0.72	34.53 ± 0.72
sniffer	35.74 ± 3.13	12.78 ± 1.11	10.50 ± 0.98	∘ **9.92 ± 0.94**
snowgeese	32.08 ± 6.33	13.85 ± 3.46	**10.49 ± 2.53**	18.50 ± 4.96
ufc	40.21 ± 1.55	23.20 ± 0.95	21.23 ± 0.90	**21.22 ± 0.90**
birthwt	41.05 ± 2.14	38.15 ± 1.96	37.55 ± 2.08	**37.19 ± 1.96**
crabs	41.52 ± 1.99	2.24 ± 0.13	NA	**2.14 ± 0.12**
GAGurine	40.75 ± 1.81	27.87 ± 1.46	16.02 ± 1.20	**14.57 ± 1.11**
geyser	41.57 ± 1.84	32.50 ± 1.23	31.03 ± 1.36	**30.75 ± 1.40**
gilgais	42.10 ± 1.51	16.12 ± 1.01	**11.72 ± 0.69**	12.40 ± 0.66
topo	42.17 ± 3.86	26.51 ± 2.71	18.58 ± 2.65	**14.39 ± 1.65**
BostonHousing	35.57 ± 1.60	17.50 ± 0.95	NA	∘ **10.76 ± 0.61**
CobarOre	41.37 ± 4.97	41.93 ± 5.20	43.61 ± 4.59	**39.29 ± 6.69**
engel	35.75 ± 2.33	13.72 ± 1.14	13.25 ± 0.92	**13.01 ± 0.85**
mcycle	38.38 ± 3.04	37.88 ± 2.76	20.87 ± 1.52	∘ **17.06 ± 1.42**
BigMac2003	33.24 ± 5.12	21.75 ± 2.85	NA	∘ **17.89 ± 3.05**
UN3	40.79 ± 2.61	26.32 ± 1.70	NA	**23.96 ± 1.84**
cpus	23.00 ± 3.30	5.73 ± 1.04	2.45 ± 0.61	∘ **1.06 ± 0.17**

Table 4: Method Comparison: Pinball Loss ($\times 100$, $\tau = 0.5$)

data set	uncond	linear	rqss	npqr
caution	52.00 (0.62)	**49.00** (**0.92**)	**51.00** (**0.76**)	**49.00** (**0.92**)
ftcollinssnow	50.60 (0.84)	**49.70** (**1.00**)	48.60 (0.84)	51.40 (0.68)
highway	**48.30** (**1.00**)	44.20 (0.52)	45.00 (0.75)	41.70 (0.34)
heights	49.30 (0.63)	**50.10** (**0.91**)	49.80 (0.91)	50.30 (0.79)
sniffer	47.80 (0.72)	**51.00** (**0.72**)	**51.00** (**0.72**)	51.30 (0.72)
snowgeese	48.10 (1.00)	49.20 (1.00)	51.70 (0.77)	**50.60** (**0.77**)
ufc	49.20 (0.80)	**50.00** (**0.96**)	51.60 (0.50)	50.60 (0.80)
birthwt	48.90 (0.77)	**50.00** (**0.88**)	47.80 (0.56)	50.30 (0.88)
crabs	49.50 (0.94)	50.50 (0.83)	NA	**50.00** (**0.94**)
GAGurine	49.20 (0.78)	50.90 (0.69)	51.40 (0.61)	**49.80** (**0.96**)
geyser	48.60 (0.64)	**49.80** (**1.00**)	49.50 (0.91)	49.20 (0.82)
gilgais	48.70 (0.68)	**50.00** (**0.92**)	49.70 (0.92)	50.70 (0.75)
topo	**47.70** (**0.89**)	**47.70** (**0.89**)	**47.70** (**0.89**)	54.80 (0.49)
BostonHousing	**49.70** (**0.89**)	49.60 (0.89)	NA	51.70 (0.40)
CobarOre	46.40 (0.87)	44.50 (0.63)	**47.90** (**0.87**)	59.40 (0.14)
engel	50.90 (0.70)	49.70 (1.00)	49.60 (1.00)	**50.00** (**0.90**)
mcycle	**49.10** (**0.86**)	51.30 (0.73)	51.40 (0.73)	48.80 (0.86)
BigMac2003	49.30 (1.00)	**50.00** (**0.81**)	NA	44.20 (0.34)
UN3	**49.40** (**1.00**)	**50.60** (**0.86**)	NA	48.60 (0.86)
cpus	49.20 (0.89)	51.30 (0.68)	**49.70** (**1.00**)	51.80 (0.58)

Table 5: Method Comparison: Ramp Loss ($\times 100, \tau = 0.5$)

data set	uncond	linear	rqss	npqr
caution	23.35 ± 3.19	15.04 ± 1.54	∘ **13.19** ± **1.57**	15.16 ± 1.76
ftcollinssnow	18.71 ± 1.21	19.77 ± 1.76	19.35 ± 1.90	**18.67** ± **1.74**
highway	25.67 ± 3.71	28.49 ± 6.75	25.34 ± 6.09	**14.48** ± **3.53**
heights	17.63 ± 0.47	**15.47** ± **0.39**	15.50 ± 0.39	**15.47** ± **0.39**
sniffer	23.01 ± 3.62	5.87 ± 0.43	5.88 ± 0.44	∘ **5.25** ± **0.40**
snowgeese	26.94 ± 6.93	7.97 ± 2.67	8.09 ± 3.52	**7.94** ± **2.61**
ufc	18.05 ± 0.96	10.94 ± 0.45	10.84 ± 0.56	**10.15** ± **0.53**
birthwt	16.21 ± 1.03	16.17 ± 1.03	16.53 ± 1.19	∘ **15.20** ± **0.91**
crabs	17.09 ± 0.90	**0.99** ± **0.07**	NA	1.02 ± 0.08
GAGurine	20.86 ± 0.67	15.22 ± 0.83	10.51 ± 1.17	**10.13** ± **1.05**
geyser	14.21 ± 0.72	12.92 ± 0.67	12.48 ± 0.63	**12.10** ± **0.61**
gilgais	18.83 ± 0.72	6.74 ± 0.49	**5.06** ± **0.37**	5.51 ± 0.37
topo	16.50 ± 2.40	13.67 ± 2.80	13.84 ± 3.04	**10.30** ± **2.17**
BostonHousing	22.68 ± 1.28	11.67 ± 0.95	NA	∘ **6.96** ± **0.63**
CobarOre	17.63 ± 2.06	22.28 ± 3.43	20.16 ± 2.92	**15.01** ± **2.12**
engel	22.44 ± 2.57	**5.44** ± **0.43**	5.64 ± 0.65	5.70 ± 0.57
mcycle	15.97 ± 1.21	14.06 ± 1.00	10.58 ± 0.89	∘ **7.02** ± **0.56**
BigMac2003	23.29 ± 4.97	13.06 ± 2.20	NA	∘ **9.45** ± **2.85**
UN3	16.36 ± 1.00	10.37 ± 0.73	NA	∘ **8.81** ± **0.61**
cpus	24.01 ± 4.26	2.67 ± 0.26	1.78 ± 0.72	**0.71** ± **0.17**

Table 6: Method Comparison: Pinball Loss ($\times 100$, $\tau = 0.9$)

data set	uncond	linear	rqss	npqr
caution	**90.00** (**0.90**)	**90.00** (**0.90**)	89.00 (0.83)	89.00 (0.83)
ftcollinssnow	**90.30** (**0.82**)	89.20 (0.91)	88.30 (0.65)	89.20 (0.91)
highway	**89.20** (**0.89**)	• 64.20 (0.00)	• 61.70 (0.00)	• 70.00 (0.00)
heights	89.50 (0.58)	**90.00** (**0.94**)	89.80 (0.85)	90.10 (0.87)
sniffer	**89.40** (**0.97**)	87.60 (0.53)	86.80 (0.37)	84.60 (0.09)
snowgeese	**88.90** (**0.95**)	85.00 (0.32)	85.00 (0.32)	83.90 (0.32)
ufc	**89.80** (**0.94**)	90.30 (0.79)	88.50 (0.36)	88.30 (0.28)
birthwt	88.70 (0.68)	87.60 (0.38)	88.00 (0.38)	**88.90** (**0.68**)
crabs	**89.00** (**0.70**)	87.00 (0.20)	NA	87.10 (0.20)
GAGurine	89.50 (0.82)	**89.80** (**0.96**)	89.40 (0.82)	87.80 (0.25)
geyser	88.50 (0.48)	89.40 (0.74)	**90.40** (**0.81**)	89.10 (0.60)
gilgais	**89.10** (**0.59**)	88.30 (0.30)	87.10 (0.09)	• 83.90 (0.00)
topo	**89.10** (**0.84**)	87.10 (0.52)	85.70 (0.52)	• 77.70 (0.01)
BostonHousing	**90.10** (**0.89**)	88.80 (0.38)	NA	• 80.30 (0.00)
CobarOre	**89.10** (**0.93**)	85.80 (0.66)	79.10 (0.06)	85.80 (0.66)
engel	88.90 (0.65)	**90.00** (**0.85**)	89.10 (0.65)	89.40 (0.81)
mcycle	88.60 (0.70)	**88.80** (**0.70**)	87.70 (0.51)	86.20 (0.23)
BigMac2003	**89.30** (**0.92**)	84.30 (0.16)	NA	• 77.70 (0.01)
UN3	**88.00** (**0.53**)	86.70 (0.24)	NA	85.80 (0.15)
cpus	**89.30** (**0.87**)	87.80 (0.40)	• 82.60 (0.00)	• 82.10 (0.00)

Table 7: Method Comparison: Ramp Loss ($\times 100, \tau = 0.9$)

	$\tau = 0.1$		$\tau = 0.5$		$\tau = 0.9$	
data set	npqr	noncross	npqr	noncross	npqr	noncross
caution	9.56 ± 0.92	**9.55 ± 0.92**	22.56 ± 2.68	**22.51 ± 2.68**	15.16 ± 1.76	**15.15 ± 1.76**
ftcollinssnow	**16.24 ± 1.17**	**16.24 ± 1.17**	39.08 ± 3.09	**38.81 ± 3.09**	**18.67 ± 1.74**	**18.67 ± 1.74**
highway	8.34 ± 1.18	**8.20 ± 1.20**	25.33 ± 3.62	**25.30 ± 3.57**	14.48 ± 3.53	**14.41 ± 3.53**
heights	**15.26 ± 0.39**	15.27 ± 0.39	**34.53 ± 0.72**	34.54 ± 0.72	**15.47 ± 0.39**	15.48 ± 0.39
sniffer	5.48 ± 0.64	**5.43 ± 0.64**	9.92 ± 0.94	**9.91 ± 0.94**	5.25 ± 0.40	**5.19 ± 0.40**
snowgeese	**5.03 ± 0.87**	**5.03 ± 0.87**	**18.50 ± 4.96**	18.59 ± 4.98	7.94 ± 2.61	**7.88 ± 2.62**
ufc	**9.70 ± 0.42**	**9.70 ± 0.39**	**21.22 ± 0.90**	21.23 ± 0.90	10.15 ± 0.53	**9.92 ± 0.49**
birthwt	**17.68 ± 1.16**	17.69 ± 1.16	**37.19 ± 1.96**	37.21 ± 1.96	**15.20 ± 0.91**	**15.20 ± 0.91**
crabs	**0.91 ± 0.07**	**0.91 ± 0.07**	**2.14 ± 0.12**	**2.14 ± 0.12**	1.02 ± 0.08	**1.01 ± 0.08**
GAGurine	6.00 ± 0.63	**5.99 ± 0.63**	**14.57 ± 1.11**	**14.57 ± 1.11**	**10.13 ± 1.05**	**10.13 ± 1.05**
geyser	**10.91 ± 0.49**	**10.91 ± 0.49**	30.75 ± 1.40	**30.71 ± 1.40**	**12.10 ± 0.61**	12.11 ± 0.61
gilgais	**5.46 ± 0.35**	**5.46 ± 0.35**	12.40 ± 0.66	**12.37 ± 0.66**	**5.51 ± 0.37**	**5.51 ± 0.37**
topo	**6.03 ± 0.91**	6.04 ± 0.91	∘ **14.39 ± 1.65**	15.54 ± 1.62	10.30 ± 2.17	**10.21 ± 2.16**
BostonHousing	5.10 ± 0.42	**5.04 ± 0.42**	10.76 ± 0.61	**10.73 ± 0.61**	6.96 ± 0.63	∘ **6.85 ± 0.62**
CobarOre	13.80 ± 2.70	**13.66 ± 2.63**	**39.29 ± 6.69**	40.00 ± 6.61	∘ **15.01 ± 2.12**	15.13 ± 2.12
engel	**5.55 ± 0.37**	**5.55 ± 0.37**	13.01 ± 0.85	**12.96 ± 0.85**	**5.70 ± 0.57**	**5.70 ± 0.57**
mcycle	**7.39 ± 0.90**	**7.39 ± 0.90**	17.06 ± 1.42	**17.03 ± 1.42**	7.02 ± 0.56	**7.00 ± 0.55**
BigMac2003	**6.13 ± 0.96**	6.36 ± 1.02	17.89 ± 3.05	∘ **17.72 ± 3.05**	**9.45 ± 2.85**	9.48 ± 2.84
UN3	**11.47 ± 1.02**	11.52 ± 1.04	23.96 ± 1.84	**23.81 ± 1.81**	**8.81 ± 0.61**	8.82 ± 0.61
cpus	∘ **0.67 ± 0.23**	1.30 ± 0.18	∘ **1.06 ± 0.17**	1.35 ± 0.17	∘ **0.71 ± 0.17**	0.87 ± 0.18

Table 8: Pinball loss comparison between the nonparametric quantile regression without (npqr) and with (noncross) non-crossing constraints.

	$\tau = 0.1$		$\tau = 0.5$		$\tau = 0.9$	
data set	npqr	noncross	npqr	noncross	npqr	noncross
caution	**12.00 (0.40)**	**12.00 (0.40)**	**49.00 (0.92)**	**49.00 (0.92)**	**89.00 (0.83)**	**89.00 (0.83)**
ftcollinssnow	**12.20 (0.44)**	**12.20 (0.44)**	**51.40 (0.68)**	**51.40 (0.68)**	**89.20 (0.91)**	**89.20 (0.91)**
highway	• 20.00 (0.03)	• **13.30 (0.03)**	41.70 (0.34)	**45.00 (0.34)**	• **70.00 (0.00)**	• 56.70 (0.00)
heights	**10.00 (0.92)**	9.90 (0.92)	**50.30 (0.79)**	**50.30 (0.79)**	**90.10 (0.87)**	**90.10 (0.87)**
sniffer	• **15.90 (0.02)**	• **15.90 (0.02)**	**51.30 (0.72)**	**51.30 (0.72)**	84.60 (0.09)	**85.40 (0.09)**
snowgeese	**13.60 (0.32)**	**13.60 (0.32)**	**50.60 (0.77)**	**50.60 (0.77)**	**83.90 (0.32)**	**83.90 (0.32)**
ufc	**10.50 (0.68)**	10.70 (0.68)	**50.60 (0.80)**	**50.60 (0.80)**	**88.30 (0.28)**	88.20 (0.28)
birthwt	11.60 (0.38)	**10.00 (0.38)**	50.30 (0.88)	**50.20 (0.88)**	**88.90 (0.68)**	**88.90 (0.68)**
crabs	13.30 (0.09)	**13.00 (0.09)**	**50.00 (0.94)**	49.50 (0.94)	**87.10 (0.20)**	87.00 (0.20)
GAGurine	12.10 (0.19)	**11.60 (0.19)**	49.80 (0.96)	**49.90 (0.96)**	87.80 (0.25)	**88.10 (0.25)**
geyser	12.20 (0.21)	**12.10 (0.21)**	49.20 (0.82)	**49.60 (0.82)**	**89.10 (0.60)**	89.00 (0.60)
gilgais	**12.40 (0.12)**	**12.40 (0.12)**	**50.70 (0.75)**	50.80 (0.75)	• 83.90 (0.00)	• **84.20 (0.00)**
topo	• **19.40 (0.03)**	• **19.40 (0.03)**	**54.80 (0.49)**	56.30 (0.49)	• **77.70 (0.01)**	• **77.70 (0.01)**
BostonHousing	• **15.00 (0.00)**	• 15.10 (0.00)	51.70 (0.40)	**51.50 (0.40)**	• 80.30 (0.00)	• **80.80 (0.00)**
CobarOre	**16.10 (0.16)**	**16.10 (0.16)**	**59.40 (0.14)**	**59.40 (0.14)**	**85.80 (0.66)**	**85.80 (0.66)**
engel	**12.20 (0.20)**	**12.20 (0.20)**	**50.00 (0.90)**	50.10 (0.90)	**89.40 (0.81)**	**89.40 (0.81)**
mcycle	**12.00 (0.35)**	**12.00 (0.35)**	**48.80 (0.86)**	48.10 (0.86)	86.20 (0.23)	**87.40 (0.23)**
BigMac2003	**14.30 (0.16)**	16.00 (0.16)	**44.20 (0.34)**	43.70 (0.34)	• 77.70 (0.01)	• **79.30 (0.01)**
UN3	**10.30 (0.74)**	**10.30 (0.74)**	**48.60 (0.86)**	47.80 (0.86)	85.80 (0.15)	**86.70 (0.15)**
cpus	• **19.10 (0.00)**	• 20.60 (0.00)	**51.80 (0.58)**	46.90 (0.58)	• 82.10 (0.00)	• **82.50 (0.00)**

Table 9: Ramp loss (quantile property) comparison between the nonparametric quantile regression without (npqr) and with (noncross) non-crossing constraints.

	$\tau = 0.1$		$\tau = 0.5$		$\tau = 0.9$	
data set	npqr	npqrm	npqr	npqrm	npqr	npqrm
cars	**0.65** ± **0.15**	0.66 ± 0.16	**1.59** ± **0.32**	1.61 ± 0.23	0.79 ± 0.16	**0.77** ± **0.16**
onions	2.68 ± 1.21	**2.27** ± **0.71**	4.93 ± 1.58	**4.89** ± **1.47**	1.86 ± 0.73	**1.84** ± **0.37**

Table 10: Pinball loss comparison between the nonparametric quantile regression without (npqr) and with (npqrm) monotonicity constraints.

	$\tau = 0.1$		$\tau = 0.5$		$\tau = 0.9$	
data set	npqr	monotonic	npqr	monotonic	npqr	monotonic
cars	12.00 (0.24)	**11.00** (**0.24**)	**51.00** (**0.88**)	**51.00** (**0.88**)	**89.00** (**0.82**)	**89.00** (**0.82**)
onions	• 18.00 (0.00)	• **17.00** (**0.00**)	**48.00** (**0.44**)	**48.00** (**0.44**)	• **86.00** (**0.01**)	• 80.00 (0.00)

Table 11: Ramp loss (quantile property) comparison between the nonparametric quantile regression without (npqr) and with (npqrm) monotonicity constraints.

References

L. K. Bachrach, T. Hastie, M. C. Wang, B. Narashimhan, and R. Marcus. Bone mineral acquisition in healthy asian, hispanic, black and caucasian youth, a longitudinal study. *Journal of Clinical Endocrinal Metabolism*, 84:4702– 4712, 1999.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Localized rademacher averages. In *Proc. Annual Conf. Computational Learning Theory*, pages 44–58, 2002.

P. J. Bickel, C. A. J. Klaassen, Y. Ritov, and J. A. Wellner. *Efficient and adaptive estimation for semiparametric models*. J. Hopkins Press, Baltimore, ML, 1994.

R. J. Bosch, Y. Ye, and G. G.Woodworth. A convergent algorithm for quantile regression with smoothing splines. *Computational Statistics and Data Analysis*, 19:613–630, 1995.

D. D. Cox. Approximation of method of regularization estimators. *Annals of Statistics*, 16:694–713, 1988.

G. Fung, O. L. Mangasarian, and A. J. Smola. Minimal kernel classifiers. *Journal of Machine Learning Research*, 3:303–321, 2002.

C. Gu and G. Wahba. Semiparametric analysis of variance with tensor product thin plate splines. *Journal of the Royal Statistical Society B*, 55:353–368, 1993.

P. Hall and N. Tajvidi. Distribution and dependence-function estimation for bivariate extreme-value distributions. *Bernoulli*, 2000.

P. Hall, J. S. Marron, and A. Neeman. Geometric representation of high dimension low sample size data. *Journal of the Royal Statistical Society - Series B*, 2005. forthcoming.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

X. He. Quantile curves without crossing. *The American Statistician*, 51(2):186–192, may 1997.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

P. J. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

R. Koenker. *Quantile Regression*. Cambridge University Press, 2005.

R. Koenker and G. Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

R. Koenker, P. Ng, and S. Portnoy. Quantile smoothing splines. *Biometrika*, 81:673–680, 1994.

Q. V. Le, A. J. Smola, and T. Gärtner. Simpler knowledge-based support vector machines. In *Proc. Intl. Conf. Machine Learning*, 2006.

E. Mammen, J. S. Marron, B. A. Turlach, and M. P. Wand. A general projection framework for constrained smoothing. *Statistical Science*, 16(3):232–248, August 2001.

S. Mendelson. A few notes on statistical learning theory. In S. Mendelson and A. J. Smola, editors, *Advanced Lectures on Machine Learning*, number 2600 in LNAI, pages 1–40. Springer, 2003.

D. Ruppert and R. J. Carroll. *Semiparametric Regression*. Wiley, 2003.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

B. Schölkopf, R. C. Williamson, A. J. Smola, and J. Shawe-Taylor. Single-class support vector machines. In J. Buhmann, W. Maass, H. Ritter, and N. Tishby, editors, *Unsupervised Learning*, Dagstuhl-Seminar-Report 235, pages 19–20, 1999.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

A. J. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211–231, 1998.

A. J. Smola, T. Frieß, and B. Schölkopf. Semiparametric support vector and linear programming machines. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 585–591, Cambridge, MA, 1999. MIT Press.

I. Takeuchi and T. Furuhashi. Non-crossing quantile regressions by SVM. In *Proc. International Joint Conference on Neural Networks*, 2004.

M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

V. Vapnik, S. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.

V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, Berlin, 1982.

S. V. N. Vishwanathan, A. J. Smola, and M. N. Murty. SimpleSVM. In Tom Fawcett and Nina Mishra, editors, *Proc. Intl. Conf. Machine Learning*, Washington DC, 2003. AAAI press.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization bounds for regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transaction on Information Theory*, 47(6):2516–2532, 2001.

Journal of Machine Learning Research 7 (2006) 1265–1281 Submitted 7/06; Published 7/06

The Interplay of Optimization and Machine Learning Research

Kristin P. Bennett BENNEK@RPI.EDU
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12018, USA

Emilio Parrado-Hernández EMIPAR@TSC.UC3M.ES
Department of Signal Processing and Communications
University Carlos III de Madrid
Leganés (Madrid), 28911, Spain

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

The fields of machine learning and mathematical programming are increasingly intertwined. Optimization problems lie at the heart of most machine learning approaches. The Special Topic on Machine Learning and Large Scale Optimization examines this interplay. Machine learning researchers have embraced the advances in mathematical programming allowing new types of models to be pursued. The special topic includes models using quadratic, linear, second-order cone, semi-definite, and semi-infinite programs. We observe that the qualities of good optimization algorithms from the machine learning and optimization perspectives can be quite different. Mathematical programming puts a premium on accuracy, speed, and robustness. Since generalization is the bottom line in machine learning and training is normally done off-line, accuracy and small speed improvements are of little concern in machine learning. Machine learning prefers simpler algorithms that work in reasonable computational time for specific classes of problems. Reducing machine learning problems to well-explored mathematical programming classes with robust general purpose optimization codes allows machine learning researchers to rapidly develop new techniques. In turn, machine learning presents new challenges to mathematical programming. The special issue include papers from two primary themes: novel machine learning models and novel optimization approaches for existing models. Many papers blend both themes, making small changes in the underlying core mathematical program that enable the develop of effective new algorithms.

Keywords: machine learning, mathematical programming, convex optimization

1. Introduction

The special topic on "Large Scale Optimization and Machine Learning" focuses on the core optimization problems underlying machine learning algorithms. We seek to examine the interaction of state-of-the-art machine learning and mathematical programming, soliciting papers that either enhanced the scalability and efficiency of existing machine learning models or that promoted new uses of mathematical programming in machine learning. The special topic was an off-shoot of the PASCAL (Pattern Analysis, Statistical Modelling and Computational Learning) Network of Excellence Workshop on "Machine Learning, SVMs and Large Scale Optimization", held in Thurnau, Germany from March 16 to 18, 2005.

Optimization lies at the heart of machine learning. Most machine learning problems reduce to optimization problems. Consider the machine learning analyst in action solving a problem for some set of data. The modeler formulates the problem by selecting an appropriate family of models and massages the data into a format amenable to modeling. Then the model is typically trained by solving a core optimization problem that optimizes the variables or parameters of the model with respect to the selected loss function and possibly some regularization function. In the process of model selection and validation, the core optimization problem may be solved many times. The research area of mathematical programming intersects with machine learning through these core optimization problems. On one hand, mathematical programming theory supplies a definition of what constitutes an optimal solution – the optimality conditions. On the other hand, mathematical programming algorithms equip machine learning researchers with tools for training large families of models.

In general, a mathematical program is a problem of the form

$$\begin{array}{ll} \min_{\mathbf{s}} & f(\mathbf{s}) \\ \text{subject to} & g(\mathbf{s}) \leq \mathbf{0} \\ & h(\mathbf{s}) = \mathbf{0} \\ & \mathbf{s} \in \Omega \end{array} . \tag{1}$$

The variables $\mathbf{s} \in \Omega$ are determined so as to minimize the objective function f possibly subject to inequality $g(\mathbf{s}) \leq \mathbf{0}$ and equality constraints $h(\mathbf{s}) = \mathbf{0}$. Examples of the set Ω include the n-dimensional real numbers, n-dimensional integers, and the set of positive semi-definite matrices. Convexity plays a key role in mathematical programming. Convex programs minimize convex optimization functions subject to convex constraints ensuring that every local minimum is always a global minimum. In general, convex problems are much more tractable algorithmically and theoretically. The complexity of nonconvex problems can grow enormously. General nonconvex programs are NP-hard. However, local solutions of such problems may be quite useful in machine learning problems, e.g. (Dempster et al., 1977; Bennett and Mangasarian, 1993; Bradley et al., 1997; Bradley and Mangasarian, 1998). Global optimization addresses the issue of nonconvex optimization. Integer or discrete optimization considers nonconvex problems with integer constraints.

A taxonomy of mathematical programs exists based on the types of objectives and constraints. There are now many flavors of mathematical programs: linear, quadratic, semi-definite, semi-infinite, integer, nonlinear, goal, geometric, fractional, etc. For example, linear programs have a linear objective and linear constraints. A more complete description of these problems can be obtained from the mathematical programming glossary (www.cudenver.edu/∼hgreenbe/glossary/) and the NEOS optimization guide (www-fp.mcs.anl.gov/otc/Guide/). Each flavor of mathematical program is a different research area in itself with extensive theory and algorithms. Very brief descriptions of the mathematical programs used in this special issue can be found in the Appendix. Good sources for theory and algorithms concerning nonlinear programming are (Nocedal and Wright, 1999), (Bertsekas, 2004), and (Bazaraa et al., 2006). An introduction to convex optimization including semi-definite programming can be found in (Boyd and Vandenberghe, 2004). Semi-infinite programming theory and algorithms are covered in (Goberna and López, 1998). Information about integer programming can be found in (Nemhauser and Wolsey, 1999).

We observe that the relationship between available mathematical programming models and machine learning models has been increasingly coupled. The adaptation of mathematical programming models and algorithms has helped machine learning research advance. Researchers in neural

networks went from backpropagation in (Rummelhart et al., 1986) to exploring the use of various unconstrained nonlinear programming techniques such as discussed in (Bishop, 1996). The fact that backpropagation worked well in turn stimulated mathematical programmers to work on stochastic gradient descent to better understand its properties, as in (Mangasarian and Solodov, 1994). With the advent of kernel methods (Cortes and Vapnik, 1995), mathematical programming terms such as quadratic program, Lagrange multipliers and duality are now very familiar to well-versed machine learning students. Machine learning researchers are designing novel models and methods to exploit more branches of the mathematical programming tree with a special emphasis on constrained convex optimization. The special topic reflects the diversity of mathematical programming models being employed in machine learning. We see how recent advances in mathematical programming have allowed rich new sets of machine learning models to be explored without initial worries about the underlying algorithm. In turn, machine learning has motivated advances in mathematical programming: the optimization problems arising from large scale machine learning and data mining far exceed the size of the problem typically reported in the mathematical programming literature.

This special topic investigates two majors themes in the interplay of machine learning (ML) and mathematical programming (MP).

The first theme contains the extension of well-known optimization methods to new learning models and paradigms. A wide range of convex programming methods is used to create novel models for problems such as uncertain and missing data, and hypothesis selection. Also, methods are developed for introducing constraints into the learning model in order to incorporate domain knowledge into graphical models and to enforce nonnegativity and sparsity in dimensionality reduction methods.

The second theme collects works aimed at solving existing machine learning models more efficiently. As data set size grows, off-the-shelf optimization algorithms become inadequate. Methods that exploit the properties of learning problems can outperform generic mathematical programming algorithms. Many of the included papers deal with well-known convex optimization problems present in ML tools such as the quadratic and linear programs at the core of the ubiquitous support vector machines (SVM) in either primal or dual forms. Tree re-weighted belief propagation is used to solve LP relaxations of large scale real-world belief nets. We see that the key to top performance is creating algorithms that exploit the structure of the problem and pay careful attention to algorithmic and numeric issues.

Many of the papers cross boundaries of both themes. They make small changes in the underlying models that enable the development of powerful new algorithms. Novel methods are developed for multi-kernel, ranking, graph-based clustering, and structured learning. The resulting algorithms decompose the problem into convex subproblems that can be more readily solved.

To summarize, in this special issue we see novel approaches to machine learning models that require solution of continuous optimization problems including: unconstrained, quadratic, linear, second-order cone, semi-definite, and semi-infinite convex programs. We first examine the interplay of machine learning and mathematical programming to understand the desirable properties of optimization methods used for training a machine learning model. We observe that the desirable properties of an optimization algorithm from a machine learning perspective can differ quite markedly from those typically seen in mathematical programming papers. Then we will examine the papers within and across the two themes and discuss how they contribute to the state of the art.

2. Interplay of Optimization and Machine Learning

The interplay of optimization and machine learning is complicated by the fact that machine learning mixes modeling and methods. In that respect, ML is much like operations research (OR). Mathematical programming/optimization is historically a subfield of OR. OR is concerned with modeling a system. Mathematical programming is concerned with analyzing and solving the model. Both OR and ML analysts address real world problems by formulating a model, deriving the core optimization problem, and using mathematical programming to solve it. According to (Radin, 1998) an OR analyst must trade off tractability – "the degree to which the model admits convenient analysis" and validity – "the degree to which inferences drawn from the model hold for real systems". So at a high level the OR and ML analysts face the same validity and tractability dilemmas and it is not surprising that both can exploit the same optimization toolbox.

In ML, generalization is the most essential property used to validate a novel approach. For a practical ML problem, the ML analyst might pick one or more families of learning models and an appropriate training loss/regularization function, and then search for an appropriate model that performs well according to some estimate of the generalization error based on the given training data. This search typically involves some combination of data preprocessing, optimization, and heuristics. Yet every stage of the process can introduce errors that can degrade the quality of the resulting inductive functions. We highlight three sources of such errors. The first source of error is the fact that the underlying true function and error distribution are unknown, thus any choice of data representation, model family and loss functions may not be suitable for the problem and thus introduce inappropriate bias. The second source of error stems from the fact that only a finite amount of (possibly noisy) data is available. Thus even if we pick appropriate loss functions, models, and out-of-sample estimates, the method may still yield inappropriate results. The third source of error stems from the difficulty of the search problem that underlies the given modeling problem. Reducing the problem to a convex optimization by appropriate choices of loss and constraints or relaxations can greatly help the search problem. Note that in many cases the ML models are made convex by an appropriate definition of the system boundaries that treats parameters as fixed. For example, ridge regression for a fixed ridge parameter is a convex unconstrained quadratic program. The generalized cross-validation method (Golub and von Matt, 1997) treats the ridge parameter as within the system boundary, and thus requires the solution of a nonconvex problem.

Consider tractability of a given model expressed as an optimization problem. Both ML and MP seek algorithms that efficiently compute "appropriate" solutions. The issues that make an algorithm more efficient – complexity, memory usage, etc. – are the same for both communities. But there is a large gap between what are considered an appropriate solutions in the two communities. In MP, "appropriate" solutions are the ones that solve the model with a high degree of accuracy as measured by the optimality conditions. As in ML, MP has large suites of benchmark problems. A benchmark study, typically would address both the speed of the algorithms as measured, for example, by performance profiles (Dolan and Moré, 2002). The quality of the solution would be measured by the objective value, a measure of the violation of the constraints, and a measure of the violations of the Karush-Kuhn-Tucker optimality conditions. Note that all of these metrics of solution quality are rarely reported in the ML literature. In MP, great care may be taken to make sure that solutions of equivalent accuracy are compared (see for example (Dolan and Moré, 2002)).

In ML, appropriateness is a much harder question due to the sources of modeling errors described above. A typical benchmarking study reports generalization errors and possibly compu-

tation times. Little or no attention is paid to how well the underlying optimization problem was solved by any of the metrics typically used in mathematical programming. Convergence tolerances are rarely reported and if they are, they typically are quite large (10^{-2} to 10^{-6}) relative to those seen in optimization papers. In machine learning the optimization problems being solved are only rough approximations of the real problem of finding a model that generalizes well. The ML modeler may change the problem formulation and algorithms, as long as generalization is not compromised. The papers in section 6 of this special topic illustrate how minor model reformulations can lead to significant improvements in algorithms. In general, it does not make much sense to require a ML model to converge to a high accuracy solution. When early stopping is used as a form of regularization, then the algorithm may never need to reach the solution. In this special topic (Keerthi et al., 2006) and (Taskar et al., 2006b) develop algorithms relying on early stopping and find that they offer advantages over alternative parametric approaches. Thus the desirable goal of a machine learning algorithm is to find a somewhat accurate solution efficiently. An optimization algorithm that has a poor asymptotic convergence rate may work quite well for ML. Ill-conditioning of the objective is typically viewed as a negative aspect of a model in MP, but ill-conditioning of the loss function and the resulting slow convergence of gradient methods may prevent overfitting. Thus not only is "good" optimization not necessary, but "bad" optimization algorithms can lead to better machine learning models.

In the ML community, Occam's razor appears to apply to algorithms as well; simpler algorithms are considered to be better. MP seeks robust optimization algorithms that find very accurate solutions to a broad class of functions with a premium for decreases in both theoretical complexity and empirical computation time. The emphasis is on solving the same size problems faster. This leads to complex algorithms. The effort to implement a simplex method for linear programming matching a state-of-art commercial solver such as CPLEX would be immense. The ML analyst's computational needs are different. An algorithm that solves the problem with good generalization in a reasonable amount of time is a good algorithm. Incremental speed increases are not so interesting. Simplicity of the algorithms is considered to be a significant plus. Scalability becomes a bigger issue as data set sizes grow. A general purpose solver is usually not the most scalable choice because it was designed to robustly solve a wide range of problems to high accuracy. However, the ML optimization can be tailored to exploit the structure of the optimization model. Robustness and ill-conditioning are not big issues since the algorithm need only be effective for a narrow class of functions and constraints and high accuracy solutions are frequently unnecessary.

To summarize, desirable properties of an optimization algorithm from the ML perspective are

- good generalization,
- scalability to large problems,
- good performance in practice in terms of execution times and memory requirements,
- simple and easy implementation of algorithm,
- exploitation of problem structure
- fast convergence to an approximate solution of model,
- robustness and numerical stability for class of machine learning models attempted,
- theoretically known convergence and complexity.

3. New Machine Learning Models Using Existing Optimization Methods

The special topic papers include novel machine learning models based on existing primarily convex programs such as linear, second order cone, and semi-definite programming. The reader unfamiliar with the basic convex programs can see their definitions in the Appendix. In these papers, the authors develop novel modeling approaches to uncertainty, hypothesis selection, incorporation of domain constraints, and graph clustering, and they use off-the-shelf optimization packages to solve the models.

3.1 Dealing with Uncertainty Using Second Order Cone Programming

The paper "Second Order Cone Programming Approaches for Handling Missing and Uncertain Data" (Shivaswamy et al., 2006) presents an extension to SVM that deals with situations where the observations are not complete or present uncertainty. The SVM Quadratic Program (QP) problem is cast into a more convenient Second Order Cone Program (SOCP) and uncertainty is represented as probabilistic constraints (SVM slack variables turn out to be random variables). They also come up with an interesting geometrical interpretation of their method as every data point being the center of an ellipsoid and the points within this ellipsoid being assigned to the class of the center. The study is extended to multiclass classification and regression.

3.2 Convex Models for Hypothesis Selection

Two papers address hypothesis selection. (Zhang et al., 2006) looks at pruning an ensemble of classifiers constructed from a pool of already trained classifiers. The goal is to make the performance of the smaller group equivalent to that of the whole pool, thus saving storage and computational resources. Traditionally, this selection process has been carried out using heuristics or by using greedy search. In (Bergkvist et al., 2006), the goal is to identify a small subset of hypotheses that exclude the true targets with a given error probability.

The first paper, "Ensemble Pruning Via Semi-Definite Programming" (Zhang et al., 2006), presents an optimization for pruning classification ensembles. The selection of the classifiers is based on a trade-off between their individual accuracies and the disparity of their predictions. This trade-off determines a quadratic integer program, i.e. a QP where the variables have to be integer numbers. The authors in (Zhang et al., 2006) propose a chain of transformations of the quadratic integer program towards a convex semi-definite program (SDP). Experimental results show that this approach beats the state-of-the-art greedy search methods. In addition, the scheme forms the basis of a powerful framework for sharing classifiers in a distributed learning environment, which enables the attack of large scale problems.

In the second paper, "Linear Programs for Hypotheses Selection in Probabilistic Inference Models", Bergkvist et al. (2006) introduce an LP for hypothesis selection in probabilistic inference problems motivated by a protein structure prediction problem. The model optimizes the expected weight of excluded hypotheses for a given error probability bound. The dual variables of the LP represent worst-case distributions of the hypotheses. The authors employ generic off-the-shelf LP optimizers but hypothesize that more efficient algorithms which exploit the problem structure may exist.

4. Models with Side Constraints

The next two papers look at traditional machine learning models with additional constraints.

Niculescu et al. (2006), in "Bayesian Network Learning with Parameter Constraints", use constraints to incorporate domain knowledge into Bayesian networks. The paper examines the cases for parameter sharing and conjugate constrained Dirichlet priors. They employ existing optimization algorithms to solve the resulting models. Addition of constraints improves generalization. Real-world results are presented for hidden process models applied to fMRI brain imaging. They formally prove that introducing constraints reduces variance.

Non-negative matrix factorisation (NMF) is a very attractive feature selection technique because it favors sparsity and data representations based on parts of the problem. However, it also poses a difficult nonconvex problem that is commonly solved via gradient descent. The paper "Learning Sparse Representations by Non-Negative Matrix Factorization and SCOP" (Heiler and Schnörr, 2006) presents an iterative algorithm to perform a sparse non-negative matrix factorization. They exploit the biconvex nature of Euclidean NMF and the reverse-convex structure of the corresponding sparsity constraints to derive an efficient optimization algorithm. This way, the strongly non-convex NMF is solved through the iterative application of a series of convex SOCP problems.

4.1 SDP Methods for Graph Clustering

The paper "Fast SDP Relaxations of Graph Cut Clustering, Transduction, and Other Combinatorial Problems" (De Bie and Cristianini, 2006) proposes an SDP relaxation to the normalized cut problem. The normalized cut problem arises when one wishes to partition a data set where similarity relationships among instances are defined. The mathematical formulation of this problem leads to an intractable combinatorial optimization problem. Spectral relaxation has been used to avoid this intractability. In spectral relaxation, the combinatorial optimization is cast onto a more simple eigendecomposition problem that gives the subsets of data. The new approach in (De Bie and Cristianini, 2006) consists of an SDP relaxation of the combinatorial problem that turns out to be tighter than the spectral one, although at the expenses of a larger computational burden. Moreover, they also present a scheme to develop a cascade of SDP relaxations that allows control of the trade-off between computational cost and accuracy. This study is extended to applications in semi-supervised learning.

5. Refining the Classics: Improvements in Algorithms for Widely Ssed Models

Widely used methods such as SVM and Bayesian networks have well-accepted core optimization problems and algorithms. The demand for the ability to learn with massive amounts of data is increasing. The immediate answer to this demand from the optimization and machine learning communities is to try to come up with more efficient implementations of these solid and reliable optimization methods.

5.1 Optimization Approaches for Dual SVMs

The SVM formulations for classification, regression, ranking, and novelty detection require the solutions of large dense QPs or LPs. These QP and LP problems were initially solved by general-purpose solvers. Now the demand for more scalable and easier to implement algorithms makes novel algorithms for SVMs an active and dynamic research area.

The primary challenges in solving the LP and QP arises from the linear inequality constraints. If the set of constraints that are active, i.e. satisfied at equality, are known then the problems reduce to

the solution of a set of linear equations. The inactive constraints have no effect on the final solution since they are satisfied as strict inequalities. Thus identification of the active constraints, or active set, represents a key step in LP and QP algorithms. One of the most common ways of solving these large QPs and LPs is to use some active set strategy. An active set strategy estimates the active set, solves the problem with respect to the estimated active set, uses the result to update the active set by adding and dropping constraints, and then repeats until an optimal solution is found. In SVMs, active set methods have a clear machine learning interpretation. For example in SVM classification, the active set in the primal corresponds to data points that are on the margin or in error. In the dual SVM formulation, there is a Lagrangian multiplier associated with each point. In the dual, the active set is determined by whether each Lagrangian multiplier is at bound or not.

The paper "An Efficient Implementation of an Active Set Method for SVMs" (Scheinberg, 2006) adapts "traditional" active set methods to the special structure of SVMs. Traditional active set methods were not thought to be tractable for large scale SVMs, but the paper concludes that they are competitive with popular methods such as SVMlight (Joachims, 1999). SVMlight is an example of a restricted active set method in which only a few variables are allowed to vary at each iteration. The restricted active set method in SVMLight decomposes the QP into subproblems, each identified by a group of variables that form an active set. Only the variables in the active set will be updated through the solution of the subproblem. These subproblems are solved until all the optimality conditions are met. These methods have the disadvantage of slow convergence when close to the optimal solution. The full active set in this paper avoids this problem. When full active sets are used, there is a corresponding speedup in the convergence of the optimization method. The paper provides a careful discussion of the details necessary for efficient implementation, active set selection, and warm starts (very valuable for cross-validation). The computational results find that the full active set method performs faster that SVMlight. This difference is most marked for higher accuracy solutions. The full active set method offers a speed scalability tradeoff, it performs faster that SVM-Light but may reach memory limitations sooner since it requires storage of a matrix of the size of the active set.

Reduced active set methods are taken to the extreme result in the popular sequential minimal optimization (SMO) method (Platt, 1999). In SMO, all variables except for a subset of two samples are fixed at each iteration. With the many subsets, the variable selection method becomes a key aspect in the convergence speed of the algorithm. The paper "Maximum-Gain Working Set Selection for SVMs" (Glasmachers and Igel, 2006) describes a new strategy to select the working set based on a greedy maximization of the progress in each single iteration. The algorithm uses precalculated information, which means no increment of the computational burden. The experiments show significant run time reductions over the broadly used SMO-based LIBSVM (Fan et al., 2005), so that full sets can be used, with a corresponding speedup in the convergence of the optimization method.

The paper "Parallel Software for Training Large Scale Support Vector Machines on Multiprocessor Systems" (Zanni et al., 2006) develops a multiprocessor solver for the standard SVM QP. Recent work in MP is used to develop a parallel gradient-projection-based decomposition technique for handling subproblems of moderate size. The subproblems and gradient calculations are done in parallel. Convergence results prove the algorithm converges to an optimal solution of the original QP. In practice, thanks to a large working set size, the algorithm converges in a few iterations. Details on how to fully exploit multiprocessors using strategies such as parallel kernel caching are provided. Results are reported for SVMs trained on millions of data points.

The paper "Incremental Support Vector Learning: Analysis, Implementation and Applications" (Laskov et al., 2006) aims at the software implementation of an efficient incremental learning algorithm for SVMs. The authors examine incremental SVM learning in two scenarios: active learning and limited resource learning. They propose a new storage strategy, simultaneously column-wise and row-wise, combined with a smarter organization of the computations for the minor iteration in terms of gaxpy-type matrix-vector products. This algorithm drops the training time of an incremental SVM by a factor of 5 to 20.

5.2 Optimization Approaches for Primal SVMs

In SVMs and other kernel methods, the computational cost of predicting a novel instance is directly related to the number of nonzero components or support vectors in the prediction function. Thus methods with a reduced number of support vectors are needed. One approach to doing this is to optimize the SVM in the primal form while directing invoking the representer theorem to allow generalization of kernels. In these direct or primal methods, the prediction function is assumed to consist of a linear combination of basis functions formed by the kernel. Prior work has established that sparse and reduced sparse or reduced complexity SVM functions can be achieved by either introducing one-norm regularization or by introducing early stopping strategies.

With respect to greedy construction methods, the paper "Building Support Vector Machines with Reduced Classifier Complexity" (Keerthi et al., 2006) contains a very efficient algorithm to develop a compact support vector machine. The efficiency of the method relies on both a primal method approach to the optimization and a cheap and accurate selection criterion for kernel basis functions. The experimental work presents a wide and systematic comparison with state-of-the-art column generation methods. This comparison points out the excellent capabilities of the algorithm in terms of compression in the number of basis functions, as well as a classification accuracy comparable to that of the full SVM.

Primal or direct kernel SVM models formulated with absolute value type losses and one-norm regularization produce LP core optimization problems. The one-norm enforces sparsity with the degree of sparsity controllable by a tradeoff parameter. Robust general purpose LP optimization tools that exploit advanced numerical analysis are available that can reliably and accurately solve massive problems. But these codes have several drawbacks from machine learning perspective: they are expensive to buy, they are complicated to implement, they do not exploit problem structure, and finally they are designed to find highly accurate solutions while in machine learning this may not be necessary. Thus alternative efficient and easy to implement LP algorithms for LP SVM type models are sorely needed.

The paper "Exact 1-Norm Support Vector Machines Via Unconstrained Convex Differentiable Minimization" (Mangasarian, 2006) introduces a Newton method for exactly solving the 1-norm SVM problem. It shows that the general LP problem can be recast as an unconstrained undifferentiable piecewise quadratic function using a dual exterior penalty function. Unlike prior penalty formulations like (Fung and Mangasarian, 2004), the penalty parameter is finite. The author introduces a generalized Newton method for solving the revised problem. The resulting algorithm outperforms CPLEX, a widely used commercial LP package.

5.3 LP Relaxations for Belief Propagation

The paper "Linear Programming Relaxations and Belief Propagation – An Empirical Study" (Yanover et al., 2006) introduces a very efficient method to find the most probable configuration of a graphical model by relaxing the corresponding integer program to a LP. The bad news is that the resulting LP has a large number of constraints and variables and it cannot be solved on desktop machines using commercial LP solvers. Fortunately the Tree-Reweighted Belief Propagation (TRBP) algorithm can be used to solve the LP. Results show that the special purpose TRBP LP solver outperforms CPLEX and can be used to solve large scale problems that are not tractable with CPLEX. The CPLEX model represents the graph as a matrix while TRBP directly represents the graph.

6. New Algorithms Starting from Reformulated Models

The special issue also illustrates how small reformulations of the model can yield much better algorithms. In the final four papers, we see how existing formulations are reformulated to admit new types of algorithms. In (Sonnenburg et al., 2006) and (Shalev-Shwartz and Singer, 2006), the revised formulations and novel algorithms can more effectively exploit special structure thus reducing the problem to a series of familiar, more easily solved problems.

6.1 Large Scale Multi-Kernel Learning Via Semi-Infinite Programming

The success of a kernel method is highly dependent on the choice of kernel. The multi-kernel learning (MKL) strategy is to consider a suite of kernels and let the algorithm decide on the choice of kernel. The paper "Large Scale Multiple Kernel Learning" (Sonnenburg et al., 2006) proposes a novel semi-infinite linear program (SILP) for the problem of learning with multiple kernels. Semi-infinite linear programs have a finite number of variables, a linear objective, and an infinite number of linear constraints. For SVM classification, the SILP solution is also optimal for the MKL quadratically constrained quadratic program in (Bach et al., 2004). The SILP is solved using a column generation method which alternates between solving a restricted master problem and an LP. The restricted master problem is solved using the corresponding off-the-shelf single kernel learning algorithms for the given loss. The LP is solved by a generic LP solver. The algorithm is very effective at seeking an approximate solution to the SILP. The authors show how the method can be extended to a variety of loss functions. Discussion of valuable details and variations of the algorithm needed for large scale problems are provided. Large scale results are achieved using parallel algorithms. They provide results for problems with up to 10 million data points and 20 kernels.

6.2 Better Ranking by Exploiting Structure

General purpose optimizers frontiers can be pushed forward in particular cases by exploiting problem structure. Often optimization problems can be cast onto simpler ones provided that the objective function follow certain structure. This is the case in the paper "Efficient Learning of Label Ranking by Soft Projection onto Polyhedra" (Shalev-Shwartz and Singer, 2006). The authors develop a fast and frugal algorithm for learning rankings by comparing the predicted graph with the feedback graph resulting in a QP with linear constraints. The algorithm decomposes the problem into a series of soft projections that can be efficiently solved using an iterative algorithm. The algorithm covers a large class of ranking and classification problems including multiclass and basic SVMs. It reduces to SOR in the classification case (Mangasarian and Musicant, 1999).

6.3 Max Margin Methods for Structured Output

Two of the papers tackled maximum margin methods for outputs defined on graphs by reformulating the problem and developing algorithms that could exploit the special structure. The first looks at hierarchical classification and the second looks at methods for more general structured data.

Maximum margin classification methods have focused on binary output problems. These methods have succesfully adapted to multicategory classification by analyzing the problem as a collection of binary problems (Rifkin and Klautau, 2004). However, emerging scenarios where the output is modeled by a vector demand a more careful analysis since their binarization involves (i) an exponential number of subproblems and (ii) the loss of the information encoded in the structured output. In this sense, (Taskar et al., 2006a) proposes an interesting combination of graphical models and maximum margin classifiers where the former allows use of the structured output information and the latter provides a reliable classification technology. Tractability from the optimization point of view is achieved through the grouping of the variables of the optimization problem into marginals defined by the graphical model.

The paper "Kernel-Based Learning of Hierarchical Multilabel Classification Models" (Rousu et al., 2006) provides a more efficient framework for scenarios where the vector output describes a hierarchical relationship. Their formulation requires the solution of a large scale quadratic program. This method's efficiency relies on a decomposition of the core problem into single variable subproblems and the use of a gradient-based approach. Moreover, the optimization is enhanced by a dynamic program that computes the best update directions in the feasible set.

The paper "Structured Prediction, Dual Extragradient and Bregman Projections" (Taskar et al., 2006b) proposes simple scalable maximimum margin algorithms for structured output models including Markov networks and combinatorial models. The problem is to take training data of instances labeled with desired structured outputs and a parametric scoring function and learn the parameters so that the highest scoring outputs match as closely as possible the desired outputs. Prior maximum margin approaches produced QP models (Taskar et al., 2005). By thinking of the problem one level up as a convex concave saddle point model, the authors can capitalize on the recent advances in optimization on extragradient methods (Nesterov, 2003). The extragradient approach produces a simple algorithm consisting of a gradient and projection step. For the class of models considered, the projection requires solution of dynamic program or network flow models for which very efficient algorithms exist. The method is regularized by early stopping. Interestingly the path of the extragradient algorithm corresponds closely to the parametric solution path of the regularized margin methods in their experiments. This demonstrates the interplay of the optimization algorithm and regularization: the path of the optimization algorithm is part of the regularization and there is no need to accurately solve the model.

7. Conclusion

Research in ML and research in MP have become increasingly coupled. ML researchers are making fuller use of the branches of the MP modeling tree. In this issue we see MP researchers using convex optimization methods including linear, nonlinear, saddle point, semi-infinite, second order cone, and semi-definite programming models. The availability of general MP models, along with robust general purpose solvers, provide tools for ML researchers to explore new ML problems. The resulting ML models challenge the capacity of general purpose solvers resulting in the development of novel special purpose algorithms that exploit problem structure. These special purpose solvers

do not necessarily possess the traits associated with good optimization algorithms. Tractability and scalability are valued in both ML and MP communities. Typically, MP demands that algorithms find high accuracy solutions and that they be robustness across wide classes of problems. In contrast, ML algorithm need to find good solutions to narrow classes of problems with special structure. Models may be reformulated to allow better algorithms provided that generalization is improved or at least not compromised. High accuracy is not required because of the inherent inaccuracies in the machine learning models and the fact that inaccurate solutions are deliberately sought as a form of regularization, for example as in early stopping. Also, ML puts more of a premium on algorithms that are easily implemented and understood at the expense of performance/complexity improvements that are typically studied in mathematical programming. In this special topic large scale problems were successfully tackled by methods that exploited both the novel MP models and their special structure and state-of-the-art MP methods. The special issue illustrates the many forms of convex programs that can be used in ML. But we expect the interplay of MP and ML will increase as more branches of the MP tree are incorporated into ML and the demands of large scale ML models exceed the capacity of existing solvers.

Acknowledgments

We would like to acknowledge support for this project from the IST Programme of the European Community under the PASCAL Network of Excellence IST2002-506788. We thank the authors contributing to this special topic for their helpful comments on this introduction. The efforts of the many anonymous reviewers of this special topic are also much appreciated.

Appendix: Standard Convex Programs

This section reviews the basic convex optimization mathematical programming models used in this special issue.

Quadratic Programming

Quadratic programming is used extensively in machine learning and statistics. The use of the least squares loss function in methods such as ridge regression and the 2-norm regularization in most support vector machine models both lead to quadratic programming models. A quadratic program (QP) has a quadratic objective with linear constraints.

Based on (Nocedal and Wright, 1999), we provide a brief review of quadratic programming and the reader can see (Nocedal and Wright, 1999) for more details. The general quadratic program can be stated as

$$\begin{array}{rll} \min_{\mathbf{s}} & \frac{1}{2}\mathbf{s}'\mathbf{Q}\mathbf{s}+\mathbf{c}'\mathbf{s} & \\ \text{subject to} & \mathbf{a}_i\mathbf{s}\leq \mathbf{b}_i & i\in \mathrm{I} \\ & \mathbf{a}_j\mathbf{s}=\mathbf{b}_j & j\in \varepsilon \end{array} \tag{2}$$

where the Hessian $\mathbf{Q}$ is a $n\times n$ symmetric matrix, I and ε are finite sets of indices and $\mathbf{a}_i$, $i\in \mathrm{I}\cup\varepsilon$ are $n\times 1$ vectors. If $\mathbf{Q}$ is positive semi-definite, i.e. $s'Qs\geq 0$ for any s, then the problem is convex. For convex QP, any local solution is also a global solution. A QP can always be solved or shown to be infeasible in a finite number of iterations. The following necessary and sufficient Karush-Kuhn-Tucker (KKT) optimality conditions of QP are formed with the use of Lagrangian multipliers α_i for

the inequality constraints and β_j for the equality constraints:

$$\begin{array}{lll} \text{Primal Feasibility} & \mathbf{a}_i'\mathbf{s} \leq \mathbf{b}_i & i \in \mathrm{I} \\ & \mathbf{a}_j'\mathbf{s} = \mathbf{b}_j & j \in \varepsilon \\ \text{Dual Feasibility} & \mathbf{Q}\mathbf{s} + \sum_{i \in \mathrm{I}} \mathbf{a}_i \alpha_i + \sum_{j \in \varepsilon} \mathbf{a}_j \beta_j = 0 & \\ & \alpha_i \geq 0 & i \in \mathrm{I} \\ \text{Complementarity} & \alpha_i(\mathbf{a}_i'\mathbf{s} - \mathbf{b}_j) = 0 & i \in \mathrm{I} \end{array} \tag{3}$$

Note that if there are no inequality constraints ($\mathrm{I} = \emptyset$), then a KKT point can be found by simply solving a system of linear equations.

Problems with inequality constraints represent more of a challenge. Two families of QP methods prevail: interior point methods and active-set methods. We focus on the latter since active set algorithms are a key component of this special topic. The optimal active set is the set of constraints satisfied as equalities at the optimal solutions. Active set methods work by making educated guesses as to the active set and solving the resulting equality constrained QP. If the guesses are wrong, the method uses gradient and Lagrangian multiplier information to determine constraints to add to or subtract from the active set.

Classical SVMs and the many subsequent variations require the solution of a QP problem.

Linear Programming

Linear programming optimizes a linear function subject to linear constraints. Since linear functions and constraints are convex, an LP is always a convex program. Linear programming can be thought of as a special case of the QP with the Hessian $\mathbf{Q}$ equal to 0. The general linear program can be stated as

$$\begin{array}{rll} \min_{\mathbf{s}} & \mathbf{c}'\mathbf{s} & \\ \text{subject to} & \mathbf{a}_i\mathbf{s} \leq \mathbf{b}_i & i \in \mathrm{I} \\ & \mathbf{a}_j\mathbf{s} = \mathbf{b}_j & j \in \varepsilon \end{array} \tag{4}$$

Interior point methods and simplex methods (active set methods) are both widely used within general purpose LP solvers.

Second-Order Cone Programming

The second-order cone program (SOCP) problems have a linear objective, second-order cone constraints, and possibly additional linear constraints:

$$\begin{array}{rll} \min_{\mathbf{s}} & \mathbf{c}'\mathbf{s} & \\ \text{subject to} & ||\mathbf{R}_i\mathbf{s} + \mathbf{d}_i||_2 \leq \mathbf{a}_i\mathbf{s} + \mathbf{b}_i & i \in \mathrm{C} \\ & \mathbf{a}_j\mathbf{s} = \mathbf{b}_j & j \in \varepsilon \end{array} \tag{5}$$

where $\mathbf{R}_i \in R^{n_i \times n}$ and $\mathbf{d}_i \in R^{n_i}$. Consult (Boyd and Vandenberghe, 2004) Chapter 4 for an introduction to SOCPs and their application to learning type problems. SOCPs are most often solved using interior point algorithms. See (Mittelmann, 2003) for a benchmark of general purpose SOCP algorithms.

Semidefinite Programming

Semidefinite programs (SDPs) are the generalization of linear programs to matrices. In standard

form an SDP minimizes a linear function of a matrix subject to linear equality constraints and a matrix nonnegativity constraint:

$$\begin{array}{rl} \min_{\mathbf{S}} & \langle \mathbf{C}, \mathbf{S} \rangle \\ \text{subject to} & \langle \mathbf{A}_i, \mathbf{S} \rangle = \mathbf{b}_i \quad i \in \mathrm{I} \\ & \mathbf{S} \succeq 0 \end{array} \tag{6}$$

where $\mathbf{S}$, $\mathbf{C}$, and $\mathbf{A}_i$ are in $R^{n \times n}$ and $\mathbf{b}_i \in R$. Here $\mathbf{S} \succeq 0$ means $\mathbf{S}$ must be positive semidefinite and $\langle \mathbf{C}, \mathbf{S} \rangle = trace(\mathbf{CS})$. SDPs are most commonly solved via interior programming methods. A comparison of SDP codes can be found in (Mittelmann, 2003).

Semi-infinite Programming

Semi-infinite linear programs (SILPs) are linear programs with infinitely many constraints. A SILP minimizes a linear objective subject to an infinite number of linear constraints:

$$\begin{array}{rll} \min_{\mathbf{s}} & \frac{1}{2}\mathbf{c}'\mathbf{s} & \\ \text{subject to} & \mathbf{as} \leq 0 & \mathbf{a} \in \mathcal{A} \\ & \mathbf{bs} = 0 & \mathbf{b} \in \mathcal{B} \end{array} \tag{7}$$

where $\mathcal{A}$ and $\mathcal{B}$ are sets (possibly infinite) of n vectors. Reviews of semi-infinite programming can be found in (Hettich and Kortanek, 1993) and (Reemtsen and Ruckmann, 1998), while the book (Goberna and López, 1998) gives extensive coverage of the topic.

References

F. R. Bach, G. R. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear Programming Theory and Algorithms*. Wiley, 2006.

K. P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computational Optimization & Applications*, 2:207–227, 1993.

A. Bergkvist, P. Damaschke, and M. Lüthi. Linear programs for hypotheses selection in probabilistic inference models. *Journal of Machine Learning Research*, 7:1339–1355, 2006.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Cambridge, 2004.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1996.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.

P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference(ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann.

P. S. Bradley, O. L. Mangasarian, and W. N. Street. Clustering via concave minimization. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems -9-*, pages 368–374, Cambridge, MA, 1997. MIT Press.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T. De Bie and N. Cristianini. Fast SDP relaxations approaches of graph cut clustering, transduction and other combinatorial problems. *Journal of Machine Learning Research*, 7:1409–1436, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 5(Dec):1889–1918, 2005.

G. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

T. Glasmachers and C. Igel. Maximum-gain working set selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

M. A. Goberna and M. A. López. *Linear Semi-Infinite Optimization*. John Wiley, New York, 1998.

G. H. Golub and U. von Matt. Generalized cross-validation for large scale problems. *Journal of Computational and Graphical Statistics*, 6(1):1–34, 1997.

M. Heiler and C. Schnörr. Learning sparse representations by non-negative matrix factorization and sequential cone programming. *Journal of Machine Learning Research*, 7:1385–1407, 2006.

R. Hettich and K. O. Kortanek. Semi-infinite programming: theory, methods and application. *SIAM Review*, 3:380–429, 1993.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169 –184. MIT Press, Cambridge, MA, 1999.

S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.

O. L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

O. L. Mangasarian and D. Musicant. Success overrelaxation for support vector machines. *IEEE Transaction on Neural Networks*, 10(5):1032–1037, 1999.

O. L. Mangasarian and M. V. Solodov. Serial and parallel backpropagation convergence via nonmonotone perturbed minimization. *Optimization Methods and Software*, 4(2):103–116, 1994.

H.D. Mittelmann. An independent benchmarking of SDP and SOCP solvers. *Mathematical Programming*, 95(2):407–430, 2003.

G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999.

Y. Nesterov. Dual extrapolation and its application to solving variational inequalities and related problems. Core, Catholic University of Louven, 2003.

R. S. Niculescu, T. M. Mitchell, and R. B. Rao. Bayesian network learning with parameter constraints. *Journal of Machine Learning Research*, 7:1357–1383, 2006.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

R. L. Radin. *Optimization in Operations Research*. Prentice-Hall, New Jersey, 1998.

R. Reemtsen and J. J. Ruckmann. *Semi-infinite programming*. Kluwer Academic, 1998.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5(Jan):101–141, 2004.

J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

D. Rummelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rummelhart and J. McClelland, editors, *Parallel Distributed Processing*, pages 318–362, Cambridge, 1986. MIT Press.

K. Scheinberg. An efficient implementation of an active set method for SVMs. *Journal of Machine Learning Research*, 7:2237–2257, 2006.

S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 7:1567–1599, 2006.

P. K. Shivaswamy, C. Bhattacharyya, and A. J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *International Conference on Machine Learning*, 2005.

B. Taskar, C. Guestrin, V. Chatalbashev, and D. Koller. Max-margin markov networks. *Journal of Machine Learning Research*, pages 1627–1653, 2006a.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006b.

C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation- an empirical study. *Journal of Machine Learning Research*, 7:1887–1907, 2006.

L. Zanni, T. Serafini, and G. Zanghirati. Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7:1467–1492, 2006.

Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.

Journal of Machine Learning Research 7 (2006) 1283–1314 Submitted 7/05; Published 7/06

Second Order Cone Programming Approaches for Handling Missing and Uncertain Data

Pannagadatta K. Shivaswamy PANNAGA@CS.COLUMBIA.EDU
Computer Science
Columbia University
New York, 10027, USA

Chiranjib Bhattacharyya CHIRU@CSA.IISC.ERNET.IN
Department of Computer Science and Automation
Indian Institute of Science
Bangalore, 560 012, India

Alexander J. Smola ALEX.SMOLA@NICTA.COM.AU
Statistical Machine Learning Program
National ICT Australia and ANU
Canberra, ACT 0200, Australia

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We propose a novel second order cone programming formulation for designing robust classifiers which can handle uncertainty in observations. Similar formulations are also derived for designing regression functions which are robust to uncertainties in the regression setting. The proposed formulations are independent of the underlying distribution, requiring only the existence of second order moments. These formulations are then specialized to the case of missing values in observations for both classification and regression problems. Experiments show that the proposed formulations outperform imputation.

1. Introduction

Denote by $(x,y) \in \mathcal{X} \times \mathcal{Y}$ patterns with corresponding labels. The typical machine learning formulation only deals with the case where (x,y) are given *exactly*. Quite often, however, this is not the case — for instance in the case of missing values we may be able (using a secondary estimation procedure) to estimate the values of the missing variables, albeit with a certain degree of uncertainty. In other cases, the observations maybe systematically censored. In yet other cases the data may represent an entire equivalence class of observations (e.g. in optical character recognition all digits, their translates, small rotations, slanted versions, etc. bear the same label). It is therefore only natural to take the potential range of such data into account and design estimators accordingly. What we propose in the present paper goes beyond the traditional imputation strategy in the context of missing variables. Instead, we integrate the fact that some observations are not completely determined into the optimization problem itself, leading to convex programming formulations.

In the context of this paper we will assume that the uncertainty is only in the patterns x, e.g. some of its components maybe missing, and the labels y are known precisely whenever given. We first consider the problem of binary classification where the labels y can take two values, $\mathcal{Y} =$

$\{1,-1\}$. This problem was partially addressed in (Bhattacharyya et al., 2004b), where a second order cone programming (SOCP) formulation was derived to design a robust linear classifier when the uncertainty was described by multivariate normal distributions. Another related approach is the Total Support Vector Classification (TSVC) of Bi and Zhang (2004) who, starting from a very similar premise, end up with a non-convex problem with corresponding iterative procedure.

One of the main contributions of this paper is to generalize the results of Bhattacharyya et al. (2004b) by proposing a SOCP formulation for designing robust binary classifiers for arbitrary distributions having finite mean and covariance. This generalization is acheived by using a multivariate Chebychev inequality (Marshall and Olkin, 1960). We also show that the formulation achieves robustness by requiring that for every uncertain datapoint an ellipsoid should lie in the correct half-space. This geometric view immediately motivates various error measures which can serve as performance metrics. We also extend this approach to the multicategory case. Next we consider the problem of regression with uncertainty in the patterns x. Using Chebyshev inequalities two SOCP fromulations are derived, namely *Close to Mean* formulation and *Small Residual* formulation, which give linear regression functions robust to the uncertainty in x. This is another important contribution of this paper. As in the classification case the formulations can be interpreted geometrically suggesting various error measures. The proposed formulations are then applied to the problem of patterns having missing values both in the case of classification and regression. Experiments conducted on real world data sets show that the proposed formulations outperform imputations. We also propose a way to extend the proposed formulations to arbitrary feature spaces by using kernels for both classification and regression problems.

Outline: The paper is organised as follows: Section 2 introduces the problem of classification with uncertain data. In section 3 we make use of Chebyshev inequalities for multivariate random variable to obtain an SOCP which is one of the main contribution of the paper. We also show that same formulation could be obtained by assuming that the underlying uncertainty can be modeled by an ellipsoid. This geometrical insight is exploited for designing various error measures. A similar formulation is obtained for a normal distribution. Instead of an ellipsoid one can think of more general sets to describe uncertainty. One can tackle such formulations by constraint sampling methods. These constraint sampling methods along with other extensions are discussed in section 4. The other major contribution is discussed in section 5. Again using Chebyshev inequalities two different formulations are derived for regression in section 5 for handling uncertainty in x. As before the formulations motivate various error measures which are useful for comparison. In section 6 we specialize the formulations to the missing value problem both in the case of classification and regression. In section 7 nonlinear prediction functions are discussed. To compare the performance of the formulations numerical experiments were performed on various real world datasets. The results are compared favourably with the imputation based strategy, details are given in section 8. Finally we conclude in section 9.

2. Linear Classification by Hyperplanes

Assume that we have n observations (x_i, y_i) drawn iid (independently and identically distributed) from a distribution over $\mathcal{X} \times \mathcal{Y}$, where x_i is the i^{th} pattern and y_i is the corresponding label. In the following we will briefly review the SVM formulation when the observations are known with certainty and then consider the problem of uncertain observations.

2.1 Classification with Certainty

For simplicity assume that $\mathcal{Y} = \{\pm 1\}$ and $\mathcal{X} = \mathbb{R}^m$ with a finite m. For linearly separable datasets we can find a hyperplane $\langle w, x\rangle + b = 0$ [1] which separates the two classes and the corresponding classification rule is given by

$$f(x) = \operatorname{sgn}(\langle w, x\rangle + b).$$

One can compute the parameters of the hyperplane (w, b) by solving a quadratic optimization problem (see Cortes and Vapnik (1995))

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 \tag{1a}$$

$$\text{subject to } y_i(\langle w, x_i\rangle + b) \geq 1 \qquad \text{for all } 1 \leq i \leq n, \tag{1b}$$

where $\|w\|$ is the euclidean norm.[2] In many cases, such separation is impossible. In this sense the constraints (1b) are hard. One can still construct a hyperplane by relaxing the constraints in (1). This leads to the following soft margin formulation with L_1 regularization (Bennett and Mangasarian, 1993; Cortes and Vapnik, 1995):

$$\underset{w,b,\xi}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2a}$$

$$\text{subject to } y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \qquad \text{for all } 1 \leq i \leq n \tag{2b}$$

$$\xi_i \geq 0 \qquad \text{for all } 1 \leq i \leq n. \tag{2c}$$

The above formulation minimizes an upper bound on the number of errors. Errors occur when $\xi_i \geq 1$. The quantity $C\xi_i$ is the "penalty" for any data point x_i that either lies within the margin on the correct side of the hyperplane ($\xi_i \leq 1$) or on the wrong side of the hyperplane ($\xi_i > 1$).

One can re-formulate (2) as an SOCP by replacing the $\|w\|^2$ term in the objective (2a) by a constraint which upper bounds $\|w\|$ by a constant W. This yields

$$\underset{w,b,\xi}{\text{minimize}} \ \sum_{i=1}^{n}\xi_i \tag{3a}$$

$$\text{subject to } y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \qquad \text{for all } 1 \leq i \leq n \tag{3b}$$

$$\xi_i \geq 0 \qquad \text{for all } 1 \leq i \leq n \tag{3c}$$

$$\|w\| \leq W. \tag{3d}$$

Instead of C the formulation (3) uses a direct bound on $\|w\|$, namely W. One can show that for suitably chosen C and W the formulations (2) and (3) give the same optimal values of (w, b, ξ). Note that (3d) is a second order cone constraint (Lobo et al., 1998).[3] With this reformulation in mind we will, in the rest of the paper, deal with (2) and, with slight abuse of nomenclature, discuss SOCPs where the transformation from (2) to (3) is implicit.

1. $\langle a, b\rangle$ denotes the dot product between $a, b \in \mathcal{X}$. For $\mathcal{X} = \mathbb{R}^m$, $\langle a, b\rangle = a^\top b$. The formulations discussed in the paper holds for arbitrary Hilbert spaces with a suitably defined dot product $\langle ., .\rangle$.
2. The Euclidean norm for element $x \in \mathcal{X}$ is defined as $\|x\| = \sqrt{\langle x, x\rangle}$ where $\mathcal{X}$ is a Hilbert space.
3. Second order cones are given by inequalities in w which take the form $\|\Sigma w + c\| \leq \langle w, x\rangle + b$. In this case $c = 0$ and the cone contains a ray in the direction of $-w$, b determines the offset from the origin, and Σ determines the shape of the cone.

2.2 Classification Under Uncertainty

So far we assumed that the (x_i, y_i) pairs are known with certainty. In many situations this may not be the case. Suppose that instead of the pattern (x_i, y_i) we only have a distribution over x_i, that is x_i is a random variable. In this case we may replace (2b) by a probabilistic constraint

$$\Pr_{x_i}\{y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i\} \geq 1 - \kappa_i \text{ for all } 1 \leq i \leq n. \tag{4}$$

In other words, we require that the random variable x_i lies on the correct side of the hyperplane with probability greater than κ_i. For high values of κ_i, which is a user defined parameter in $(0,1]$, one can obtain a good classifier with a low probability of making errors.

Unless we make some further assumptions or approximations on (4) it will be rather difficult to solve it directly. For this purpose the following sections describe various approaches on how to deal with the optimization. We begin with the assumption that the second moments of x_i exist. In this case we may make use of Chebyshev inequalities (Marshall and Olkin, 1960) to obtain a SOCP.

2.3 Inequalities on Moments

The key tool are the following inequalities, which allow us to bound probabilities of misclassification subject to second order moment constraints on x. Markov's inequality states that if ξ is a random variable, $h : \mathbb{R} \to [0,\infty)$ and a is some positive constant then

$$\Pr\{h(\xi) \geq a\} \leq \frac{\mathbf{E}[h(\xi)]}{a}.$$

Consider the function $h(x) = x^2$. This yields

$$\Pr\{|\xi| \geq a\} \leq \frac{\mathbf{E}\left[\xi^2\right]}{a^2}. \tag{5}$$

Moreover, considering $h(x) = (x - \mathbf{E}[x])^2$ yields the Chebyshev inequality

$$\Pr\{|\xi - \mathbf{E}(\xi)| \geq a\} \leq \frac{\mathbf{Var}[\xi]}{a^2}. \tag{6}$$

Denote by $\bar{x}, \Sigma$ mean and variance of a random variable x. In this case the multivariate Chebyshev inequality (Marshall and Olkin, 1960; Lanckriet et al., 2002; Boyd and Vandenberghe, 2004) is given by

$$\sup_{x\sim(\bar{x},\Sigma)} \Pr\{\langle w, x\rangle \leq t\} = (1+d^2)^{-1} \text{ where } d^2 = \inf_{x|\langle x,w\rangle \leq t} (x-\bar{x})^\top \Sigma^{-1}(x-\bar{x}). \tag{7}$$

This bound always holds for a family of distributions having the same second order moments and in the worst case equality is attained. We will refer to the distribution corresponding to the worst case as the *worst distribution*. These bounds will be used to turn the linear inequalities used in Support Vector Machine classification and regression into inequalities which take the uncertainty of the observed random variables into account.

3. Classification

The main results of our work for the classification problem are presented in this section. Second order cone programming solutions are developed which can handle uncertainty in the observations.

3.1 Main Result

In order to make progress we need to specify properties of (4). Several settings come to mind and we will show that all of them lead to an SOCP.

Robust Formulation Assume that for each x_i we only know its mean $\bar{x_i}$ and variance Σ_i. In this case we want to be able to classify correctly even for the *worst distribution* in this class. Denote by $x \sim (\mu, \Sigma)$ a family of distributions which have a common mean and covariance, given by μ and Σ respectively. In this case (4) becomes

$$\inf_{x_i \sim (\bar{x_i}, \Sigma_i)} \Pr_{x_i} \left(y_i \left(\langle x_i, w \rangle + b \right) \geq 1 - \xi_i \right) \geq 1 - \kappa_i. \tag{8}$$

This means that even for the worst distribution we still classify x_i correctly with high probability $1 - \kappa_i$.

Normal Distribution Equally well, we might assume that x_i is, indeed, distributed according to a normal distribution with mean $\bar{x_i}$ and variance Σ_i. This should allow us to provide tighter bounds, as we have perfect knowledge on how x_i is distributed. In other words, we would like to solve the classification problem, where (4) becomes

$$\Pr_{x_i \sim \mathcal{N}(\bar{x_i}, \Sigma_i)} \left(y_i \left(\langle x_i, w \rangle + b \right) \geq 1 - \xi_i \right) \geq 1 - \kappa_i. \tag{9}$$

Using a Gaussian assumption on the underlying data allows one to use readily available techniques like EM (Dempster et al., 1977; Schneider, 2001) to impute the missing values.

It turns out that both (8) and (9) lead to the same optimization problem.

Theorem 1 *The classification problem with uncertainty, as described in (4) leads to the following second order cone program, when using constraints (8), (9):*

$$\underset{w,b,\xi}{\text{minimize}} \ \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \tag{10a}$$

$$\text{subject to } y_i \left(\langle w, \bar{x_i} \rangle + b \right) \geq 1 - \xi_i + \gamma_i \left\| \Sigma_i^{\frac{1}{2}} w \right\| \qquad \text{for all } 1 \leq i \leq n \tag{10b}$$

$$\xi_i \geq 0 \qquad \text{for all } 1 \leq i \leq n, \tag{10c}$$

where $\Sigma^{\frac{1}{2}}$ is a symmetric square matrix and is the matrix square root of $\Sigma = \Sigma^{\frac{1}{2}} \Sigma^{\frac{1}{2}}$.

More specifically, the following formula for γ_i hold:

- *In the robust case $\bar{x_i}$, Σ_i correspond to the presumed means and variances and*

$$\gamma_i = \sqrt{\kappa_i / (1 - \kappa_i)}. \tag{11}$$

- *In the normal distribution case, again $\bar{x_i}, \Sigma_i$ correspond to mean and variance. Moreover γ_i is given by the functional inverse of the normal CDF, that is*

$$\gamma_i = \phi^{-1}(\kappa_i) \text{ where } \phi(u) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} e^{-\frac{s^2}{2}} ds. \tag{12}$$

Note that for $\kappa_i < 0.5$ the functional inverse of the Gaussian cumulative distribution function becomes negative. This means that in those cases the joint optimization problem is *nonconvex*, as the second order cone constraint enters as a *concave* function. This is the problem that Bi and Zhang (2004) study. They find an iterative procedure which will converge to a local optimum. On the other hand, whenever $\gamma_i \geq 0$ we have a *convex* problem with unique minimum value.

As expected $\phi^{-1}(\kappa_i) < \sqrt{\frac{\kappa_i}{1-\kappa_i}}$. What this means in terms of our formulation is that, by making Gaussian assumption we only scale down the size of the uncertainty ellipsoid with respect to the Chebyshev bound.

Formulation (10) can be solved efficiently using various interior point optimization methods (Boyd and Vandenberghe, 2004; Lobo et al., 1998; Nesterov and Nemirovskii, 1993) with freely available solvers, such as SeDuMi (Sturm, 1999) making them attractive for large scale missing value problems.

3.2 Proof of Theorem 1

Robust Classification We can restate (8) as

$$\sup_{x\sim(\bar{x}_i,\Sigma_i)} \Pr_x \{y_i(\langle w,x\rangle + b) \leq 1-\xi_i\} \leq \kappa_i.$$

See that it is exactly equivalent to (8) and using Eq. (7) we can write

$$\sup_{x\sim(\bar{x}_i,\Sigma_i)} \Pr_x \{y_i(\langle w,x\rangle + b) \geq 1-\xi_i\} = (1+d^2)^{-1} \leq \kappa_i, \tag{13a}$$

$$\text{where, } d^2 = \inf_{x|y_i(\langle x,w\rangle+b)\leq 1-\xi_i} (x-\bar{x}_i)^\top \Sigma_i^{-1}(x-\bar{x}_i). \tag{13b}$$

Now we solve (13b) explicitly. In case $\bar{x}_i$ satisfies $y_i(\langle w,x_i\rangle + b) \geq 1-\xi_i$ then clearly the infimum in (13b) is zero. If not, d^2 is just the distance of the mean $\bar{x}_i$ from the hyperplane $y_i(\langle w,x_i\rangle + b) = 1-\xi_i$, that is

$$d^2 = \frac{y_i(\langle w,\bar{x}_i\rangle + b - 1 + \xi_i)}{\sqrt{w^\top \Sigma_i w}}. \tag{14}$$

The expression for d^2 in (14) when plugged into the requirement $\frac{1}{1+d^2} \leq \kappa_i$ gives (10b) where γ_i is given as in (11) thus proving the first part.

Normal Distribution Since projections of a normal distributions are themselves normal we may rewrite (9) as a scalar probabilistic constraint. We have

$$\Pr\left\{\frac{z_i-\bar{z}_i}{\sigma_{z_i}} \geq \frac{y_i b + \xi_i - 1 - \bar{z}_i}{\sigma_{z_i}}\right\} \quad \leq \quad \kappa_i, \tag{15}$$

where $z_i := -y_i\langle w,x_i\rangle$ is a normal random variable with mean $\bar{z}_i$ and variance $\sigma_{z_i}^2 := w^\top \Sigma_i w$. Consequently $(z_i-\bar{z}_i)/\sigma_{z_i}$ is a random variable with zero mean and unit variance and we can compute the lhs of (15) by evaluating the cumulative distribution function $\phi(x)$ for normal distributions. This makes (15) equivalent to the condition

$$\phi\left(\sigma_{z_i}^{-1}(y_i b + \xi_i - 1 - \bar{z}_i)\right) \geq \kappa_i,$$

which can be solved for the argument of ϕ.

3.3 Geometric Interpretation and Error Measures

The constraint (10b) can also be derived from a geometric viewpoint. Assume that x takes values in an ellipsoid with center $\bar{x}$, metric Σ and radius[4] γ, that is

$$x \in \mathcal{E}(\bar{x}, \Sigma, \gamma) := \left\{ x | \, (x-\bar{x})^\top \Sigma^{-1} (x-\bar{x}) \leq \gamma^2 \right\}. \tag{16}$$

The robustness criteria can be enforced by requiring that that we classify x correctly for all $x \in \mathcal{E}(\bar{x}, \Sigma, \gamma)$, that is

$$y(\langle x, w \rangle + b) \geq 1 - \xi \text{ for all } x \in \mathcal{E}(\bar{x}, \Sigma, \gamma). \tag{17}$$

In the subsequent section we will study other constraints than ellipsoid sets for x.

Lemma 2 *The optimization problem*

$$\underset{x}{\text{minimize}} \ \langle w, x \rangle \ \text{subject to } x \in \mathcal{E}(\bar{x}, \Sigma, \gamma)$$

has its minimum at $\bar{x} - \gamma \left(w^\top \Sigma w \right)^{-\frac{1}{2}} \Sigma w$ *with minimum value* $\langle \bar{x}, w \rangle - \gamma \left(w^\top \Sigma w \right)^{\frac{1}{2}}$. *Moreover, the maximum of* $(\langle w, x \rangle - \langle w, \bar{x} \rangle)$ *subject to* $x \in \mathcal{E}(\bar{x}, \Sigma, \gamma)$ *is given by* $\gamma \left\| \Sigma^{\frac{1}{2}} w \right\|$.

Proof We begin with the second optimization problem. Substituting $v := \Sigma^{-\frac{1}{2}} (x - \bar{x})$ one can see that the problem is equivalent to maximizing $\langle w, \Sigma^{\frac{1}{2}} v \rangle$ subject to $\|v\| \leq \gamma$. The latter is maximized for $v = \gamma \Sigma^{\frac{1}{2}} w / \left\| \Sigma^{\frac{1}{2}} w \right\|$ with maximum value $\gamma \left\| \Sigma^{\frac{1}{2}} w \right\|$. This proves the second claim.

The first claim follows from the observation that maximum and minimum of the second objective function match (up to a sign) and from the fact that the first objective function can be obtained form the second by a constant offset $\langle w, \bar{x} \rangle$. ■

This means that for fixed w the minimum of the lhs of (17) is given by

$$y_i (\langle \bar{x_i}, w \rangle + b) - \gamma_i \sqrt{w^\top \Sigma_i w}. \tag{18}$$

The parameter γ is a function of κ, and is given by (11) in the general case. For the normal case it is given by (12). We will now use this ellipsoidal view to derive quantities which can serve as performance measures on a test set.

Worst Case Error: given an uncertainty ellipsoid, we can have the following scenarios:

1. The centroid is classified correctly and the hyperplane does not cut the ellipsoid: The error is zero as all the points within the ellipsoid are classified correctly.

2. The centroid is misclassified and the hyperplane does not cut the ellipsoid: Here the error is 1 as all the points within the ellipsoid are misclassified.

3. The hyperplane cuts the ellipsoid. Here the worst case error is one as we can always find points within the uncertainty ellipsoid that get misclassified.

4. Note that we could as well dispose of γ by transforming $\Sigma \leftarrow \gamma^{-2} \Sigma$. The latter, however, leads to somewhat inconvenient notation.

Figure 1 illustrates these cases. It shows a scenario in which there is uncertainty in two of the features. Figure corresponds to those two dimensions. It shows three ellipsoids corresponding to the possible scenarios.

To decide whether the ellipsoid, $\mathcal{E}(\mu,\Sigma,\gamma)$, intersects the hyperplane, $w^\top x+b=0$, one needs to compute

$$z=\frac{w^\top\mu+b}{\sqrt{w^\top\Sigma w}}.$$

If $|z|\leq\gamma$ then the hyperplane intersects the ellipsoid, see (Bhattacharyya et al., 2004a). For an uncertain observation, i.e. given an ellipsoid, with the label y, the worst case error is given by

$$e_{wc}(\mathcal{E})=\begin{cases}1 & \text{if } yz<\gamma\\ 0 & \text{otherwise.}\end{cases}$$

Expected Error The previous measure is a pessimistic one. A more optimistic measure could be the expected error. We find out the volume of the ellipsoid on the wrong side of the hyperplane and use the ratio of this volume to the entire volume of the ellipsoid as the expected error measure. When the hyperplane doesn't cut the ellipsoid, expected error is either zero or one depending on whether the ellipsoid lies entirely on the correct side or entirely on the wrong side of the hyperplane. In some sense, this measure gives the expected error for each sample when there is uncertainty. In figure 1 we essentially take the fraction of the area of the shaded portion of the ellipsoid as the expected error measure. In all our experiments, this was done by generating large number of uniformly distributed points in the ellipsoid and then taking the fraction of the number of points on the correct side of the hyperplane to the total number of points generated.

4. Extensions

We now proceed to extending the optimization problem to a larger class of constraints. The following three modifications come to mind: (a) extension to multiclass classification, (b) extension of the setting to different types of set constraints, and (c) the use of constraint sampling to deal with nontrivial constraint sets

4.1 Multiclass Classification

An obvious and necessary extension of above optimization problems is to deal with multiclass classification. Given $y\in\mathcal{Y}$ one solves the an optimization problem maximizing the multiclass margin (Collins, 2002; Rätsch et al., 2002; Taskar et al., 2003):

$$\underset{w,\xi}{\text{minimize}}\ \sum_{i=1}^{n}\xi_i \tag{19a}$$

$$\text{subject to } \langle w_{y_i},x_i\rangle-\max_{y\neq y_i}\langle w_y,x_i\rangle\geq 1-\xi_i \text{ and } \xi_i\geq 0 \qquad \text{for all } 1\leq i\leq n \tag{19b}$$

$$\sum_{i=1}^{|\mathcal{Y}|}\|w_{y_i}\|^2\leq W^2. \tag{19c}$$

Here w_i are the weight vectors corresponding to each class. Taking square roots of (19c) yields a proper SOCP constraint on $w\in\mathbb{R}^{d\times|\mathcal{Y}|}$. Note that instead of (19b) we could also state $|\mathcal{Y}|-1$ linear

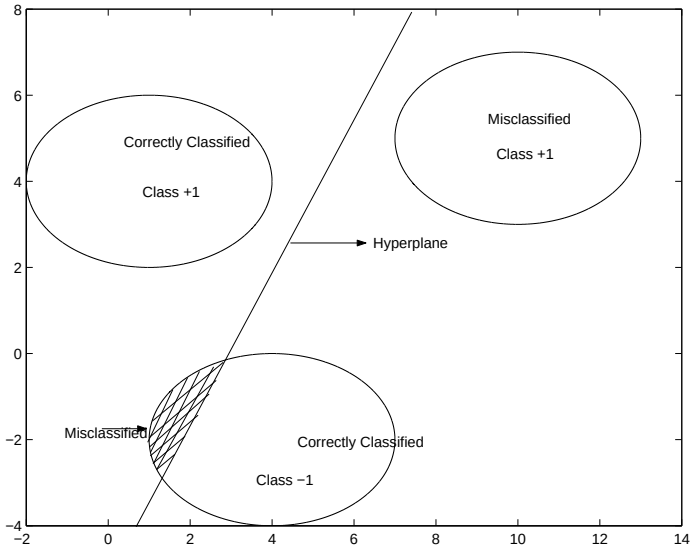


Figure 1: Three scenarios occurring when classifying a point: One of the unshaded ellipsoids lies entirely on the "correct" side of the hyperplane, the other lies entirely on the "wrong" side of the hyperplane. The third, partially shaded ellipsoid has parts on either sides. In the worst case we count the error for this pattern as one whereas in the expected case we count the error as the fraction of the volume (in this case area) on the "wrong" side as the error

inequalities on w_i according to each (y_i, y) combination. The latter allows us apply a reasoning analogous to that of Theorem 1 (we skip the proof as it is identical to that of Section 3.2 with small modifications for a union bound argument). This yields:

$$\underset{w,b,\xi}{\text{minimize}} \ \frac{1}{2}\sum_{i=1}^{|\mathcal{Y}|}\|w_i\|^2 + C\sum_{i=1}^{n}\xi_i \tag{20a}$$

$$\text{subject to } (\langle w_{y_i} - w_y, \overline{x_i}\rangle) \geq 1 - \xi_i + \gamma_i \left\| \Sigma_i^{\frac{1}{2}}(w_{y_i} - w_y) \right\| \qquad \text{for } 1 \leq i \leq n, y \neq y_i \tag{20b}$$

$$\xi_i \geq 0 \qquad \text{for } 1 \leq i \leq n. \tag{20c}$$

The key difference between (10) and (20) is that we have a set of $|\mathcal{Y}| - 1$ second order cone constraints per observation.

4.2 Set Constraints

The formulations presented so far can be broadly understood in the context of robust convex optimization (see Ben-Tal and Nemirovski (1998, 2001)). In the following we discuss a few related formulations which were proposed in the context of pattern classification. This subsection lists types of the constraint set and the kind of optimization problems used for solving SVM for the underlying constraint sets.

Note that we may rewrite the constraints on the classification as follows:

$$y_i(\langle x, w\rangle + b) \geq 1 - \xi_i \text{ for all } x \in S_i. \tag{21}$$

Here the sets S_i are given by $S_i = \mathcal{E}(\bar{x_i}, \Sigma_i, \gamma_i)$. This puts our optimization setting into the same category as the knowledge-based SVM (Fung et al., 2002) and SDP for invariances (Graepel and Herbrich, 2004), as all three deal with the above type of constraint (21), but the set S_i is different. More to the point, in (Graepel and Herbrich, 2004) $S_i = S(b_i, \beta)$ is a polynomial in β which describes the set of invariance transforms of x_i (such as distortion or translation). (Fung et al., 2002) define S_i to be a polyhedral "knowledge" set, specified by the intersection of linear constraints.

By the linearity of (21) it follows that if (21) holds for S_i then it also holds for coS_i, the convex hull of S_i. Such considerations suggest yet another optimization setting: instead of specifying a polyhedral set S_i by constraints we can also specify it by its vertices. Depending on S_i such a formulation may be computationally more efficient.

In particular if S_i is the convex hull of a set of generators x_{ij} as in

$$S_i = \text{co}\{x_{ij} \text{ for } 1 \leq j \leq m_i\}.$$

We can replace (21) by

$$y_i(\langle w, x_{ij}\rangle + b) \geq 1 - \xi_i \text{ for all } 1 \leq j \leq m_i.$$

In other words, enforcing constraints for the convex hull is equivalent to enforcing them for the *vertices* of the set. Note that the index ranges over j rather than i. Such a setting is useful e.g. in the case of range constraints, where variables are just given by interval boundaries. Table 1 summarizes the five cases. Clearly all the above constraints can be mixed and matched. More central is the notion of stating the problems via (21) as a starting point.

Table 1: Constraint sets and corresponding optimization problems.

Name	Set S_i	Optimization Problem
Plain SVM	$\{x_i\}$	Quadratic Program
Knowledge Based SVM	Polyhedral set	Quadratic Program
Invariances	trajectory of polynomial	Semidefinite Program
Normal Distribution	$\mathcal{E}(x_i, \Sigma_i, \gamma_i)$	Second Order Cone Program
Convex Hull	$\text{co}\{x_{ij} \; \forall \; 1 \leq j \leq m_i\}$	Quadratic Program

4.3 Constraint Sampling Approaches

In the cases of Table 1 reasonably efficient convex optimization problems can be found which allow one to solve the domain constrained optimization problem. That said, the optimization is often quite costly. For instance, the invariance based SDP constraints of Graepel and Herbrich (2004) are computationally tractable only if the number of observations is in the order of tens to hundreds, a far cry from requirements of massive datasets with thousands to millions of observations.

Even worse, the set S may not be finite and it may not be convex either. This means that the optimization problem, while convex, will not be able to incorporate S efficiently. We could, of course, circumscribe an ellipsoid for S by using a large γ to obtain a sufficient condition. This

approach, however, would typically lead to overly pessimistic classifiers. An alternative is constraint sampling, as proposed by (de Farias and Roy, 2004; Calafiore and Campi, 2004).

Let $f : \mathbb{R}^d \to \mathbb{R}$ and $c : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^l$ be convex functions, with $\Omega \subseteq \mathbb{R}^d$ being a closed convex set and $S \subseteq \mathbb{R}^l$. Consider the following optimization problem which is an instance of well known semi-infinite program

$$\underset{\theta \in \Omega}{\text{minimize}}\, f(\theta) \text{ subject to } c(\theta, x) \leq 0 \text{ for all } x \in S. \tag{22}$$

Depending on S the problem may have infinite number of constraints, and is in general intractable for arbitrary f and c. The constraint sampling approach for such problems proceeds by first imposing a probability distribution over S and then obtaining N independent observations, $x_1, \ldots, x_N$ from the set S by sampling. Finally one solves the finite convex optimization problem

$$\underset{\theta \in \Omega}{\text{minimize}}\, f(\theta) \text{ subject to } c(\theta, x_i) \leq 0 \text{ for all } 1 \leq i \leq N. \tag{23}$$

The idea is that by satisfying N constraints there is a high probability that an arbitrary constraint $c(x, \theta)$ is also satisfied. Let θ_N be the solution of (23). Note that since x_i are random variables θ_N, is also a random variable. The choice of N is given by a theorem due to Calafiore and Campi (2004).

Theorem 3 *Let $\varepsilon, \beta \in (0,1)$ and let $\theta \in \mathbb{R}^d$ be the decision vector then*

$$\Pr\{V(\theta_N) \leq \varepsilon\} \geq 1 - \beta \text{ where } V(\theta_N) = \Pr\{c(\theta_N, x) > 0 | x \in S\}$$

holds if

$$N \geq 2\left[d\varepsilon^{-1}\log\varepsilon^{-1} + \varepsilon^{-1}\log\beta^{-1} + d\right],$$

provided the set $\{x \in S | c(\theta_N, x) > 0\}$ is measurable.

Such a choice of N guarantees that the optimal solution θ_N of the sampled problem (23) is ε level feasible solution of the robust optimization problem (22) with high probability. Specializing this approach for the problem at hand would require drawing N independent observations from the set S_i, for each uncertain constraint, and replacing the SOCP constraint by N linear constraints of the form

$$y(w^\top x_i^j + b) \geq 1 \text{ for all } j \in \{1, \ldots N\}.$$

The choice of N is given by Theorem 3. Clearly the resulting problem is convex and has finite number of constraints. More importantly this makes the robust problem same as the standard SVM optimization problem but with more number of constraints.

In summary the advantage with the constraint sampling approach is one can still solve a robust problem by using a standard SVM solver instead of an SOCP. Another advantage is the approach easily carries over to arbitrary feature spaces. The downside of Theorem 3 is that N depends linearly on the *dimensionality* of w. This means that for nonparametric setting tighter bounds are required.[5]

5. Such bounds are subject to further work and will be reported separately.

5. Regression

Beyond classification the robust optimization approach can also be extended to regression. In this case one aims at finding a function $f : \mathcal{X} \to \mathcal{Y}$ such that some measure of deviation $c(e)$ between the observations and predictions, where $e(f(x),y) := f(x) - y$, is small. For instance we penalize

$$c(e) = \tfrac{1}{2}e^2 \qquad \text{LMS Regression } (l_2) \tag{24a}$$

$$c(e) = |e| \qquad \text{Median Regression } (l_1) \tag{24b}$$

$$c(e) = \max(0, |e| - \varepsilon) \qquad \varepsilon\text{-insensitive Regression} \tag{24c}$$

$$c(e) = \begin{cases} |e| - \frac{\sigma}{2} & \text{if } |e| \leq \sigma \\ \frac{1}{2\sigma}e^2 & \text{otherwise} \end{cases} \qquad \text{Huber's robust regression} \tag{24d}$$

The ℓ_1 and ℓ_2 losses are classical. The ε-insensitive loss was proposed by Vapnik et al. (1997), the robust loss is due to Huber (1982). Typically one does not minimize the empirical average over these losses directly but rather one minimizes the regularized risk which is composed of the empirical mean plus a penalty term on f controlling the capacity. See e.g. (Schölkopf and Smola, 2002) for further details.

Relatively little thought has been given so far to the problem when x may not be well determined. Bishop (1995) studies the case where x is noisy and he proves that this has a regularizing effect on the estimate. Our aim is complementary: we wish to find robust estimators which do not change significantly when x is only known approximately subject to some uncertainty. This occurs, e.g. when some coordinates of x are missing.

The basic tool for our approach are the Chebyshev and Gauss-Markov inequalities respectively to bound the first and second moment of $e(f(x),y)$. These inequalities are used to derive two SOCP formulations for designing robust estimators useful for regression with missing variables. Note that no distribution assumptions are made on the underlying uncertainty, except that the first and the second moments are available. Our strategy is similar to (Chandrasekaran et al., 1998; El Ghaoui and Lebret, 1997) where the worst case residual is limited in presence of bounded uncertainties.

5.1 Penalized Linear Regression and Support Vector Regression

For simplicity the main body of our derivation covers the linear setting. Extension to kernels is discussed in a later section Section 7. In penalized linear regression settings one assumes that there is a function

$$f(x) = \langle w, x \rangle + b, \tag{25}$$

which is used to minimize a regularized risk

$$\underset{w,b}{\text{minimize}} \sum_{i=1}^{n} c(e_i) \text{ subject to } \|w\| \leq W \text{ and } e_i = f(x_i) - y_i. \tag{26}$$

Here $W > 0$. As long as $c(e_i)$ is a convex function, the optimization problem (26) is a convex programming problem. More specifically, for the three loss functions of (24a) we obtain a quadratic program. For $c(e) = \frac{1}{2}e^2$ we obtain Gaussian Process regression estimators (Williams, 1998), in the second case we obtain nonparametric median estimates (Le et al., 2005), and finally $c(e) = \max(0, |e| - \varepsilon)$ yields ε-insensitive SV regression (Vapnik et al., 1997).

Eq. (26) is somewhat nonstandard insofar as the penalty on $\|w\|$ is imposed via the constraints rather than via a penalty in the objective directly. We do so in order to obtain second order cone programs for the robust formulation more easily without the need to dualize immediately. In the following part of the paper we will now seek means of bounding or estimating e_i subject to constraints on x_i.

5.2 Robust Formulations for Regression

We now discuss how to handle uncertainty in x_i. Assume that x_i is a random variable whose first two moments are known. Using the inequalities of Section 2.3 we derive two formulations which render estimates robust to the stochastic variations in x_i.

Denote by $\bar{x} := \mathbf{E}[x]$ the expected value of x. One option of ensuring robustness of the estimate is to require that the prediction errors are insensitive to the distribution over x. That is, we want that

$$\Pr_x\{|e(f(x),y) - e(f(\bar{x}),y)| \geq \theta\} \leq \eta, \tag{27}$$

for some confidence threshold θ and some probability η. We will refer to (27) as a "close to mean" (CTM) requirement. An alternative is to require that the residual $\xi(f(x),y)$ be small. We make use of a probabilistic version of the constraint $|e(f(x),y)| \leq \xi + \varepsilon$, that is equivalent to

$$\Pr_x\{|e(f(x),y)| \geq \xi + \varepsilon\} \leq \eta. \tag{28}$$

This is more geared towards good performance in terms of the loss function, as we require the estimator to be robust only in terms of deviations which lead to *larger* estimation error rather than requiring smoothness overall. We will refer to (28) as a "small residual" (SR) requirement. The following theorem shows how both quantities can be bounded by means of the Chebyshev inequality (6) and modified markov inequality (5).

Theorem 4 (Robust Residual Bounds) *Denote by $x \in \mathbb{R}^n$ a random variable with mean $\bar{x}$ and covariance matrix Σ. Then for $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ a sufficient condition for (27) is*

$$\left\|\Sigma^{\frac{1}{2}} w\right\| \leq \theta\sqrt{\eta}, \tag{29}$$

where $\Sigma^{\frac{1}{2}}$ is the matrix square root of Σ. Moreover, a sufficient condition for (28) is

$$\sqrt{w^\top \Sigma w + (\langle w,\bar{x}\rangle + b - y)^2} \leq (\xi + \varepsilon)\sqrt{\eta}. \tag{30}$$

Proof To prove the first claim note that for f as defined in (25), $\mathbf{E}(e(f(x),y)) = e(f(\bar{x}),y)$ which means that $e(f(x),y) - e(f(\bar{x}),y)$ is a zero-mean random variable whose variance is given by $w^\top \Sigma w$. This can be used with Chebyshev's inequaltiy (6) to bound

$$\Pr_x\{|e(f(x),y) - e(f(\bar{x}),y)| \geq \theta\} \leq \frac{w^\top \Sigma w}{\theta^2}. \tag{31}$$

Hence $w^\top \Sigma w \leq \theta^2 \eta$ is a sufficient condition for (27) to hold. Taking square roots yields (29). To prove the second part we need to compute the second order moment of $e(f(x),y)$. The latter is computed easily by the bias-variance decomposition as

$$\begin{aligned}\mathbf{E}\left[e(f(x),y)^2\right] &= \mathbf{E}\left[\left(e(f(x),y) - e(f(\bar{x}),y)\right)^2\right] + e(f(\bar{x}),y)^2 \\ &= w^\top \Sigma w + (\langle w,\bar{x}\rangle + b - y)^2.\end{aligned} \tag{32}$$

Using (5), we obtain a sufficient condition for (28)

$$w^\top \Sigma w + (\langle w, \bar{x}\rangle + b - y)^2 \leq (\xi + \varepsilon)^2 \eta. \tag{33}$$

As before, taking the square root yields (30). ■

5.3 Optimization Problems for Regression

The bounds obtained so far allow us to recast (26) into a robust optimization framework. The key is that we replace the equality constraint $e_i = f(x_i) - y_i$ by one of the two probabilistic constraints derived in the previous section. In the case of (27) this amounts to solving

$$\begin{aligned}
\underset{w,b,\theta}{\text{minimize}} \; & \sum_{i=1}^n c(e_i) + D\sum_{i=1}^n \theta_i && && \text{(34a)}\\
\text{subject to } & \|w\| \leq W \text{ and } \theta_i \geq 0 && \text{for all } 1 \leq i \leq n && \text{(34b)}\\
& \langle \bar{x_i}, w\rangle + b - y_i = e_i && \text{for all } 1 \leq i \leq n && \text{(34c)}\\
& \|\Sigma_i^{\frac{1}{2}} w\| \leq \theta_i \sqrt{\eta_i} && \text{for all } 1 \leq i \leq n, && \text{(34d)}
\end{aligned}$$

where (34d) arises from $\Pr_{x_i}\{|e(f(x_i), y_i) - e(f(\bar{x_i}), y_i)| \geq \theta_i\} \leq \eta_i$. Here D is a constant determining the degree of uncertainty that we are going to accept large deviations. Note that (34) is a *convex* optimization problem for all convex loss functions $c(e)$. This means that it constitutes a general robust version of the regularized linear regression problem and that all adjustments including the ν-trick can be used in this context. For the special case of ε-insensitive regression (34) specializes to an SOCP. Using the standard decomposition of the positive and negative branch of $f(x_i) - y_i$ into ξ_i and ξ_i^* Vapnik et al. (1997) we obtain

$$\begin{aligned}
\underset{w,b,\xi,\xi^*,\theta}{\text{minimize}} \; & \sum_{i=1}^n (\xi_i + \xi_i^*) + D\sum_{i=1}^n \theta_i && && \text{(35a)}\\
\text{subject to } & \|w\| \leq W \text{ and } \theta_i, \xi_i, \xi_i^* \geq 0 && \text{for all } 1 \leq i \leq n && \text{(35b)}\\
& \langle \bar{x_i}, w\rangle + b - y_i \leq \varepsilon + \xi_i \text{ and } y_i - \langle \bar{x_i}, w\rangle - b \leq \varepsilon + \xi_i^* && \text{for all } 1 \leq i \leq n && \text{(35c)}\\
& \|\Sigma_i^{\frac{1}{2}} w\| \leq \theta_i \sqrt{\eta_i} && \text{for all } 1 \leq i \leq n. && \text{(35d)}
\end{aligned}$$

In the same manner, we can use the bound (30) for (28) to obtain an optimization problem which minimizes the regression error directly. Note that (28) already allows for a margin ε in the regression error. Hence the optimization problem becomes

$$\begin{aligned}
\underset{w,b,\xi}{\text{minimize}} \; & \sum_{i=1}^n \xi_i && && \text{(36a)}\\
\text{subject to } & \|w\| \leq W \text{ and } \xi_i \geq 0 && \text{for all } 1 \leq i \leq n && \text{(36b)}\\
& \sqrt{w^\top \Sigma_i w + (\langle w, \bar{x_i}\rangle + b - y_i)^2} \leq (\xi_i + \varepsilon)\sqrt{\eta_i} && \text{for all } 1 \leq i \leq n. && \text{(36c)}
\end{aligned}$$

Note that (36) is an SOCP. In our experiments we will refer to (35) as the "close-to-mean" (CTM) formulation and to (36) as the "small-residual" (SR) formulation.

5.4 Geometrical Interpretation and Error Measures

The CTM formulation can be motivated by a similar geometrical interpretation to the one in the classification case, using an ellipsoid with center $\bar{x}$, shape and size determined by Σ and γ.

Theorem 5 *Assume that x_i is uniformly distributed in $\mathcal{E}(\bar{x}_i, \Sigma_i, \frac{1}{\sqrt{\eta_i}})$ and let f be defined by (25). In this case (35d) is a sufficient condition for the following requirement:*

$$|e(f(x_i),y) - e(f(\bar{x}_i),y)| \leq \theta_i \;\; \forall x_i \in \mathcal{E}_i \text{ where } \mathcal{E}_i := \mathcal{E}\left(\bar{x}_i, \Sigma_i, \eta_i^{-\frac{1}{2}}\right). \tag{37}$$

Proof Since $f(x) = \langle w,x\rangle + b$, left inequality in (37) amounts to $|\langle w,x_i\rangle - \langle w,\bar{x}_i\rangle| \leq \theta_i$. The inequality holds for all $x_i \in \mathcal{E}_i$ if $\max_{x_i\in\mathcal{E}_i} |\langle w,x_i\rangle - \langle w,\bar{x}_i\rangle| \leq \theta_i$. Application of Lemma 2 yields the claim. ■

A similar geometrical interpretation can be shown for SR. Motivated from this we define the following error measures.

Robustness Error: from the geometrical interpretation of CTM it is clear that $\gamma\|\Sigma^{\frac{1}{2}}w\|$ is the maximum possible difference between x and any other point in $\mathcal{E}(\bar{x},\Sigma,\gamma)$, since a small value of this quantity means smaller difference between $e(f(x_i),y_i))$ and $e(f(\bar{x_i}),y_i))$, we call $e_{\text{robust}}(\Sigma,\gamma)$ the *robustness error* measure for CTM

$$e_{\text{robust}}(\Sigma,\gamma) = \gamma\|\Sigma^{\frac{1}{2}}w\|. \tag{38}$$

Expected Residual: from (32) and (33) we can infer that SR attempts to bound the expectation of the square of the residual. We denote by $e_{\text{exp}}(\Sigma,\bar{x})$ an error measure for SR where,

$$e_{\text{exp}}(\bar{x},\Sigma) = \sqrt{w^\top\Sigma w + (e(f(\bar{x}),y))^2}. \tag{39}$$

Worst Case Error: since both CTM and SR are attempting to bound $w^\top\Sigma w$ and $e(f(\bar{x}_i),y_i)$ by minimizing a combination of the two and since the maximum of $|e(f(x),y)|$ over $\mathcal{E}(\bar{x},\Sigma,\gamma)$ is $|e(f(\bar{x}),y)| + \gamma\|\Sigma^{\frac{1}{2}}w\|$ (see Lemma 2) we would expect this worst case residual $w(\bar{x},\Sigma,\gamma)$ to be low for both CTM and SR. This measure is given by

$$e_{\text{worst}}(\bar{x},\Sigma,\gamma) = |e(f(\bar{x}),y)| + \gamma\|\Sigma^{\frac{1}{2}}w\|. \tag{40}$$

6. Robust Formulation For Missing Values

In this section we discuss how to apply the robust formulations to the problem of estimation with missing values. While we use a linear regression model to fill in the missing values, the linear assumption is not really necessary: as long as we have information on the first and second moments of the distribution we can use the robust programming formulation for estimation.

6.1 Classification

We begin by computing the sample mean and covariance for each class from the available observations, using a linear model and Expectation Maximization (EM) (Dempster et al., 1977) to take care of missing variables wherever appropriate:

Let (x,y) have parts x_m and x_a, corresponding to missing and available components respectively. With mean μ and covariance Σ for the class y and with decomposition

$$\mu = \begin{bmatrix} \mu_a \\ \mu_m \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{am} \\ \Sigma_{am}^\top & \Sigma_{mm} \end{bmatrix}, \tag{41}$$

we can now find the imputed means and covariances. They are given by

$$\mathbf{E}[x_m] = \mu_m + \Sigma_{ma}\Sigma_{aa}^{-1}(x_a - \mu_a) \tag{42}$$

$$\text{and } \mathbf{E}\left[x_m x_m^\top\right] - \mathbf{E}[x_m]\,\mathbf{E}[x_m]^\top = \Sigma_{mm} - \Sigma_{ma}\Sigma_{aa}^{-1}\Sigma_{ma}^\top. \tag{43}$$

In standard EM fashion one begins with initial estimates for mean and covariance, uses the latter to impute the missing values for the entire class of data and iterates by re-estimating mean and covariance until convergence.

Optimization Problem Without loss of generality, suppose that the patterns 1 to c are complete and that patterns $c+1$ to n have missing components. Using the above model we have the following robust formulation:

$$\underset{w,b,\xi}{\text{minimize}} \sum_{i=1}^{n} \xi_i \tag{44a}$$

$$\text{subject to } y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \qquad \text{for all } 1 \leq i \leq c \tag{44b}$$

$$y_i\left(\langle w, \bar{x}_i\rangle + b\right) \geq 1 - \xi_i + \gamma_i \left\| \Sigma_i^{\frac{1}{2}} w \right\| \qquad \text{for all } c+1 \leq i \leq n \tag{44c}$$

$$\|w\| \leq W \;\text{ and }\; \xi_i \geq 0 \qquad \text{for all } 1 \leq i \leq n, \tag{44d}$$

where $\bar{x}_i$ denotes the pattern with the missing values filled in and

$$\Sigma_i = \begin{bmatrix} 0 & 0 \\ 0 & \Sigma_{mm} - \Sigma_{ma}\Sigma_{aa}^{-1}\Sigma_{am} \end{bmatrix}$$

according to the appropriate class labels. By appropriately choosing γ_i's, we can control the degree of robustness to uncertainty that arises out of imputation. The quantities γ_i's are defined only for the patterns with missing components.

Prediction After determining w and b by solving (44) we predict the label y of the pattern x by the following procedure.

1. If x has no missing values use it for step 4.
2. Fill in the missing values x_m in x using the parameters (mean and the covariance) of each class, call the resulting patterns x_+ and x_- corresponding to classes $+1$ and -1 respectively.
3. Find the distances d_+, d_- of the imputed patterns from the hyperplane, that is

   $$d_\pm := \left(w^\top x_\pm + b\right)\left(w^\top \Sigma_\pm w\right)^{-\frac{1}{2}}.$$

   Here $\Sigma_\pm$ are the covariance matrices of x_+ and x_-. These values tell which class gives a better fit for the imputed pattern. We choose that imputed sample which has higher distance from the hyperplane as the better fit: if $|d_+| > |d_-|$ use x_+, otherwise use x_- for step 4.
4. Calculate $y = \operatorname{sgn}\left(w^\top x + b\right)$.

6.2 Regression

As before we assume that the first c training samples are complete and the remaining training samples have missing values. After using the same linear model an imputation strategy as above we now propose to use the CTM and SR formulations to exploit the covariance information to design robust prediction functions for the missing values.

Once the missing values are filled in, it is straightforward to use our formulation. The CTM formulation for the missing values case takes the following form

$$\begin{aligned}
\underset{w,b,\theta,\xi,\xi^*}{\text{minimize}} \; & \sum_{i=1}^{n}(\xi_i+\xi_i^*)+D\sum_{i=c+1}^{n}\theta_i && && \text{(45a)}\\
\text{subject to } & \langle w,x_i\rangle+b-y_i\le\varepsilon+\xi_i \,,\, y_i-\langle w,x_i\rangle-b\le\varepsilon+\xi_i^* && \text{for all } 1\le i\le c && \text{(45b)}\\
& \langle w,\bar{x}_i\rangle+b-y_i\le\varepsilon+\xi_i \,,\, y_i-\langle w,\bar{x}_i\rangle-b\le\varepsilon+\xi_i^* && \text{for all } c+1\le i\le n && \text{(45c)}\\
& \left\|\Sigma_i^{\frac{1}{2}}w\right\|\le\theta_i\sqrt{\eta_i} && \text{for all } c+1\le i\le n && \text{(45d)}\\
& \theta_i\ge 0 \;\text{ for all } c+1\le i\le n \;\text{ and }\; \xi_i,\xi_i^*\ge 0 && \text{for all } 1\le i\le n && \text{(45e)}\\
& \|w\|\le W.
\end{aligned}$$

Only partially available data have the constraints (45d). As before, quantities θ_i's are defined only for patterns with missing components. A similar SR formulation could be easily obtained for the case of missing values:

$$\begin{aligned}
\underset{w,b,\xi,\xi^*}{\text{minimize}} \; & \sum_{i=1}^{c}(\xi_i+\xi_i^*)+\sum_{i=c+1}^{n}\xi_i\\
\text{subject to } & \langle w,x_i\rangle+b-y_i\le\varepsilon+\xi_i \;,\; y_i-\langle w,x_i\rangle-b\le\varepsilon+\xi_i^* && \text{for all } 1\le i\le c\\
& \sqrt{w^\top\Sigma_i w+(\langle w,\bar{x}_i\rangle+b-y_i)^2}\le(\varepsilon+\xi_i)\sqrt{\eta_i} && \text{for all } c+1\le i\le n\\
& \xi_i^*\ge 0 \text{ for all } 1\le i\le c \;\text{ and }\; \xi_i\ge 0 && \text{for all } 1\le i\le n\\
& \|w\|\le W.
\end{aligned}$$

7. Kernelized Robust Formulations

In this section we propose robust formulations for designing nonlinear classifiers by using kernel function. Note that a kernel function is a function $K:\Omega\times\Omega\to\mathbb{R}$, where K obeys the Mercer conditions (Mercer, 1909). We also extend these ideas to nonlinear regression functions.

7.1 Kernelized Formulations for Classification

The dual of the formulation (44), is given below (for a proof, please see Appendix A).

$$\underset{\lambda,\delta,\beta,u}{\text{maximize}} \sum_{i=1}^{n} \lambda_i - W\delta, \tag{47a}$$

$$\text{subject to } \sum_{i=1}^{n} \lambda_i y_i = 0, \tag{47b}$$

$$\|\sum_{i=1}^{c} \lambda_i y_i x_i + \sum_{i=c+1}^{n} \lambda_i y_i (\bar{x}_i + \gamma_i \Sigma_i^{\frac{1}{2}T} u_i)\| \leq \delta, \tag{47c}$$

$$\lambda_i + \beta_i = 1 \qquad \text{for all } 1 \leq i \leq n \tag{47d}$$

$$\|u_i\| \leq 1 \qquad \text{for all } c+1 \leq i \leq n \tag{47e}$$

$$\lambda_i, \beta_i, \delta \geq 0 \qquad \text{for all } 1 \leq i \leq n. \tag{47f}$$

The KKT conditions can be stated as (see Appendix A)

$$\sum_{i=1}^{c} \lambda_i y_i x_i + \sum_{i=c+1}^{n} \lambda_i y_i (\bar{x}_i + \gamma_i \Sigma_i^{\frac{1}{2}} u_i) = \delta u_{n+1} \tag{48a}$$

$$\sum_{i=1}^{n} \lambda_i y_i = 0, \delta \geq 0 \tag{48b}$$

$$\lambda_i + \beta_i = 1,\ \beta_i \geq 0,\ \lambda_i \geq 0,\ \beta_i \lambda_i = 0 \qquad \text{for all } 1 \leq i \leq n \tag{48c}$$

$$\lambda_i (y_i (\langle w, x_i \rangle + b) - 1 + \xi_i) = 0 \qquad \text{for all } 1 \leq i \leq c \tag{48d}$$

$$\lambda_j (y_j (\langle w, \bar{x}_j \rangle + b) - 1 + \xi_j - \gamma_j (\Sigma_j^{\frac{1}{2}} u_j)) = 0 \qquad \text{for all } c+1 \leq j \leq n \tag{48e}$$

$$\delta(\langle w, u_{n+1} \rangle - W) = 0. \tag{48f}$$

The KKT conditions of the problem give some very interesting insights:

1. When $\gamma_i = 0$ $c+1 \leq i \leq n$ the method reduces to standard SVM expressed as an SOCP as it is evident from formulation (47).

2. When $\gamma_i \neq 0$ the problem is still similar to SVM but instead of a fixed pattern the solution chooses the vector $\bar{x}_i + \gamma_i \Sigma_i^{\frac{1}{2}} u_i$ from the uncertainty ellipsoid. Which vector is chosen depends on the value of u_i. Figure (2) has a simple scenario to show the effect of robustness on the optimal hyperplane.

3. The unit vector u_i maximizes $u_i^\top \Sigma_i^{\frac{1}{2}} w$ and hence u_i has the same direction as $\Sigma_i^{\frac{1}{2}} w$.

4. The unit vector u_{n+1} has the same direction as w. From (48a), for arbitrary data, one obtains $\delta > 0$, which implies $\langle w, u_{n+1} \rangle = W$ due to (48f). Substituting for u_{n+1} in (48a) gives the following expression for w,

$$w = \frac{W}{\delta} \left(\sum_{i=1}^{c} \lambda_i y_i x_i + \sum_{i=c+1}^{n} \lambda_i y_i \left(\bar{x}_i + \gamma_i \Sigma_i^{\frac{1}{2}} u_i \right) \right). \tag{49}$$

This expression for w is very similar to the expression obtained in the standard SVM. The vector w has been expressed as a combination of complete patterns and vectors from the uncertainty ellipsoid of the incomplete patterns.

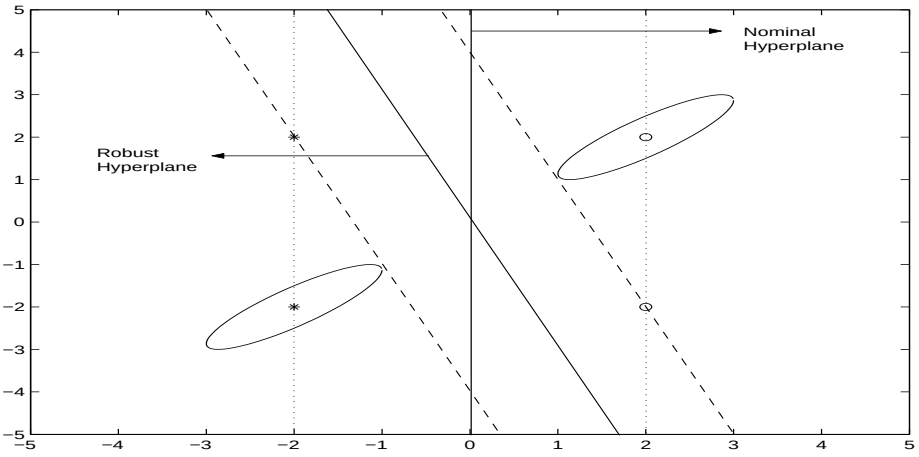


Figure 2: Circles and stars represent patterns belonging to the two classes. The ellipsoid around the pattern denotes the uncertainty ellipsoid. Its shape is controlled by the covariance matrix and the size by γ. The vertical solid line represents the optimal hyperplane obtained by nominal SVM while the thick dotted line represents the optimal hyperplane obtained by the robust classifier

Kernelized Formulation It is not simple to solve the dual (47) as a kernelized formulation. The difficulty arises from the fact that the constraint containing the dot products of the patterns (47c) involves terms such as $\left(\bar{x}_i+\gamma_i\Sigma_i^{\frac{1}{2}}u_i\right)^T\left(\bar{x}_j+\gamma_j\Sigma_j^{\frac{1}{2}}u_j\right)$ for some i and j. As u's are unknown, it is not possible to calculate the value of the kernel function directly. Hence we suggest a simple method to solve the problem from the primal itself.

When the shape of the uncertainty ellipsoid for a pattern with missing values is determined by the covariance matrix of the imputed values, any point in the ellipsoid is in the span of the patterns used in estimating the covariance matrix. This is because the eigenvectors of the covariance matrix span the entire ellipsoid. The eigenvectors of a covariance matrix are in the span of the patterns from which the covariance matrix is estimated. Since eigenvectors are in the span of the patterns and they span the entire ellipsoid, any vector in the ellipsoid is in the span of the patterns from which the covariance matrix is estimated.

The above fact and the equation to construct w from the dual variables (49) imply w is in the span of the imputed data (all the patterns: complete and the incomplete patterns with missing values imputed). Hence, $w=\sum_{i=1}^{c}\alpha_i x_i+\sum_{i=c+1}^{n}\alpha_i\bar{x}_i$.

Now, consider the constraint

$$y_i\left(\langle w,x_i\rangle+b\right)\geq 1-\xi_i.$$

It can be rewritten as,

$$y_i\left(\left\langle\left(\sum_{l=1}^{c}\alpha_l x_l+\sum_{l=c+1}^{n}\alpha_l\bar{x}_l\right),x_i\right\rangle+b\right)\geq 1-\xi_i.$$

We replace the dot product in the above equation by a kernel function to get

$$y_i\left(\left\langle \alpha, \tilde{K}(x_i)\right\rangle + b\right) \geq 1-\xi_i,$$

where $\tilde{K}(x_i)^T = [K(x_1,x_i),\ldots,K(x_c,x_i),K(\bar{x}_{c+1},x_i),\ldots,K(\bar{x}_n,x_i)]$ and $\alpha^\top = [\alpha_1,\ldots,\alpha_n]$. The observation x_i is either a complete pattern or a pattern with missing values filled in. Now, we consider the uncertainty in $\tilde{K}(x_i)$ to obtain the non-linear version of our formulation that can be solved easily. When we consider the uncertainty in $\tilde{K}(x_i)$ the probabilistic constraint takes the form

$$Pr\left(y_i\left(\left\langle \alpha, \tilde{K}(x_i)\right\rangle + b\right) \geq 1-\xi_i\right) \geq \kappa_i. \tag{50}$$

As in the original problem we now treat $\tilde{K}(x_i)$ as a random variable. The equation (50) has the same structure as the probabilistic constraint of Section 3. Following the same steps as in Section 3, it can be shown that the above probabilistic constraint is equivalent to

$$y_i\left(\left\langle \alpha, \tilde{K}(x_i)\right\rangle + b\right) \geq 1-\xi_i + \sqrt{\frac{\kappa_i}{1-\kappa_i}}\sqrt{\alpha^T\Sigma_i^k\alpha},$$

where Σ_i^k and $\tilde{K}(\bar{x}_i)$ are the covariance and the mean of $\tilde{K}(x_i)$ (in $\tilde{K}$-space). In view of this, the following is the non-linear version of the formulation:

$$\underset{\alpha,b,\xi}{\text{minimize}} \sum_{i=1}^{n}\xi_i \tag{51a}$$

$$\text{subject to } y_i\left(\left\langle \alpha, \tilde{K}(x_i)\right\rangle + b\right) \geq 1-\xi_i \qquad \text{for all } 1\leq i\leq c \tag{51b}$$

$$y_i\left(\left\langle \alpha, \tilde{K}(\bar{x}_j)\right\rangle + b\right) \geq 1-\xi_j+\gamma_j\left\|\Sigma_j^{k\frac{1}{2}}\alpha\right\| \qquad \text{for all } c+1\leq j\leq n \tag{51c}$$

$$\|\alpha\| \leq W \;\; \xi_i \geq 0 \qquad \text{for all } 1\leq i\leq n. \tag{51d}$$

The constraint (51d) follows from the fact that we are now doing linear classification in $\tilde{K}$-space. The constraint is similar to the constraint $\|w\| \leq W$ which we had in the linear versions.

Estimation of Parameters A point to be noted here is that Σ_j^k defines the uncertainty in $\tilde{K}(x_j)$. In the original lower dimensional space we had a closed form formula to estimate the covariance for patterns with missing values. However, now we face a situation where we need to estimate the covariance in $\tilde{K}$-space. A simple way of doing this is to assume spherical uncertainty in $\tilde{K}$-space. Another way of doing this is by a nearest neighbour based estimation. To estimate the covariance of $\tilde{K}(x_i)$, we first find out k nearest neighbours of x_i and then we estimate the covariance from $\tilde{K}(x_{i_1}),\ldots,\tilde{K}(x_{i_k})$ where $x_{i_1},\ldots,x_{i_k}$ are the nearest neighbours of x_i.

It is straight forward to extend this more general result (51) to the missing value problem following the same steps as in (6).

Classification Once α's are found, given a test pattern t its class is predicted in the following way: If the pattern is incomplete, it is first imputed using the way it was done during training. However, this can be done in two ways, one corresponding to each class as the class is unknown for the pattern. In that case the distance of each imputed pattern from the hyperplane is computed from

$$h_1 = \frac{\alpha^T\tilde{K}(t)+b}{\sqrt{\alpha^T\Sigma_1\alpha}} \text{ and } h_2 = \frac{\alpha^T\tilde{K}(t)+b}{\sqrt{\alpha^T\Sigma_2\alpha}},$$

where Σ_1 and Σ_2 are the covariances obtained by the same strategy as during training. Higher of the above two is selected as it gives a better fit for the pattern. The prediction for the pattern is the prediction of its centroid (i.e. the prediction for the centroid which gives a better fit). Let $h = \max(|h_1|, |h_2|)$, if $h = |h_1|$ then $y = \text{sgn}(h_1)$ else $y = \text{sgn}(h_2)$ where y is the prediction for the pattern t. In case the pattern is complete, there is no ambiguity we can give $\text{sgn}(\alpha^T \tilde{K}(t) + b)$ as the prediction.

7.2 Kernelized Robust Formulations for Regressions

As discussed for the case of classification we derive nonlinear regressions functions by using the $\tilde{K}$. We fit a hyperplane (α, b) in the $\tilde{K}$ where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]$. Whenever x is a random variable we consider $\tilde{K}(x)$ as a random variable with mean $\tilde{K}(\bar{x})$ and with either unit covariance or a covariance estimated from nearest neighbours in the $\tilde{K}$-space. Instead of finding (w, b) we resort to finding (α, b) where α plays the role of w but in the $\tilde{K}$-space. Essentially, we just have to replace w by α and x_i by $\tilde{K}x_i$ and the covariance by the estimate covariance in the $\tilde{K}$-space. Given these facts, we get the following kernelized version of the Close To Mean formulation:

$$\begin{aligned}
\underset{\alpha,b,\theta,\xi,\xi^*}{\text{minimize}} & \sum_{i=1}^{n}(\xi_i + \xi_i^*) + D\sum_{i=1}^{n}\theta_i \\
\text{subject to } & \left\langle \alpha, \tilde{K}(\bar{x}_i)\right\rangle + b - y_i \le \varepsilon + \xi_i && \text{for all } 1 \le i \le n \\
& y_i - \left\langle \alpha, \tilde{K}(\bar{x}_i)\right\rangle - b \le \varepsilon + \xi_i^* && \text{for all } 1 \le i \le n \\
& \sqrt{\alpha^\top \Sigma_i^k \alpha} \le \theta_i \sqrt{\eta_i} && \text{for all } 1 \le i \le n \\
& \|\alpha\| \le W \text{ and } \theta_i, \xi_i, \xi_i^* \ge 0 && \text{for all } 1 \le i \le n.
\end{aligned}$$

Similarly, the kernelized version of formulation SR is given by,

$$\begin{aligned}
\underset{\alpha,b,\xi}{\text{minimize}} & \sum_{i=1}^{n}\xi_i \\
\text{subject to } & \sqrt{\alpha^\top \Sigma_i^k \alpha + (\left\langle \alpha, \tilde{K}(\bar{x}_i)\right\rangle + b - y_i)^2} \le (\varepsilon + \xi_i)\sqrt{\eta_i} && \text{for all } 1 \le i \le n \\
& \|\alpha\| \le W \text{ and } \xi_i \ge 0 && \text{for all } 1 \le i \le n.
\end{aligned}$$

In the above formulations, Σ_i^k is the estimate covariance in the $\tilde{K}$-space. If the patterns 1 through c are complete and the patterns $c+1$ through n have missing values, then assuming $\eta_i = 1$ and $\Sigma_i^k = 0$ for i from 1 through c, would make the above formulations directly applicable to the case.

8. Experiments

In this section we empirically test the derived formulations for both classification and regression problems which have missing values in the observations. In all the cases interior point method was used to solve SOCP using the commercially avilable Mosek solver.

8.1 Classification

We consider the classification case first. Consider a binary classification problem with training data having missing values. The missing values are filled in by imputation and subsequently a

SVM classifier was trained on the complete data to obtain the *nominal classifier*. We compared the proposed formulations with the nominal classifiers by performing numerical experiments on real life data bench mark datasets. We also use a non-linear separable data set to show that the kernelized version works when the linear version breaks down. In our formulations we will assume that $\gamma_j = \gamma$.

For evaluating the results of robust classifier we used the worst case error and the expected error along with the actual error. A test pattern with no missing values can be directly classified. In case it has missing values, we first impute the missing values and then classify the pattern. We refer to the error on a set of patterns using this approach the actual error.

We first consider the problem of classifying OCR data where missing values can occur more frequently. Specifically we consider the classification problem between the two digits '3' and '8'. We have used the UCI (Blake and Merz, 1998) OCR data set, A data set is generated by deleting 75% of the pixels from 50% of the training patterns. Missing values were then imputed using linear regression. We trained a SVM on this imputed data, to obtain the nominal classifier. This was compared with the robust classifier trained with different values of γ, corresponding to different degrees of confidence as stated in (11).

The error rates of the classifiers were obtained on the test data set by randomly deleting 75% of the pixels from each pattern. We then repeated 10 such iterations and obtained the average error rates. Figure 3 shows some of the digits that were misclassified by the nominal classifier but were correctly classified by the robust classifier. The effectiveness of our formulation is evident from these images. With only partial pixels available, our formulation did better than the nominal classifier. Figure 4 show the different error rates obtained on this OCR data set. In all the three measures, the robust classifier outperformed the nominal classifier.

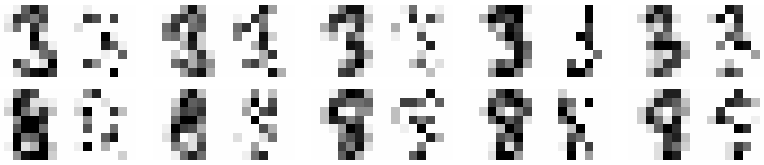


Figure 3: In all images the left image shows a complete digit, the right image shows the digit after randomly deleting 75% of the pixels. The first five are '3' while the next five are '8'.

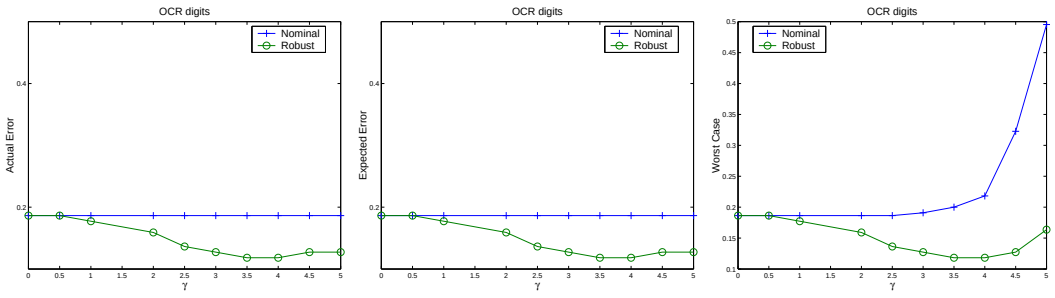


Figure 4: Error rates against γ with linear classifier on the OCR data.

Here we report the error rates using three measures we defined for three other UCI data sets (Blake and Merz (1998)), Heart, Ionosphere and Sonar. Linear version of our formulation was used. Experiments were done with low noise (50% patterns with missing values) and high noise (90% patterns with missing values). The data sets were divided in the ratio 9:1, the larger set was used for training the nominal and robust classifiers while the smaller set was used as test data set. 50% of the feature values (chosen at random) were deleted from 50% of the training patterns (in the low noise case) and 90% of the training patterns (in the high noise case). Linear regression based model was used to fill in the missing values. Nominal classifier and robust classifiers with different values of γ were trained using each such data set. Error rates were obtained for the test data set after deleting 50% of the feature values from each test pattern. The error rates reported here are over ten such randomized iterations.

The error rates as a function of γ are plotted in Figures 5,6 and 7. In case of actual error, the plots also show a horizontal line labeled 'clean' which is the error rate on the actual data set without any missing values. In this case, we did not delete any feature values from the data set. Nominal classifiers were trained and testing was also done on complete test samples. Our aim was to see how close our robust classifier could get near the error rates obtained using the complete data set.

It can be seen that the robust classifier, with suitable amount of robustness comes very close to the error rates on the clean data set. Amongst the three error measures the worst case error, the last column of Figure 7, brings out the advantage of the robust classifier over the nominal classifier. Clearly with increasing γ the robust formulation gives dividends over the nominal classifier.

We also did experiments to compare the kernelized version of the formulation over the linear formulation. For this purpose, we generated a dataset as follows. The positive class was obtained by generating uniformly distributed points in a hypershpere in $\mathbb{R}^5$ of unit radius centered at the origin. The negative class was obtained by generating uniformly distributed points in a annular band of thickness one, with the inner radius two, centered around the origin. In summary

$$y = \begin{cases} 1 & \|x\| \leq 1 \\ -1 & 2 \leq \|x\| \leq 3, \end{cases}$$

where $x \in \mathbb{R}^5$. An illustration of how such a dataset looks in two dimensions is given in the left of Figure 8. Hundred patterns were generated for each class. The data set was divided in the ratio 9:1. The larger part was used for training, the smaller part for testing. Three randomly chosen values were deleted from the training data set. The missing values were filled in using linear regression based strategy. We trained a classifier for different values of γ. Actual Error was found out for both the kernelized version and the linear version of the formulation. The results reported here are over ten such randomized runs. Gaussian kernel $(K(x,y) = \exp(-q\|x-y\|^2))$ was used in the case of kernelized formulation. The parameter q was chosen by cross validation. Spherical uncertainty was assumed in $\tilde{K}$-space for samples with missing values in case of kernelized robust formulations.

Figure 8 shows actual error rates with linear nominal, linear robust, kernelized nominal and kernelized robust. It can be seen that the linear classifier has broken down, while the kernelized classifier has managed a smaller error rate. It can also be observed that the robust kernelized classifier has the least error rate.

8.2 Regression

Given a regression problem with training data having missing values in the observations we obtained the *nominal regression* function by training a Support Vector Regression(SVR) formulation over the

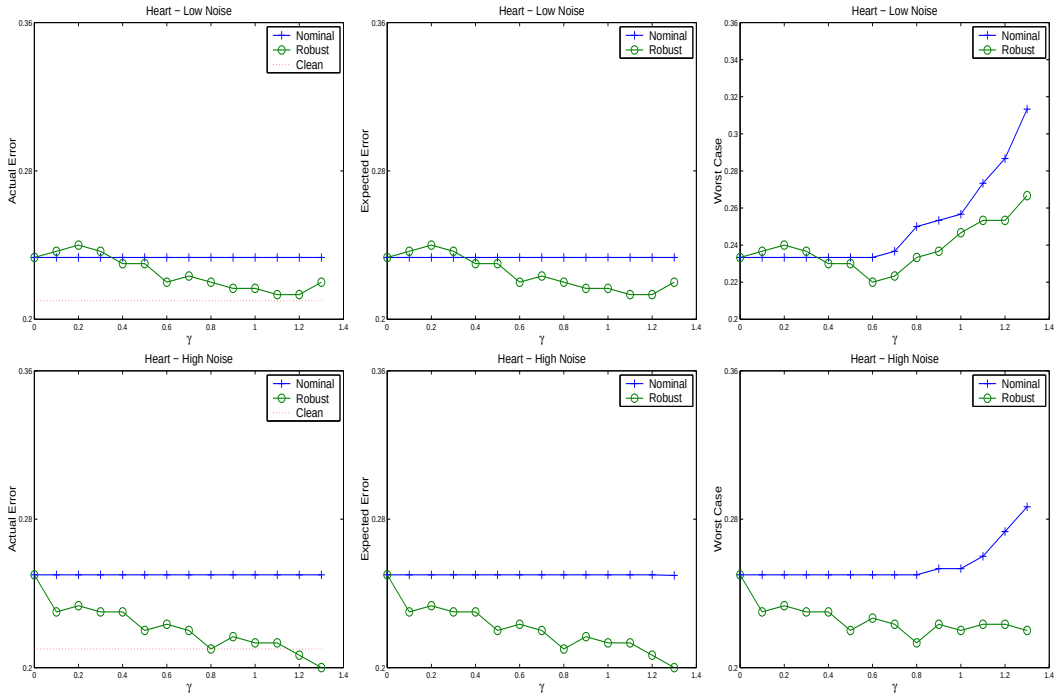


Figure 5: Error rates as a function of γ for Heart. Patterns in the top row contained 50% missing variables, even ones 90%. From left to right — actual error, expected error, and worst case error.

imputed data. The obtained regression function will be called the nominal SVR. In this section we compare our formulations with nominal SVR on a toy dataset and one real world dataset in the linear setting. We also compared the kernelized formulations with the linear formulations.

The first set of results is on a toy data set consisting of 150 observations. Each observation consisted (y,x) pair where

$$y = w^\top x + b, \quad w^\top = [1,2,3,4,5], \quad b = -7.$$

Patterns x were generated from a Gaussian distribution with mean, $\mu = 0$, and randomly chosen covariance matrix, Σ. The results are reported for the following choice of Σ:

$$\begin{bmatrix} 0.1872 & 0.1744 & 0.0349 & -0.3313 & -0.2790 \\ 0.1744 & 0.4488 & 0.0698 & -0.6627 & -0.5580 \\ 0.0349 & 0.0698 & 0.1140 & -0.1325 & -0.1116 \\ -0.3313 & -0.6627 & -0.1325 & 1.3591 & 1.0603 \\ -0.2790 & -0.5580 & -0.1116 & 1.0603 & 0.9929 \end{bmatrix}.$$

Missing values were introduced by randomly choosing 50% of the examples and deleting 2 of the entries example selected at random for each chosen example. The data was divided in the ratio 9:1, the larger one was used for training and the smaller one was used for testing. The results reported here are the average over ten such randomly partitioned training and test data. After imputing

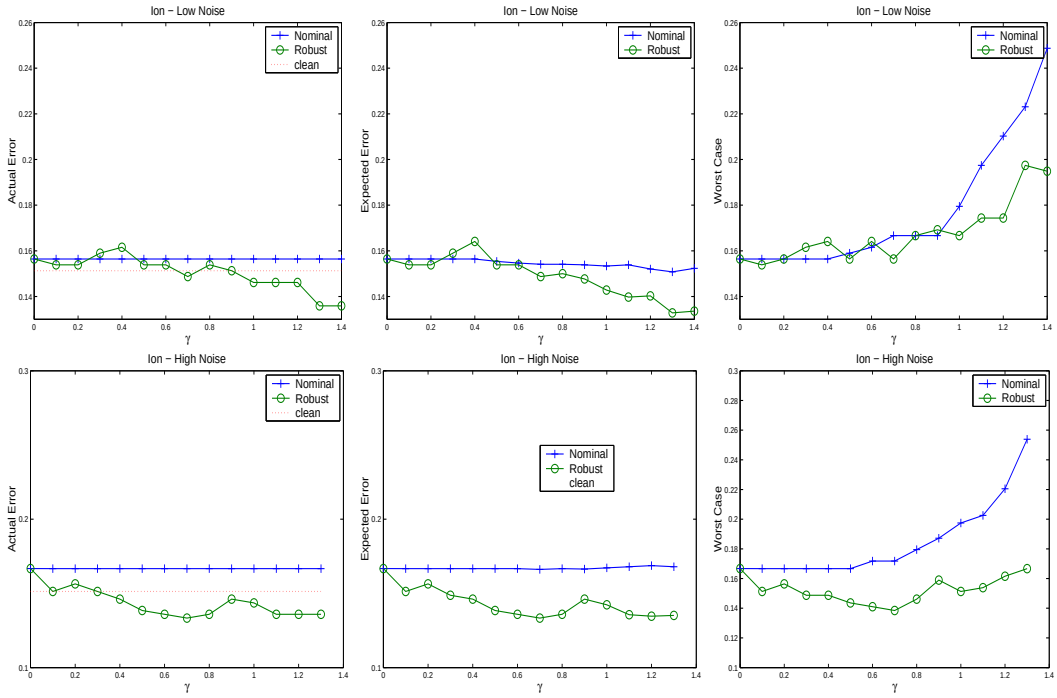


Figure 6: Error rates as a function of γ for Ionosphere. Patterns in the top row contained 50% missing variables, even ones 90%. From left to right — actual error, expected error, and worst case error.

the missing values using a linear regression model, training data was used with different formulations. The first row of Figure 9 shows robustness error (38), worst case error (40) for CTM and expected residual (39) and worst case error (40) for SR. The second row gives the results on UCI Blake and Merz (1998) boston data set with the same test methodology. The performance of our formulation over nominal regression is evident.

To validate our kernelized formulation, 150 samples in $\mathbb{R}^5$ were randomly generated as in the above case. For each x, the output is given by $y = c^T\phi(x) - c_0$, see footnote.[6] The mapping $\phi(x)$ is such that a hyperplane in $\mathbb{R}^{15}$ is actually a quadratic curve in $\mathbb{R}^5$. Randomly generated c and c_0 were used in this mapping. 40% of the values were deleted at random from 50% and 20% of the training samples for CTM and SR, they were filled in using the linear regression model. A Gaussian kernel $K(a,b) = exp(-\gamma\|a-b\|^2)$ with kernel parameter $\gamma = 0.1$ was used. Figure 10 shows the test errors per sample on 10 runs with different randomly deleted values. Test error is the error rate on a test set with missing values filled in. Essentially, we calculate $\sum_{i=1}^{n} e(f(\bar{x}_i, y_i))$ for all the test samples where the missing values are filled in using the training data set parameters using a linear

6. Let $x = [x_1, x_2, \ldots, x_5]$. The mapping $\phi : \mathbb{R}^5 \to \mathbb{R}^{15}$ is defined by

$$\phi(x) = [x_1^2 x_2^2\ x_3^2\ \ x_4^2\ x_5^2\ \ \sqrt{2}x_1x_2\ \sqrt{2}x_1x_3\ \sqrt{2}x_1x_4\ \sqrt{2}x_1x_5\ \sqrt{2}x_2x_3\ \sqrt{2}x_2x_4\ \sqrt{2}x_2x_5\ \sqrt{2}x_3x_4\ \sqrt{2}x_3x_5\ \sqrt{2}x_4x_5]^\top.$$

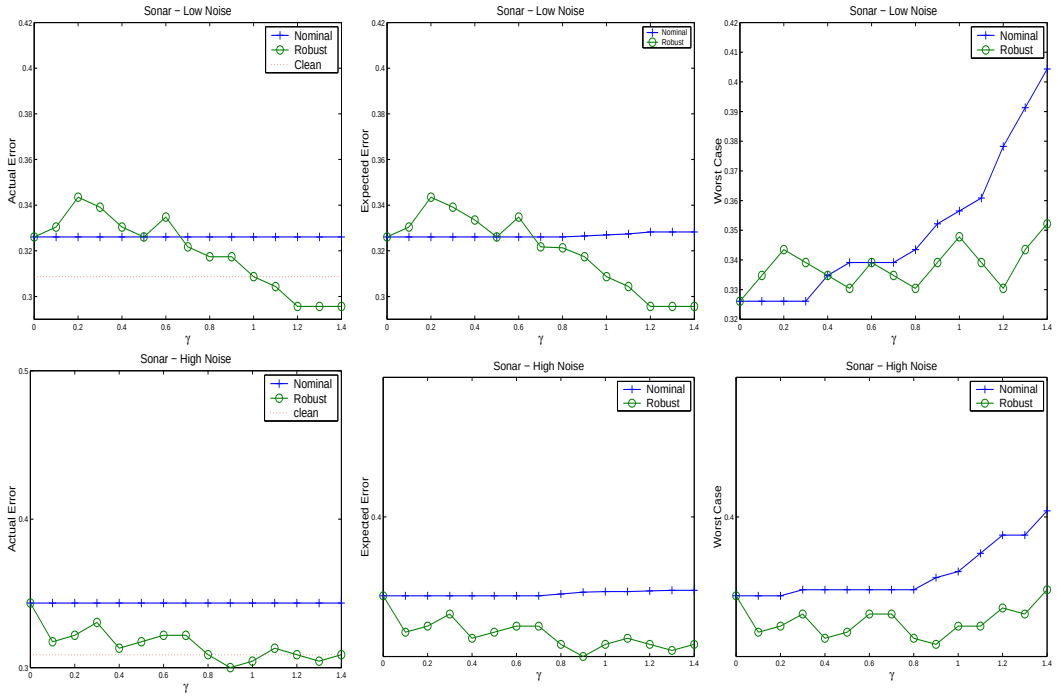


Figure 7: Error rates as a function of γ for Sonar. Patterns in the top row contained 50% missing variables, even ones 90%. From left to right — actual error, expected error, and worst case error.

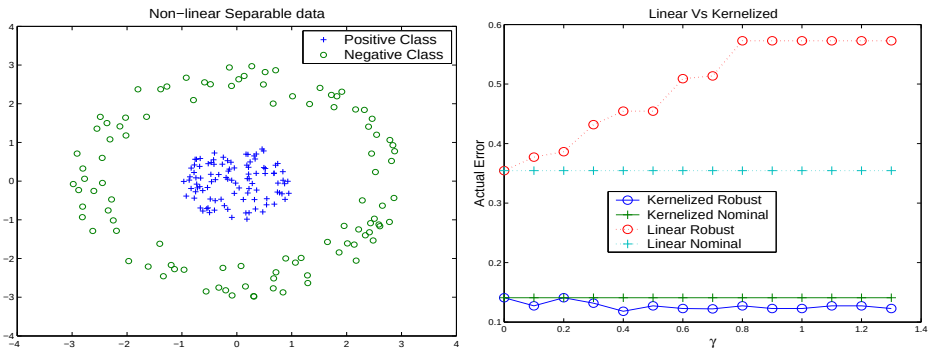


Figure 8: The left figure shows how the data set looks in two dimensions, the right figure gives the actual error rate for linear and kernelized formulations for the robust and nominal cases.

regression model. Essentially it is the absolute residual for imputed mean test data. The figures show that the kernelized version of the robust formulation does a better job than the linear version when the underlying function is non-linear.

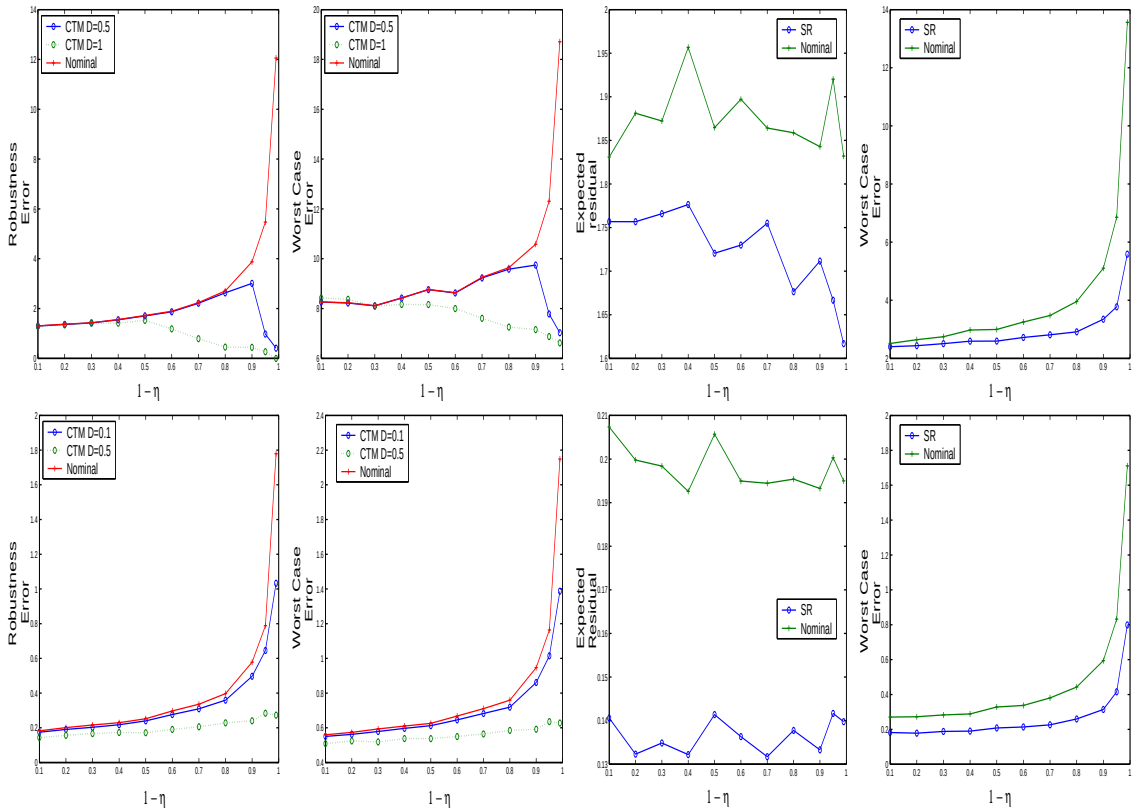


Figure 9: Top row — toy data set, Bottom row — Boston Housing estimation problem; From left to right: robustness (CTM), worst case error (CTM), expected residual (SR), and worst case error (SR). All graphs describe the error as a function of the robustness η.

9. Conclusions

In this paper we have proposed SOCP formulations for designing robust linear prediction functions which are capable of tackling uncertainty in the patterns both in classification and regression setting. The formulations are applicable to any uncertainty distribution provided the first two moments are computable. When applied to the missing variables problem the formulations outperform the imputation based classifiers and regression functions. We have also proposed a way to design nonlinear prediction functions by using regression setting.

The robustness in the context of classification can be geometrically interpreted as requiring that all points in the ellipsoid occur on one side of the hyperplane. Instead of having an ellipsoidal uncertainty one can have situations where the uncertainty is described by arbitrary sets. The constraint sampling approaches can serve as useful alternatives for such problems. Future work will consist in examining this approach for the problem at hand.

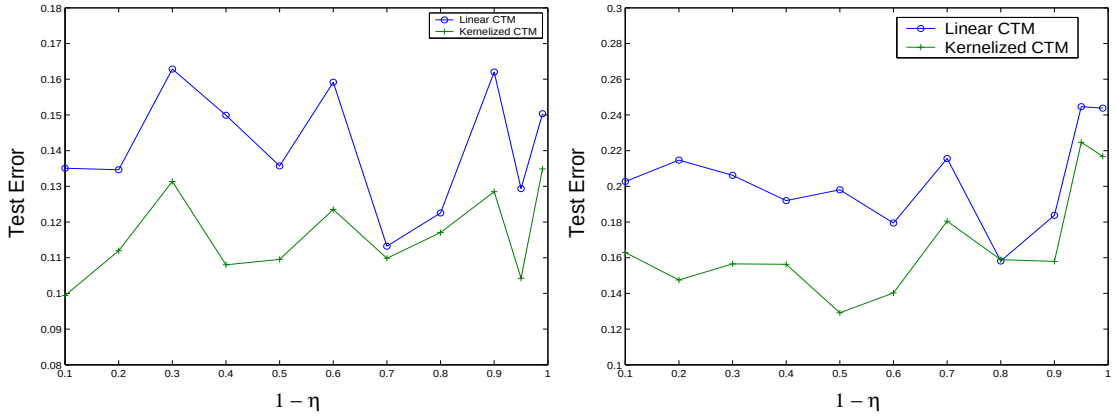


Figure 10: Linear vs. nonlinear regression. Left: CTM formulation, right: SR formulation.

Acknowledgments

CB was partly funded by MHRD (Grant number F26-11/2004). National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. AS was supported by grants of the ARC. We thank Laurent ElGhaoui, Michael Jordan, Gunnar Rätsch, and Frederik Schaffalitzky for helpful discussions and comments.

Appendix A. Dual of the SOCP

The Lagrangian of (44) is given by

$$\begin{aligned}\mathcal{L}(w,\xi,b,\lambda,\beta,\delta) = &\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\beta_i\xi_i - \sum_{i=1}^{c}\lambda_i\left(y_i\left(w^T x_i + b\right) - 1 + \xi_i\right) \\ &- \sum_{j=c+1}^{n}\lambda_j\left(y_j(w^T\bar{x}_j + b) - 1 + \xi_j - \gamma_j\|\Sigma_j^{\frac{1}{2}}w\|\right) + \delta\left(\|w\| - W\right) \qquad (54)\\ &\beta_i,\lambda_i,\delta \geq 0.\end{aligned}$$

Recall that for any $x \in \mathbb{R}^n$ the relationship $\|x\|_2 = \max_{\|x\|\leq 1} x^\top y$ holds. This can be used to handle terms like $\left\|\Sigma_j^{\frac{1}{2}}w\right\|$ and $\|w\|$ leading to a modified Lagrangian given as follows

$$\begin{aligned}\mathcal{L}_1(w,\xi,b,\lambda,\beta,\delta,u) = &\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\beta_i\xi_i - \sum_{i=1}^{c}\lambda_i\left(y_i\left(w^T x_i + b\right) - 1 + \xi_i\right) \\ &- \sum_{j=c+1}^{n}\lambda_j\left(y_j(w^T\bar{x}_j + b) - 1 + \xi_j - \gamma_j\left(\Sigma_j^{\frac{1}{2}}w\right)^T u_j\right) + \delta\left(w^T u_{n+1} - W\right).\end{aligned} \qquad (55)$$

The Lagrangian $\mathcal{L}_1$ has the same optimal value as $\mathcal{L}$ when maximized with respect to u's subject to the constraints $\|u_i\| \leq 1$ for all $c+1 \leq i \leq n+1$. Note that the u's are defined only for patterns with

missing values and u_{n+1} is defined for the constraint $\|w\| \leq W$. Therefore

$$\mathcal{L}_1(w,\xi,b,\lambda,\beta,\delta) = \max_u \mathcal{L}(w,\xi,b,\lambda,\beta,\delta,u) \text{ subject to } \|u_i\| \leq 1 \text{ for all } i \in \{c+1,\ldots n+1\}.$$

By definition, solving (44) is equivalent to finding the saddle-point of the Lagrangian $\mathcal{L}_1$. By virtue of the above reasoning and due to convexity we obtain

$$\underset{w,b,\xi}{\text{minimize}} \underset{\lambda,\delta,\beta}{\text{maximize}} \mathcal{L}(w,\xi,b,\lambda,\beta,\delta) \tag{56a}$$

$$= \underset{w,b,\xi}{\text{minimize}} \underset{\lambda,\delta,\beta,\|u\|\leq 1}{\text{maximize}} \mathcal{L}_1(w,\xi,b,\lambda,\beta,\delta,u) \tag{56b}$$

$$= \underset{\lambda,\delta,\beta,\|u\|\leq 1}{\text{maximize}} \underset{w,b,\xi}{\text{minimize}} \mathcal{L}_1(w,\xi,b,\lambda,\beta,\delta,u). \tag{56c}$$

Eq (56c) now enables us to eliminate the primal variables to give the dual. Taking partial derivatives of $\mathcal{L}$ with respect to $w, b,$ and ξ yields

$$\partial_w \mathcal{L}(w,\xi,b,\lambda,\beta,\delta,u) = -\sum_{i=1}^{c} \lambda_i y_i x_i - \sum_{j=c+1}^{n} \lambda_j \left(y_j \bar{x}_j - \gamma_i {\Sigma_j}^{\frac{1}{2}T} u_j \right) + \delta u_{n+1} \tag{57a}$$

$$\partial_{\xi_i} \mathcal{L}(w,\xi,b,\lambda,\beta,\delta,u) = 1 - \lambda_i - \beta_i \tag{57b}$$

$$\partial_b \mathcal{L}(w,\xi,b,\lambda,\beta,\delta,u) = \sum_{i=1}^{n} \lambda_i y_i. \tag{57c}$$

Changing the sign of u_j for $c+1 \leq i \leq n$ does not matter since the optimal value of maximization of both $w^\top u_j$ and $-w^\top u_j$ over $\|u_j\| \leq 1$ are the same. Substituting $-u_j$ in (57a) by $y_j u_j$ and then equating (57a) , (57b) and (57c) to zero gives

$$\sum_{i=1}^{c} \lambda_i y_i x_i + \sum_{j=c+1}^{n} \lambda y_j \left(\bar{x}_j + \gamma_i {\Sigma_j}^{\frac{1}{2}T} u_j \right) = \delta u_{n+1} \tag{58a}$$

$$1 - \lambda_i - \beta_i = 0 \tag{58b}$$

$$\sum_{i=1}^{n} \lambda_i y_i = 0. \tag{58c}$$

Substituting (58a), (58b) and (58c) in (55) subject to the relevant constraints yields the dual stated as follows

$$\underset{u,\lambda,\beta,\delta}{\text{maximize}} \sum_{i=1}^{n} \lambda_i - W\delta \tag{59a}$$

$$\text{subject to } \sum_{i=1}^{n} \lambda_i y_i = 0 \tag{59b}$$

$$\sum_{i=1}^{c} \lambda_i y_i x_i + \sum_{j=c+1}^{n} \lambda_j y_j \left(\bar{x}_j + \gamma_i {\Sigma_j}^{\frac{1}{2}T} u_j \right) = \delta u_{n+1} \tag{59c}$$

$$\lambda_i + \beta_i = 1 \qquad \text{for all } 1 \leq i \leq n \tag{59d}$$

$$\|u_i\| \leq 1 \qquad \text{for all } c+1 \leq i \leq n+1 \tag{59e}$$

$$\lambda_i, \beta_i, \delta \geq 0 \qquad \text{for all } 1 \leq i \leq n. \tag{59f}$$

For arbitrary data $\delta > 0$, which when plugged into (58a), gives

$$u_{n+1} = \frac{\sum_{i=1}^{c} \lambda_i y_i x_i + \sum_{j=c+1}^{n} \lambda_j y_j \left(\overline{x}_j + \gamma_i \Sigma_j{}^{\frac{1}{2}T} u_j \right)}{\delta}$$

and hence the dual (47) follows.

References

A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Math. Oper. Res.*, 23(4):769–805, 1998.

A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.

K. P. Bennett and O. L. Mangasarian. Multicategory separation via linear programming. *Optimization Methods and Software*, 3:27 – 39, 1993.

C. Bhattacharyya, L. R. Grate, M. I. Jordan, L. El Ghaoui, and Saira I. Mian. Robust sparse hyperplane classifiers: application to uncertain molecular profiling data. *Journal of Computational Biology*, 11(6):1073 – 1089, 2004a.

C. Bhattacharyya, K. S. Pannagadatta, and A. J. Smola. A second order cone programming formulation for classifying missing data. In *Advances in Neural Information Processing Systems (NIPS 17)*, 2004b.

J. Bi and T. Zhang. Support vector classification with input data uncertainty. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, 2004.

C. M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7:108 – 116, 1995.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

G. Calafiore and M. C. Campi. The scenario approach to robust control design. Technical report, Universita di Brescia, 2004. submitted.

S. Chandrasekaran, G. H. Golub, M. Gu, and A. H. Sayed. Parameter Estimation in the Presence of Bounded Data Uncertainties. *SIAM J. Matrix Anal. Appl.*, 19(1):235–252, 1998.

M. Collins. Discriminative training methods for hidden markov models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

D. Pucci de Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society B*, 39(1):1 – 22, 1977.

L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM J. Matrix Anal. Appl.*, 18(4):1035–1064, 1997.

G. Fung, O. L. Mangasarian, and J. W. Shavlik. Knowledge-based support vector machine classifiers. In *Advances in Neural Information Processing Systems 15*, volume 15. MIT Press, 2002.

T. Graepel and R. Herbrich. Invariant pattern recognition by semidefinite programming machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

P. J. Huber. *Robust statistics*. John Wiley, 1982.

G. R. G. Lanckriet, L. E. Ghaoui, C. Bhattacharrya, and M. I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, 3:555–582, 2002.

Q. V. Le, T. Sears, and A. J. Smola. Nonparametric quantile regression. Technical report, National ICT Australia, June 2005. Available at http://sml.nicta.com.au/∼quoc.le.

M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1 - 3):193 – 228, 1998.

A. W. Marshall and I. Olkin. Multivariate chebyshev inequalities. *Annals of Mathematical Statistics*, 31(4):1001–1014, 1960.

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415 – 446, 1909.

Y. Nesterov and A. Nemirovskii. *Interior Point Algorithms in Convex Programming*. Number 13 in Studies in Applied Mathematics. SIAM, Philadelphia, 1993.

G. Rätsch, S. Mika, and A. J. Smola. Adapting codes and embeddings for polychotomies. In *Neural Information Processing Systems*, volume 15. MIT Press, 2002.

T. Schneider. Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, 14:853 – 871, 2001.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11/12(1 - 4):625 – 653, 1999.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, 2003.

V. Vapnik, S. Golowich, and A. J. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281 – 287, Cambridge, MA, 1997. MIT Press.

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599 – 621. Kluwer Academic, 1998.

Journal of Machine Learning Research 7 (2006) 1315–1338 Submitted 8/05; Revised 4/06; Published 7/06

Ensemble Pruning Via Semi-definite Programming

Yi Zhang YI-ZHANG-2@UIOWA.EDU
Samuel Burer SAMUEL-BURER@UIOWA.EDU
W. Nick Street NICK-STREET@UIOWA.EDU
Department of Management Sciences
University of Iowa
Iowa City, IA 52242-1944, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

An ensemble is a group of learning models that jointly solve a problem. However, the ensembles generated by existing techniques are sometimes unnecessarily large, which can lead to extra memory usage, computational costs, and occasional decreases in effectiveness. The purpose of ensemble pruning is to search for a good subset of ensemble members that performs as well as, or better than, the original ensemble. This subset selection problem is a combinatorial optimization problem and thus finding the exact optimal solution is computationally prohibitive. Various heuristic methods have been developed to obtain an approximate solution. However, most of the existing heuristics use simple greedy search as the optimization method, which lacks either theoretical or empirical quality guarantees. In this paper, the ensemble subset selection problem is formulated as a quadratic integer programming problem. By applying semi-definite programming (SDP) as a solution technique, we are able to get better approximate solutions. Computational experiments show that this SDP-based pruning algorithm outperforms other heuristics in the literature. Its application in a classifier-sharing study also demonstrates the effectiveness of the method.

Keywords: ensemble pruning, semi-definite programming, heuristics, knowledge sharing

1. Introduction

Ensemble methods are gaining more and more attention in the machine-learning and data-mining communities. By definition, an ensemble is a group of learning models whose predictions are aggregated to give the final prediction. It is widely accepted that an ensemble is usually better than a single classifier given the same amount of training information. A number of effective ensemble generation algorithms have been invented during the past decade, such as bagging (Breiman, 1996), boosting (Freund and Schapire, 1996), arcing (Breiman, 1998) and random forest (Breiman, 2001). The effectiveness of the ensemble methods relies on creating a collection of diverse, yet accurate learning models.

Two costs associated with ensemble methods are that they require much more memory to store all the learning models, and it takes much more computation time to get a prediction for an unlabeled data point. Although these extra costs may seem to be negligible with a small research data set, they may become serious when the ensemble method is applied to a large scale real-world data set. In fact, a large scale implementation of ensemble learning can easily generate an ensemble with thousands of learning models (Street and Kim, 2001;

Chawla et al., 2004). For example, ensemble-based distributed data-mining techniques enable large companies (like WalMart) that store data at hundreds of different locations to build learning models locally and then combine all the models for future prediction and knowledge discovery. The storage and computation time will become non-trivial under such circumstances.

In addition, it is not always true that the larger the size of an ensemble, the better it is. For example, the boosting algorithm focuses on those training samples that are misclassified by the previous classifier in each round of training and finally squeezes the training error to zero. If there is a certain amount of noise in the training data, the boosting ensemble will overfit (Opitz and Maclin, 1999; Dietterich, 2000). In such cases, it will be better to reduce the complexity of the learning model in order to correct the overfitting, like pruning a decision tree. For a boosting ensemble, selecting a subset of classifiers may improve the generalization performance.

Ensemble methods have also been applied to mine streaming data (Street and Kim, 2001; Wang et al., 2003). The ensemble classifiers are trained from sequential chunks of the data stream. In a time-evolving environment, any change in the underlying data-generating pattern may make some of the old classifiers obsolete. It is better to have a screening process that only keeps classifiers that match the current form of the drifting concept. A similar situation occurs when classifiers are shared among slightly different problem domains. For example, in a peer-to-peer spam email filtering system, each email user can introduce spam filters from other users and construct an ensemble-filter. However, because of the difference of interest among email users, sharing filters indiscriminately is not a good solution. The sharing system should be able to pick filters that fit the individuality of each user.

All of the above reasons motivate the appearance of various ensemble pruning algorithms. A straightforward pruning method is to rank the classifiers according to their individual performance on a held-out test set and pick the best ones (Caruana et al., 2004). This simple approach may sometimes work well but is theoretically unsound. For example, an ensemble of three identical classifiers with 95% accuracy is worse than an ensemble of three classifiers with 67% accuracy and least pairwise correlated error (which is perfect!). Margineantu and Dietterich (1997) proposed four approaches to prune ensembles generated by Adaboost. KL-divergence pruning and Kappa pruning aim at maximizing the pairwise difference between the selected ensemble members. Kappa-error convex hull pruning is a diagram-based heuristic targeting a good accuracy-divergence trade-off among the selected subset. Back-fitting pruning is essentially enumerating all the possible subsets, which is computationally too costly for large ensembles. Prodromidis et al. invented several pruning algorithms for their distributed data mining system (Prodromidis and Chan, 2000; Chan et al., 1999). One of the two algorithms they implemented is based on a diversity measure they defined, and the other is based on class specialty metrics. The major problem with the above algorithms is that when it comes to optimizing some criteria of the selected subset, they all resort to greedy search, which is on the lower end of optimization techniques and usually without either theoretical or empirical quality guarantees. Kim et al. used an evolutionary algorithm for ensemble pruning and it turned out to be effective (Kim et al., 2002). A similar approach can also be found in (Zhou et al., 2001).

Unlike previous heuristic approaches, we formulate the ensemble pruning problem as a quadratic integer programming problem to look for a subset of classifiers that has the opti-

mal accuracy-diversity trade-off. Using a state-of-the-art semi-definite programming (SDP) solution technique, we are able to get a good approximate solution efficiently, although the original problem is NP-hard. In fact, SDP is not new to the machine learning and data mining community. It has been used for problems such as feature selection (d'Aspremont et al., 2004) and kernel optimization (Lanckriet et al., 2004). Our new SDP-based ensemble pruning method is tested on a number of UCI repository data sets with Adaboost as the ensemble generation technique and compares favorably to two other metric-based pruning algorithms: diversity-based pruning and Kappa pruning. The same subset selection procedure is also applied to a classifier sharing study. In that study, classifiers trained from different but closely related problem domains are pooled together and then a subset of them is selected and assigned to each problem domain. Computational results show that the selected subset performs as well as, and sometimes better than, including all elements of the ensemble.

Ensemble pruning can be viewed as a discrete version of weight-based ensemble optimization. The more general weight-based ensemble optimization aims to improve the generalization performance of the ensemble by tuning the weight on each ensemble member. If the prediction target is continuous, derivative methods can be applied to obtain the optimal weight on each ensemble model (Krogh and Vedelsby, 1995; Zhou et al., 2001; Hashem, 1997). In terms of classification problems, approximate mathematical programs are built to look for good weighting schemes (Demiriz et al., 2002; Wolpert, 1992; Mason et al., 1999). Those optimization approaches are effective in performance enhancement according to empirical results and are sometimes able to significantly reduce the size the ensemble when there are many zeros in the weights (Demiriz et al., 2002). However, size-reduction is not explicitly built into those programs and there is thus no control over the final size of the ensemble. The proposed ensemble pruning method distinguishes from the above methods by explicitly constraining the weights to be binary and using a cardinality constraint to set the size of the final ensemble. The goal of ensemble pruning is to contain the size of the ensemble without compromising its performance, which is subtly different from that of general weight-based ensemble optimization.

The rest of the paper is organized as follows. Section 2 describes the pruning algorithm in detail, including the mathematical formulation and the solution technique. Section 3 shows the experimental results on the UCI repository data sets and compares our method with other pruning algorithms. Section 4 is devoted to the algorithm's application in a classifier-sharing case study with a direct marketing data set. Section 5 concludes the paper.

2. Problem Formulation and Solution Technique

As the literature has shown, a good ensemble should be composed of classifiers that are not only accurate by themselves, but also independent of each other (Krogh and Vedelsby, 1995; Margineantu and Dietterich, 1997; Breiman, 2001), or in other words, they should make different errors. Some previous work has demonstrated that making errors in an uncorrelated manner leads to a low error rate (Hansen and Salamon, 1990; Perrone and Cooper, 1993). The individual accuracy and pairwise independence of classifiers in an ensemble are often referred to as strength and divergence of the ensemble. Breiman (2001)

showed that the generalization error of an ensemble is loosely bounded by $\frac{\bar{\rho}}{s^2}$, where $\bar{\rho}$ is the average correlation between classifiers and s is the overall strength of the classifiers. For continuous prediction problems, there are even closed-form representations for the ensemble generalization performance based on individual error and diversity. Krogh and Vedelsby (Krogh and Vedelsby, 1995) showed that for a neural network ensemble, the generalization error

$$E = \bar{E} - \bar{A},$$

where $\bar{E}$ is the weighted average of the error of the individual networks and $\bar{A}$ is the variance among the networks. Zhou et al. (2001) give another form,

$$E = \sum_{i,j} C_{ij},$$

where

$$C_{ij} = \int p(x)\Big(f_i(x) - d(x)\Big)\Big(f_j(x) - d(x)\Big)\,dx,$$

$p(x)$ is the density of input x, $f_i(x)$ is the output of the ith network and $d(x)$ is the true output. Note that C_{ii} is the error of the ith network and $C_{ij,\,i\neq j}$ is a pairwise correlation-like measurement.

The problem is that the more accurate the classifiers are, the less different they become. Therefore, there must be a trade-off between the strength and the divergence of an ensemble. What we are looking for is a subset of classifiers with the best trade-off so that the generalization performance can be optimized.

In order to get the mathematical formulation of the ensemble pruning problem, we need to represent the error structure of the existing ensemble in a nice way. Unlike the case of continuous prediction, there is no exact closed-form representation for the ensemble error in terms of strength and diversity for a discrete classification problem. However, we are still able to obtain some approximate metrics following the same idea. From the error analysis of continuous problems, we notice that the ensemble error can be represented by a linear combination of the individual accuracy terms and pairwise diversity terms. Therefore, if we are able to find strength and diversity measurements for a classification ensemble, a linear combination of them should serve as a good approximation of the overall ensemble error. Minimizing this approximate ensemble error function will be the objective of the mathematical programming formulation.

First, we record the misclassifications of each classifier on the training set in the error matrix P as follows:

$$\begin{aligned} P_{ij} &= 0, \quad \text{if } j\text{th classifier is correct on data point } i, \\ P_{ij} &= 1, \quad \text{otherwise.} \end{aligned} \tag{1}$$

Let $G = P^T P$. Thus, the diagonal term G_{ii} is the total number of errors made by classifier i and the off-diagonal term G_{ij} is the number of common errors of classifier pair i and j. To put all the elements of the G matrix on the same scale, we normalize them by

$$\begin{aligned} \tilde{G}_{ii} &= \frac{G_{ii}}{N}, \\ \tilde{G}_{ij,\,i\neq j} &= \frac{1}{2}\left(\frac{G_{ij}}{G_{ii}} + \frac{G_{ij}}{G_{jj}}\right), \end{aligned} \tag{2}$$

where N is the number of training points. After normalization, all elements of the $\tilde{G}$ matrix are between 0 and 1. $\tilde{G}_{ii}$ is the error rate of classifier i and $\tilde{G}_{ij}$ measures the overlap of errors between classifier pair i and j. Note that $\frac{G_{ij}}{G_{ii}}$ is the conditional probability that classifier j misclassifies a point, given that classifier i does. Taking the average of $\frac{G_{ij}}{G_{ii}}$ and $\frac{G_{ij}}{G_{jj}}$ as the off-diagonal elements of $\tilde{G}$ makes the matrix symmetric. The constructed $\tilde{G}$ matrix captures both the strength (diagonal elements) and the pairwise divergence (off-diagonal elements) of the ensemble classifiers. It is self-evident that for a good ensemble, all elements of the $\tilde{G}$ matrix should be small. Intuitively, $\sum_i \tilde{G}_{ii}$ measures the overall strength of the ensemble classifiers and $\sum_{ij,i\neq j} \tilde{G}_{ij}$ measures the diversity. A combination of these two terms, $\sum_{ij} \tilde{G}_{ij}$ should be a good approximation of the ensemble error. The diversity term defined here is ad hoc. We have noticed that there exist many other heuristic pairwise measurements for ensemble diversity. For instance, the disagreement measure (Ho, 1998), κ statistic (Fleiss, 1981), Yule's Q statistic (Yule, 1900), and so on. However, we found that the choice of the diversity measurement did not make a significant difference in terms of performance according to our computational experiments. Therefore, we stick to our definition because of its simplicity and intuitive appeal.

Now we can formulate the subset selection problem as a quadratic integer programming problem. Essentially, we are looking for a fixed-size subset of classifiers, with the sum of the corresponding elements in the $\tilde{G}$ matrix minimized. The mathematical programming formulation is as follows,

$$\begin{aligned} \min_{x} \quad & x^T \tilde{G} x \\ \text{s.t.} \quad & \sum_i x_i = k, \\ & x_i \in \{0,\ 1\}. \end{aligned} \tag{3}$$

The binary variable x_i represents whether the ith classifier will be chosen. If $x_i = 1$, which means that the ith classifier is included in the selected set, its corresponding diagonal and off-diagonal elements will be counted in the objective function. Note that the cardinality constraint $\sum_i x_i = k$ is mathematically important because without it, there is only one trivial solution to the problem with none of the classifiers picked. In addition, it gives us control over the size of the selected subset.

This quadratic integer programming problem is a standard 0-1 optimization problem, which is NP-hard in general. Fortunately, we found that this formulation is close to that of the so-called "max cut with size k" problem (written MC-k), in which one partitions the vertices of an edge-weighted graph into two sets, one of which has size k, so that the total weight of edges crossing the partition is maximized. The MC-k problem can be formulated as

$$\begin{aligned} \max_{y} \quad & \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j) \\ \text{s.t.} \quad & \sum_i y_i = N_v - 2k, \\ & y_i \in \{-1,\ 1\}. \end{aligned} \tag{4}$$

where N_v is the number of the vertices in the graph.

Roughly speaking, this optimization involves partitioning the vertices via the assignment of $y_i = 1$ or $y_i = -1$ to each vertex i in the graph (subject to the size requirement) and minimizing the sum $\sum_{ij} w_{ij} y_i y_j$, where w_{ij} is the edge weight between vertices i and j. Notice that the interaction term $y_i y_j$ equals -1 when i and j are in different sets of the partition, which, in the context of the minimization described, contributes to the maximization of edges crossing the partition. The MC-k problem is known to have a very good approximate solution algorithm based on semi-definite programming (SDP). The key point of the SDP approximation algorithm is to relax each binary variable $y_i \in \{-1, 1\}$ into a unit vector. Therefore, if we are able to transform the ensemble pruning formulation so that it fits into the framework of MC-k, we may obtain a good solution for the ensemble pruning problem.

The above MC-k formulation (4) is equivalent to

$$\begin{aligned} \min_{y} \quad & y^T W y \\ \text{s.t.} \quad & \sum_i y_i = N_v - 2k, \\ & y_i \in \{-1, 1\}. \end{aligned} \tag{5}$$

where W is the edge-weight matrix, with $w_{i,i} = 0$. If we compare this formulation with that of the ensemble pruning problem, the only barrier that prevents the application of the SDP approximation algorithm on the ensemble pruning problem is the difference in the possible values of the binary variable. Specifically, in ensemble pruning, $x_i \in \{0, 1\}$ and in MC-k, $y_i \in \{-1, 1\}$. Therefore, we need to make a transformation of variables for the ensemble pruning problem. Let

$$x_i = \frac{v_i + 1}{2}, \tag{6}$$

and $v_i \in \{-1, 1\}$. Now the objective function becomes

$$\frac{1}{4}(v + e)^T \tilde{G}(v + e), \tag{7}$$

where e is a column vector of all 1s. The cardinality constraint $\sum_i x_i = k$ can be rewritten into quadratic form

$$x^T I x = k, \tag{8}$$

where I is the identity matrix. After variable transformation, this constraint becomes

$$(v + e)^T I (v + e) = 4k. \tag{9}$$

A variable expansion trick can be applied to put both the transformed objective function and the cardinality constraint back into a nice quadratic form. We expand the variable vector $v = (v_1, v_2, ..., v_n)$ into $v = (v_0, v_1, v_2, ..., v_n)$, and let $v_0 = 1$. We then construct a new matrix

$$H_{(n+1)\times(n+1)} = \begin{pmatrix} e^T \tilde{G} e & e^T \tilde{G} \\ \tilde{G} e & \tilde{G} \end{pmatrix}. \tag{10}$$

Thus the objective function is equivalent to $v^T H v$ (dropping the coefficient). We use the same trick to construct a matrix D

$$D_{(n+1)\times(n+1)} = \begin{pmatrix} n & e^T \\ e & I \end{pmatrix}, \tag{11}$$

so that the cardinality constraint becomes $v^T D v = 4k$.

After this whole transformation, the problem formulation becomes

$$\begin{aligned} \min_{v} \quad & v^T H v \\ \text{s.t.} \quad & v^T D v = 4k, \\ & v_0 = 1, \\ & v_i \in \{-1,\ 1\},\ \forall i \neq 0, \end{aligned} \tag{12}$$

with which we will show later that the SDP relaxation applicable to the MC-k formulation (5) (Goemans and Williamson, 1995; Han et al., 2002) may also be applied here.

It is important to keep in mind that the equivalence of our pruning problem and MC-k, established through the transformations (4–5) and (6–12), is an equivalence between optimal solution sets, not optimal values. Said differently, the optimal solutions of our problem are in one-to-one correspondence with the optimal solutions of MC-k via the given transformations, but the optimal values of the two problems are not equal. In particular, even though we can interpret (12) as an instance of MC-k with nonnegative weights on the complete graph, the objective function of (12) does not measure the total weight of cut edges. Instead, it more closely measures the total weight of un-cut edges (which is consistent with minimization) and, further, differs by a scaling as well as the inclusion of a constant term, which is based on the diagonal entries of H. These differences in the objective functions are byproducts of the transformations employed.

Given that our problem is equivalent to MC-k (in terms of optimal solutions), it is worthwhile to ask what is known about MC-k. Based on the seminal work of Goemans and Williamson (1995) for the maximum cut problem with no size restriction, the papers of Feige and Langberg (2001) and Han, Ye, and Zhang (2002) demonstrate an approximation algorithm for instances of MC-k with nonnegative edge weights. The approximation algorithm is based on solving a SDP relaxation of MC-k and applying a simple-to-implement randomization scheme.

Because the objective function of our problem does not precisely match the objective of MC-k (even though the two problems are equivalent through optimal solutions), the approximation guarantee for MC-k—which is with respect to its objective function—does not necessarily transform into a guarantee for our problem. In fact, it seems extremely difficult to derive directly any approximation guarantees for our problem. SDP-based approximation approaches have proven most successful for problems in maximization form due to technical details concerning how an SDP relaxes a binary quadratic program, and unfortunately, our problem is in minimization form.

Nevertheless, we feel that the strong connection of our problem with MC-k justifies the following heuristic procedure: (i) transform an instance of our problem into an instance of MC-k; (ii) use the SDP-based approximation algorithm to obtain a good solution of the

MC-k instance; and (iii) transform back to a solution of our problem. Further, it is not difficult to see that the above three steps are equivalent to the more direct approach of relaxing (12) as an SDP and applying the randomization scheme of Feige and Langberg (2001) and Han, Ye, and Zhang (2002). In other words, it is not explicitly necessarily to convert to an instance of MC-k first.

For completeness, we now return our attention to the SDP relaxation itself. Problem (12) is equivalent to

$$\begin{array}{ll} \min\limits_{v} & H \bullet vv^T \\ \text{s.t.} & D \bullet vv^T = 4k, \\ & v_0 = 1, \\ & v_i \in \{-1,\ 1\},\ \forall i \neq 0, \end{array} \tag{13}$$

where $A \bullet B = \sum_{i,j} A_{ij}B_{ij}$.

To construct the relaxation, we first note that the constraint $v_0 = 1$ can be relaxed to $v_0 \in \{-1, 1\}$ without changing the problem since $-v$ is feasible for the remaining constraints if and only if v is and since $H \bullet vv^T = H \bullet (-v)(-v)^T$. Next, we rewrite the constraints $v_i \in \{-1, 1\}$, $i = 0, 1, \ldots, n$ as the single, collective constraint $\text{diag}(vv^T) = e$ to arrive at the following formulation:

$$\begin{array}{ll} \min\limits_{v} & H \bullet vv^T \\ \text{s.t.} & D \bullet vv^T = 4k, \\ & \text{diag}(vv^T) = e. \end{array} \tag{14}$$

We next substitute $V = vv^T$, and note that V can be expressed as vv^T if and only if $V \succeq 0$ with $\text{rank}(V) = 1$, which gives us

$$\begin{array}{ll} \min\limits_{V} & H \bullet V \\ \text{s.t.} & D \bullet V = 4k, \\ & \text{diag}(V) = e \\ & V \succeq 0, \quad \text{rank}(V) = 1. \end{array} \tag{15}$$

Although this problem is written in a different form, it is completely equivalent to our original 0-1 quadratic problem.

The SDP relaxation is now obtained by dropping the rank constraint, which yields the following (convex) SDP:

$$\begin{array}{ll} \min\limits_{V} & H \bullet V \\ \text{s.t.} & D \bullet V = 4k, \\ & \text{diag}(V) = e \\ & V \succeq 0. \end{array} \tag{16}$$

Now the original NP-hard problem (3) is relaxed into a convex SDP problem which can be solved to any preset precision in polynomial time. We solve the SDP relaxation using the publicly available package SDPLR (Burer and Monteiro, 2003; SDPLR) and have implemented the approximation algorithm described in (Han et al., 2002).

3. Computational Experiments

The SDP-based ensemble pruning algorithm is tested on sixteen UCI repository data sets (Blake and Merz, 1998): Autompg, Bupa, Cmc, Crx, Glass, Haberman, Housing, Cleveland-heart-disease, Hepatitis, Ion, Pima, Sonar, Vehicle, WDBC, Wine and WPBC. Some of the data sets do not originally depict two-class problems so we did some transformation on the dependent variables to get binary class labels. Specifically in our experiments, the Autompg data is labeled by whether the mileage is greater than 25mpg, the Housing data by whether the value of the house exceeds $\$25,000$, and the Cleveland-heart-disease by the presence or absence of the disease. Vehicle and Wine are multi-class problems so we set the problem as separating one class pair each time, resulting in a total of 24 data sets. All data points with missing values are removed.

It has been shown that the pruning effect is more striking on a diverse ensemble (Margineantu and Dietterich, 1997). Boosting-like algorithms usually generate diverse classifiers by concentrating on widely different parts of the training samples at each round of training. So we use Adaboost to create the original ensemble for pruning. The unpruned C4.5 decision tree is used as the base classifier training algorithm (Quinlan, 1993). One hundred decision trees are built for each data set following the standard Adaboost procedure. If the training error reaches zero before the number of trees gets 100, the Adaboost process is repeated (with different random seeds). Note that the accuracy of all the classifiers created by Adaboost is higher than 50%.

Two existing metric-based ensemble pruning algorithms, diversity-based pruning and kappa pruning, are picked as the benchmarks because the objectives of the these two algorithms are somewhat close to that of the new algorithm. In addition, these two algorithms can prune the ensemble to any pre-set size, which makes a fair comparison possible. Diversity-based pruning tries to maximize the sum of pairwise diversity of the selected subset of classifiers (Prodromidis and Stolfo, 1998). Pairwise diversity is defined as the proportion of the training points on which the two classifiers disagree. The greedy search starts with the most accurate classifier and adds the classifier that improves the objective most at each round. Kappa-pruning, invented by Margineantu and Dietterich (Margineantu and Dietterich, 1997), attempts to minimize the sum of pairwise similarity of the selected subset. A κ statistic is used to measure the pairwise similarity,

$$\kappa = \frac{\theta_1 - \theta_2}{1 - \theta_2}$$

where θ_1 is the proportion of points on which the two classifiers agree with each other on the training set, and θ_2 is the probability that the two classifiers agree purely by chance. The κ statistics among all pairs are calculated and ranked from low to high. The greedy search picks classifier pairs from the ranked pair list until the pre-set size of the pruned ensemble is met.

The performance of the three algorithms on the 24 data sets are listed in Table 3. Here, the size of the pruned ensemble is 25. Empirical research suggests that, in most cases, most or all of the generalization gain in a well-constructed ensemble comes from the first 25 classifiers added (Breiman, 1996; Opitz and Maclin, 1999). The result is averaged over five ten-fold cross-validations. The number in the parentheses is the standard deviation over the five runs. A win-loss-tie summarization based on mean value and t test (95%

dataset	SDP-25	Div-25	Kappa-25	No Pruning
AUTOMPG	10.35(0.62)	13.88(0.83)	11.91(1.52)	10.71(0.70)
BUPA	30.52(1.43)	35.87(1.25)	38.39(2.65)	30.32(0.43)
CMC	32.82(0.87)	41.70(1.66)	43.66(1.37)	34.50(1.19)
CRX	13.88(0.46)	22.40(3.41)	21.78(4.44)	13.58(0.85)
GLASS	12.53(1.12)	17.30(2.88)	16.39(2.76)	11.29(0.70)
HABERMA	32.73(2.21)	38.63(1.30)	38.88(3.09)	34.56(1.74)
HEART	20.13(2.19)	28.09(2.73)	27.81(4.09)	20.40(2.39)
HEPATIT	15.81(1.17)	19.83(2.66)	16.86(2.00)	14.71(1.84)
HOUSING	11.03(0.61)	12.37(0.91)	12.21(1.59)	10.67(0.66)
ION	7.46(2.17)	10.94(5.13)	14.02(10.61)	9.85(7.72)
IRIS	12.20(9.47)	15.00(7.84)	19.60(8.44)	13.40(11.01)
PIMA	25.34(0.67)	31.31(3.78)	30.24(2.90)	25.06(0.78)
SONAR	21.43(1.68)	26.24(4.32)	25.24(2.31)	18.96(1.13)
WDBC	3.38(0.34)	3.51(0.59)	3.76(0.76)	2.88(0.30)
WINE1-2	2.92(0.64)	6.15(3.17)	12.62(18.45)	11.85(18.82)
WINE1-3	1.11(0.76)	1.27(0.50)	2.58(1.01)	1.31(0.47)
WINE2-3	3.05(0.75)	4.56(2.50)	5.67(6.27)	2.86(3.54)
WPBC	24.06(2.51)	32.35(1.88)	29.54(3.52)	24.04(2.79)
VEHICLE1-2	41.15(2.62)	42.32(1.01)	45.17(4.65)	41.40(1.28)
VEHICLE1-3	1.07(0.42)	3.67(3.79)	9.26(12.11)	3.21(3.24)
VEHICLE1-4	4.52(0.36)	6.47(1.24)	6.18(2.42)	3.84(0.21)
VEHICLE2-3	2.25(0.55)	7.34(4.01)	11.99(7.06)	5.82(4.36)
VEHICLE2-4	5.00(1.28)	10.33(3.26)	13.57(12.65)	5.96(4.35)
VEHICLE3-4	1.15(0.11)	0.96(0.41)	1.39(1.63)	0.67(0.39)
	Absolute W-L-T	23-1-0	24-0-0	12-12-0
	Significant W-L-T	9-0-15	8-0-16	2-0-22

Table 1: Comparison of SDP pruning, Diversity-based pruning, Kappa-pruning and original ensembles, by % error and (standard deviation)

significance level) is attached at the bottom of the table. Note that simple majority voting is used to combine the predictions of the ensembles. Prodromidis et al. (Prodromidis and Stolfo, 1998; Prodromidis and Chan, 2000) built a higher level classifier (meta-classifier) to aggregate the output of the pruned ensembles. There has been other research in this direction (Wolpert, 1992; Bennett et al., 2000; Mason et al., 1999; Grove and Schuurmans, 1998). However, there is so far no strong evidence that such a meta-classifier is generally better than simple majority voting.

Table 3 shows that the performance of the SDP-based pruning is better than that of the other two algorithms for most of the data sets involved in the computational experiments. Also, although only a quarter of the classifiers are left, the error of the pruned ensemble by SDP-based pruning is statistically the same as that of the original ensemble. In addition, we may conclude that the SDP-based pruning is more stable in terms of accuracy, by looking at the error standard deviation of the three algorithms. The error fluctuation range of the other two algorithms is often much larger, which might explain why the SDP-based pruning is sometimes not statistically better.

There are several possible reasons why the SDP-based pruning outperforms the other two. First, the optimization procedure of the SDP-based pruning is better than greedy search. Although greedy search may produce optimal solutions in some cases, it may also perform badly when the problem is ill-conditioned, which is not a rare case for ensemble pruning. On the other hand, while the SDP approximation algorithm may not be able find optimal solutions all the time, it can provide a relatively good solution in most cases. Therefore, the SDP-based pruning turns out to be better on average and more stable. Second, the definition of divergence of the SDP-based pruning is subtly different from that of the two others. SDP-based pruning only considers the correlation between errors while the other two define pair-wise correlation based on all the data points. In fact, we want the classifiers to agree with each other when they make correct predictions. The divergence we need is indeed the divergence of the way they commit errors. Hence, the error-based divergence definition is more closely related to generalization performance. Finally, the objective function of the SDP-based pruning may reflect a better trade-off between individual accuracy and pair-wise divergence. As a matter of fact, the other two algorithms do not explicitly include individual accuracy in their objective functions. The diversity-based pruning starts the search with the most accurate classifier, and both favor more accurate classifiers when the diversity measures tie. For a systematically constructed ensemble, the individual accuracy of each classifier is not critical as long as its accuracy is over 50% and there are enough of classifiers in the ensemble. However, for a pruned ensemble, with the initial structure broken and a large portion of the members removed, the importance of individual accuracy may increase. Therefore, the individual strength of the classifiers should be appropriately considered in ensemble pruning. Although our way of including individual accuracy in the objective may not be perfect, it may contribute to the improvement of performance.

The improved performance comes with a price: more computation time. Table 3 shows the time it takes to solve the pruning problem with increasing size of the original ensemble. The size of the pruned ensemble is fixed at 25. The timing test was conducted on an AMD 1.2GHz computer with 1G memory. As we mentioned before, with the SDP relaxation heuristic, the pruning problem can be solved in polynomial time. This can be verified by looking at the linearity of the $\log \log$ plot of time vs. size of the original ensemble, as illustrated in Figure 1. Selecting 25 classifiers from an original ensemble with 3000 classifiers will take roughly a day, which is within the reasonable region for most applications. If each classifier (a decision tree, e.g.) is trained with 1000 points, such an ensemble learning procedure would be able to handle a dataset with 3,000,000 points. Therefore, we may cautiously conclude that the SDP-based ensemble pruning method is applicable to large scale data mining problems. Note that our subset selection routine is coded partly in MATLAB. If it were coded in C, it would have been much faster.

4. Selective Sharing of Classifiers Among Related Problem Domains: A Case Study

We have so far demonstrated the effectiveness of the SDP-based pruning algorithm based on small research data sets. As mentioned before, ensemble pruning is more relevant to large-scale data mining implementations. Therefore, we wish to demonstrate that the SDP-based pruning algorithm is also effective and efficient under real-world conditions. We imple-

Size	Time (s)	Size	Time (s)
100	3	1500	4889
200	36	2000	23255
400	113	2500	42419
800	713	3000	83825

Table 2: Computation time of SDP-based pruning

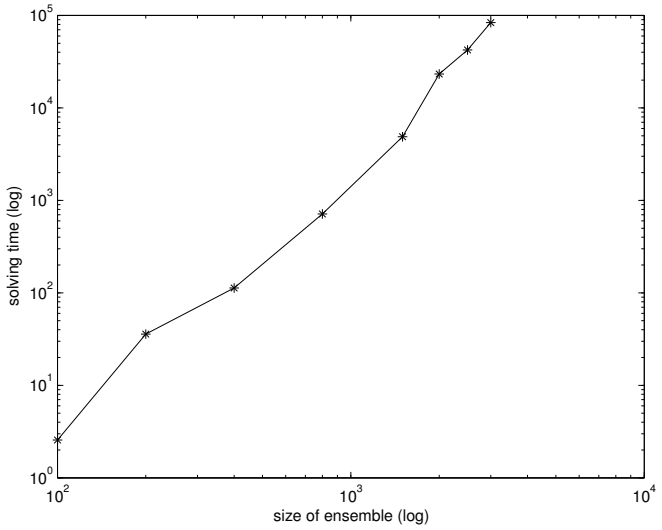


Figure 1: Timing test of SDP-based ensemble pruning

mented the pruning algorithm in the following case study, where classifiers from different but closely-related problem domains are pooled together and a subset of them is then selected for each problem domain. This case study involves a much larger data set than the UCI sets used in the above experiments, and an original ensemble with a larger number of more divergent classifiers. The results of the study verify the algorithm's performance in real-world applications, and show one type of situation where ensemble pruning can be particularly useful.

As we know, ensemble methods not only improve classification accuracy, but also provide a way to share knowledge. If the data for a problem domain are distributed at different locations, classifiers can be trained locally and then combined to create an ensemble. This centralized ensemble gathers information from each site and can potentially be more powerful for further predictions. Now the questions is, if we are working with several different but closely related problem domains, is sharing classifiers among those domains still a good idea?

The essence of sharing classifiers is sharing common knowledge among different but closely related problem domains. A famous example of the research in this direction is the multi-task neural network (Caruana, 1997). Each problem domain is assigned one or more output nodes, while sharing some of the input and mid-layer nodes with other problem domains. Although this method is sometimes successful, it has several limitations. First, it requires that the data be at the central location for training. Second, the training speed becomes a big issue if the size of the data is large. The classifier-sharing strategy may avoid both of the drawbacks. Since the classifiers can be trained locally, the data need not be centralized. Moreover, one can use efficient algorithms like decision trees to train classifiers, so computation time becomes less of a problem.

The prerequisite for sharing classifiers among different problem domains is that the data schema for each problem domain should be identical, or at least very similar. It ensures that the classifiers trained on one problem domain can also be applied to other problem domains, although the accuracy can possibly be low. Sometimes, this requires transformation of variables to satisfy this condition. It is in fact hard to tell a priori whether sharing classifiers among different problem domains will improve the overall performance. The success of this strategy depends on the connections among the problem domains and the self-completeness of information within each problem domain. However, with a screening process (ensemble pruning) that will be described later, the chance that sharing classifiers will eventually downgrade the overall performance is minimal.

The cross-domain classifier-sharing strategy is tested on a publicly available marketing data set. To address the concern that sharing classifiers blindly sometimes may do harm to some of the problem domains if there exist conflicting elements among them, we apply the SDP-based pruning algorithm to select a good subset of classifiers from the entire ensemble for each problem domain.

The data set is a catalog marketing data set from the Direct Marketing Association (DMEF Academic Data Set Three, Specialty Catalog Company, Code 03DMEF). The dependent variables are the customers' responses to a particular promotion held by a catalog company. There are 19 different categories of products involved in the promotion, which correspond to 19 response variables. Unfortunately, the identities of the product categories are not available. The data mining problem we try to solve is to predict which categories

Category	%Pos	Category	%Pos
1	0.14	11	0.13
2	0.27	12	0.05
3	0.04	13	0.44
4	0.92	14	2.65
5	0.12	15	1.70
6	0.83	16	2.09
7	0.44	17	0.65
8	0.37	18	0.31
9	0.64	19	1.17
10	0.02		

Table 3: Percentage of positive points (response rate) of 19 categories.

of products a customer is going to buy based on the available historical and demographic data. We decomposed this task into 19 separate binary classification problems, building one model for each category and predicting whether the customer is going to buy from that category. Note that this whole task cannot be treated as a classification problem with 19 classes because one customer can buy products from any number of categories, including zero, so the class assignment for each individual is not exclusive. This kind of problem is sometimes referred to as "multi-label" classification problem and is more often seen in the text-mining domain (McCallum, 1999; Shen et al., 2004).

Table 4 shows the percentage of positive points, or response rate, of each category. A positive point for a category represents a customer that buys one or more products from that category. It can be seen from the table that for most categories, the response rate is lower than one percent, which makes the data set highly skewed toward the non-buyers. A common way to deal with unbalanced data is to create training data sets with a balanced class distribution through sampling (Fan et al., 2004).

For each category, twenty five training sets with 400 positive points and 400 negatives are bootstrapped from the original training data. A C4.5 decision tree is then built based on each training set. In total, twenty-five decision trees are obtained for each category. These twenty-five trees are grouped together to create an ensemble for future predictions. This bagging-like approach is a standard way to solve such marketing problems and often very powerful. However, it does not work well on the marketing data set used in this study. For example, the lift curve for Category 10 is almost a 45 degree line, which implies that the ensemble learned hardly anything useful from the training sets. Our explanation is that the number of distinctive positive points in the training sets for Category 10 is too small. As shown in Table 4, the original response rate for category 10 is only 0.02%, so there is simply not enough information to build a decent classifier.

To improve the performance of the ensembles built by the original bagging approach, extra useful information is necessary. Sharing classifiers among different categories is our proposed solution. We cite two reasons to support this proposition. First, we are dealing with customers from the same catalog company. It is reasonable to expect that those customers who did place orders should share some common properties and those properties should be reflected in the classifiers belonging to those categories with relatively higher

response rate. If these classifiers can be included into the ensembles of those categories without enough positive points, they may help hit the targets and improve the overall performance. Second, the purchase patterns of some different categories may be similar. For example, people are more likely to buy clothes when there are discount coupons. This may also be true for shoes. Therefore, a marketing model for clothes that stresses the importance of discount coupons may also work for shoes although they belong to different product categories.

A naive way of sharing classifiers is to pool all the classifiers from the 19 categories into one big ensemble and use it for every category. However, there exist risks behind this sharing-all strategy. Specifically, when there are strikingly conflicting concepts among the problem domains, mixing classifiers blindly will degrade the effectiveness of the ensemble method. Therefore, it is safer to bring in the SDP-based screening process that is able to select a subset of classifiers for each problem domain. Unlike the experiments on the UCI data sets, here we use a held-out tuning set for each subset selection process since there is a large amount of data. Therefore, the P matrix in (1) is constructed based on classification results on the tuning set instead of the training set. Each tuning set reflects the original class distribution. Note that there are 19 categories in the marketing data, so there will be 19 tuning sets and the subset selection process will be repeated 19 times, once for each category.

There is still one more problem with the current setup of the $\tilde{G}$ matrix. Since the tuning data sets here reflect the original class distributions, which are highly biased towards non-buyers, the resulting $\tilde{G}$ matrix will reflect the performance of the ensemble on the negative points while almost ignoring the influence of the positive points. It is necessary to balance the influence of the positive points and the negative points on $\tilde{G}$. To achieve this goal, we define a third $\hat{G}$ matrix as a convex combination of the $\tilde{G}$ matrix on the positive points and the $\tilde{G}$ matrix on the negative points in the tuning set,

$$\hat{G} = \lambda \tilde{G}_{pos} + (1 - \lambda)\tilde{G}_{neg}.$$

Note that $\tilde{G}_{pos}$ is computed based only on the positive points in the tuning set and $\tilde{G}_{neg}$ only on the negative points, using formulas (1) and (2). In the following computational experiments, λ is set to 0.5.

The original data schema for each category is identical: same independent variables (purchase history, promotions and demographic information) and a binary dependent variable indicating whether the customer buys products from this category. However, we found that some of the independent variables are category-specific. For example, there are 19 variables each representing whether there is a discount for each category. Intuitively, the discount variable for Category i is more informative to the prediction problem for Category i than for other categories. There is likely a split on the discount variable for Category i in the decision trees for Category i. Since this discount variable is probably (not absolutely) not relevant to other categories, the decision trees induced on Category i are less likely to be useful for other categories. To make the decision trees more interchangeable among different categories, we did some transformations on those category-specific variables. In the data set for Category i, a copy of each category-specific variable related to Category i is appended to the end of the data schema and labeled as "xxxx-for-this-category". The values of the original category-specific variables for this Category i (which already have

Cat.	Non-sharing	Naive-Sharing	Selective-sharing
1	77.69(4.08)	82.93(1.21)	82.31(2.22)
2	76.89(2.03)	80.45(0.47)	77.71(5.81)
3	63.95(1.69)	84.79(1.15)	83.02(5.52)
4	82.38(1.13)	80.93(0.40)	77.94(3.15)
5	73.31(4.35)	82.06(0.95)	80.38(1.57)
6	80.45(0.60)	80.32(0.42)	78.20(2.58)
7	79.79(1.41)	81.20(0.65)	79.72(4.73)
8	75.96(0.43)	78.67(0.46)	76.71(1.51)
9	81.72(0.78)	81.33(0.58)	79.04(2.55)
10	52.67(9.71)	74.82(3.55)	72.33(6.29)
11	68.62(6.54)	80.46(1.58)	76.74(2.55)
12	57.11(5.56)	78.76(2.81)	75.42(3.56)
13	79.45(0.97)	78.56(0.52)	77.03(2.30)
14	83.50(0.26)	81.41(0.30)	81.25(2.74)
15	97.34(0.16)	93.05(1.58)	97.06(0.35)
16	97.63(0.07)	93.51(1.33)	97.66(0.13)
17	82.86(0.85)	84.82(0.61)	83.80(0.40)
18	85.60(1.05)	87.37(0.73)	88.01(0.35)
19	91.27(0.23)	89.69(0.82)	91.16(0.55)
		Comparison	**Selective vs. other**
	11-8-0	4-15-0	Abs.W-L-T
	3-0-16	3-1-15	Sig.W-L-T

Table 4: AUCs of non-sharing, naive-sharing and selective-sharing ensembles

copies) are then set to be uniform for each record so that there will be no splits on these variables in the decision trees. The splits, if necessary, will be made on those newly appended variables. After transformation, each category has the same number of appended category-specific variables related only to itself so the data schema of each category is still identical and the tree splits on the category-specific variables are more meaningful across categories. For instance, if there is a rule induced on a category like "if there is a discount on this category, the customer is going to buy products from this category", it is reasonably applicable to other categories. Therefore, the interchangeability of trees among categories is increased.

Since the original bagging approach builds ensembles of 25 classifiers, here we set the size of the selected ensemble to be 25 as well for a fair comparison. Another reason to set the size of the selected ensemble equal to that of the original ensemble is that for each category, there are at least 25 relevant classifiers: those that are trained on its own data. In fact, we have experimented with other sizes such as 50, but it didn't result in any significant difference for this particular problem.

Table 4 lists the AUCs (Area Under the ROC Curve) of three methods: original bagging (non-sharing), naive-sharing and selective-sharing, on each category. Summaries based on mean value and paired t tests (95% significance level) are attached at the bottom of the table.

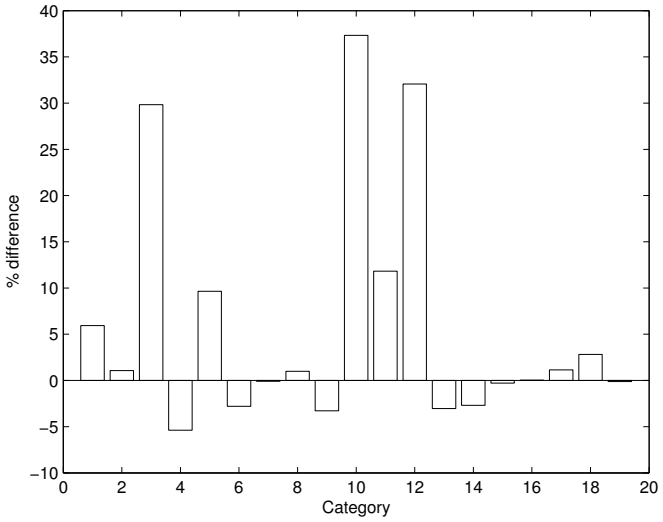


Figure 2: % Difference of AUCs of selective-sharing ensemble and the original bagging ensembles on 19 categories, $\frac{AUC_{sel}-AUC_{orig}}{AUC_{orig}}$

The results show that selective sharing of classifiers does improve the AUCs for almost half of the categories. Especially for those categories without enough positive information, the rise in AUC is substantial (Figure 2). The overall performance of the ensembles produced by selective-sharing is almost as good as the naive-sharing ensemble. For Categories 15, 16 and 19, it is even statistically better. Note that the selected ensembles use only 25 classifiers each as compared to 475 for the naive-sharing ensemble. There are two implications of this result. First, the ensemble pruning process is quite effective. Second, the reason that the including-all ensemble is better than the individual bagging ensembles is not simply because it's larger. There is truly useful additional information in the including-all ensemble that can be singled out by the pruning process.

Moreover, the selective sharing is a conservative version of sharing classifiers, hence it should be more robust. If there were wildly conflicting concepts among the categories, the selective-sharing would have performed better. In fact, the selective-sharing ensembles of Categories 15 and 16 do outperform the including-all ensemble by throwing away bad classifiers, as expected.

It is also interesting to look at the classifier sharing structure among different categories. Table 4 provides a list of statistics that summarizes the sharing structure, averaged over the same five runs in Table 4. The "Prior" column shows the response rate for each category. The "Used" column is the total number of classifiers trained on this category used for all categories. The "Used Own" column is the total number of classifiers trained on this category used for this category. The "Most Used" column shows the most common training category for the classifiers for each ensemble. Finally, HI index is the Herfindahl index

Category	**Prior** (%)	**Used**	**Used Own**	**Most Used**	**HI index**
1	0.14	2	1	13	0.16
2	0.27	0	0	14	0.27
3	0.04	0	0	6	0.16
4	0.92	12	1	17	0.25
5	0.12	1	0	17	0.27
6	0.83	17	0	14	0.35
7	0.44	9	0	14	0.35
8	0.37	14	1	14	0.67
9	0.64	22	3	14	0.29
10	0.02	1	0	19	0.33
11	0.13	3	0	19	0.57
12	0.05	2	0	19	0.47
13	0.44	18	0	19	0.67
14	2.65	80	7	14	0.30
15	1.70	11	0	19	0.33
16	2.09	14	1	19	0.35
17	0.65	94	9	19	0.34
18	0.31	24	2	19	0.49
19	1.17	150	18	19	0.63

Table 5: Statistics of classifier sharing structure

(Rose and Engel, 2002), which is computed by the following formula:

$$HI_i = \sum_{j=1}^{C} (\frac{n_{ij}}{N_e})^2$$

where n_{ij} is the number of classifiers trained on jth category used for ith category, and N_e is the size of the pruned ensemble which is 25 in this case. The smaller the HI index, the more diverse the ensemble, in terms of original training categories. From our observation, the classifier sharing structure is reasonably stable for most of the categories over the five runs. For instance, classifiers from Categories 14, 17 and 19 are always the top three used classifiers in the selected ensembles, the most used column seldom varies among different runs, etc.

There are a few things that one can tell after studying these numbers.

- The most surprising fact shown by the table is that for most categories, the classifiers trained on the category are seldom chosen for its own ensemble. Only Categories 14, 17 and 19 used a reasonable number of their own classifiers. However, the performance of the pruned ensembles are close to that of the original bagging ensembles as shown in Table 4. This may imply that some of the categories are closely related therefore their classifiers are highly interchangeable through combination.

- The higher the original response rate for each category, the more times its classifiers are used by other categories. Figure 3 plots the number of times its classifiers are used

versus the original response rate for each category. The slope of the linear regression is 24.00 with p value 0.053, which verifies the conjecture that the classifiers trained on categories with more positive information may well capture the general properties of the potential customers and are thus widely used by other categories. The fact that the p value is slightly below the 95% significance level implies that the response rate is not the sole factor that decides the popularity of the classifiers.

- Categories 15, 16 and 17 are the major outliers from the above rule. Categories 15 and 16 have relatively high response rate, but their classifiers are seldom used, even hardly used in their own ensembles. The including-all ensemble performs badly on these two categories. On the other hand, the pruned ensembles perform almost as well as the original bagging ensembles. Our explanation is that these two categories are a little different from the main trend. However, these two categories are somehow closely related to a subset of categories so that a combination of classifiers from those categories may still predict these two categories well. It is unclear why their own classifiers are so rarely used for their own categories. One guess is that the original bagging ensemble members for these two categories are not good individually, though they perform well as a group. So the pruning algorithm throws them away because they are not strong enough by themselves. Category 17 is an outlier from the other side. It has relatively low response rate, but its classifiers are the second most used. This indicates that despite the low response rate, the classifiers trained on Category 17 are still good or often provide information that is complementary to that provided from other categories. In addition, there is a subset of categories in which category 17 may be a good representative. This subset includes Categories 4 and 5 because these two categories used many classifiers from Category 17, as shown in the "Most Used" column of Table 4.

- Classifiers from categories with response rate less than 0.3% are hardly used. Usually, the including-all ensemble performs pretty well on those categories. It shows that for those problem domains without enough information, using aggregated information from all related problem domains may be a good solution.

- Classifiers from Categories 14 and 19 occupy almost half of the classifiers that are selected. Classifiers from Category 14 dominate pruned ensembles in six categories while classifiers from Category 19 dominate in nine categories. This might be because the customers of these two categories well-represent the potential buyers of the whole catalog. There exists a subtle difference between these two categories. For Category 14, although its classifiers are widely used, there are only 7 selected for its own problem, while for Category 19, almost all of its own classifiers are selected. There are two explanations of this phenomenon. First, classifiers of Category 14 captures well only part of the characteristics of its customers, therefore, it still needs extra information for its own problem. Second, those classifiers are good but very close to each other. So the divergence criteria of the subset selection process prevents including too many of its own classifiers.

- The Herfindahl index shows the homogeneity of the sources of the classifiers in the pruned ensembles. If a category prefers universal knowledge for its prediction, the

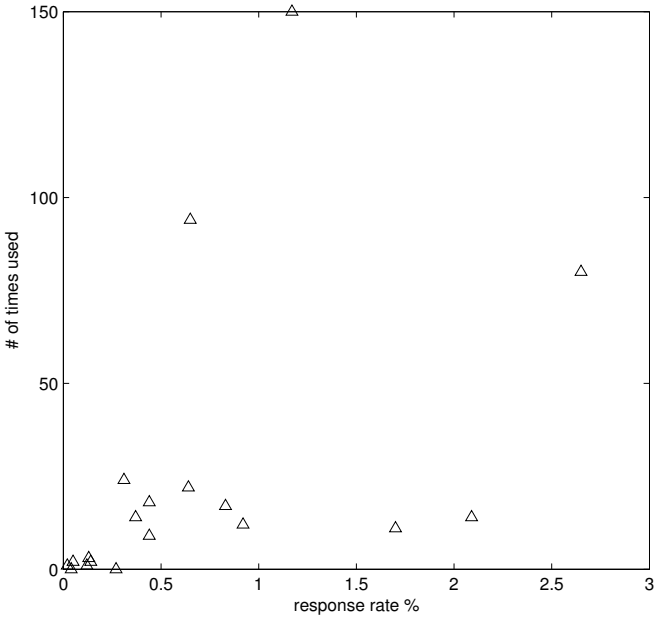


Figure 3: Response rate of each category vs. total number of times its classifiers are used in the selected ensembles

HI index will be low, which means that the pruned ensemble picks classifiers from many categories. On the other hand, if a category needs specialized knowledge for its prediction, the HI index will be higher, which implies that the pruned ensemble picks classifiers only from a small number of categories that share some particular properties with the category in question. Usually for those categories with small response rate, the HI index is low, for example Category 1 and Category 3. Categories 8 and 13 have the highest HI index. From the "Most Used" column, we know that Category 8 used classifiers mostly from Category 14. So it can be inferred that Category 8 and category 14 are closely related. For the same reason, the customers of Category 13 may share some special properties with that of Category 19.

These pieces of information might make more sense if we knew what these categories were. Unfortunately, this knowledge is currently unavailable. The information gained by studying the classifier sharing structure may help improve the mailing and promotion strategy of the catalog company. For instance, customers that buy products from Categories 14, 17 and 19 seem to capture the main trend of the customer base, so they are likely the core customers of the company and may need extra attention. Customers of Category 14 and 8 seem to be similar from some perspective. Therefore, a promotion strategy that proves to be successful for Category 14 may also work well for Category 8.

5. Conclusion and Future Work

This paper introduced a new ensemble pruning method to improve efficiency and effectiveness of an existing ensemble. Unlike previous heuristic approaches, we formulate the ensemble pruning problem as a strict mathematical programming problem and apply SDP relaxation techniques to obtain a good approximate solution. The computational experiments on the UCI repository data sets show that this SDP-based pruning algorithm performs better than two other metric-based pruning algorithms. Its application in a classifier sharing study also indicates that this subset selection procedure is effective in picking classifiers that fit the needs of different problem domains. Besides the peer-to-peer email filtering problem mentioned before, this method can also be useful when a company is trying to promote a new product. Usually, there is only limited information about a new product's potential customers. However, if the company has good marketing models for its old products, especially those closely related to the new product, selecting some of the old models based on the limited data of the new product may be a better solution than building models directly.

There is yet room for improvement in the current version of the algorithm. For example, there are several parameters in the model that can be fine-tuned, such as the method of normalization and the relative weight between the diagonal terms and off-diagonal terms. Another thing that's worth exploring is whether there exits a nice form of the objective function so that a "real" optimal subset can be found without enforcing the cardinality constraint. Under the current setup, removing the cardinality constraint will result in a trivial solution.

Acknowledgment

Samuel Burer would like to acknowledge support from NSF Grant CCR-0203426 and CCF-0545514.

References

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In *Proc. 17th International Conf. on Machine Learning*, pages 65–72. Morgan Kaufmann, San Francisco, CA, 2000.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Arcing classifiers. *Annals of Statistics*, 26:801–849, 1998.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (Series B)*, 95:329–357, 2003.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the 21st International Conference on Machine Learning*, pages 18–25, 2004.

P.K. Chan, W. Fan, A. Prodromidis, and S.J. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems Journal*, November/December:67–74, 1999.

N.V. Chawla, L.O. Hall, K.W. Bowyer, and W.P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, 5: 421–451, 2004. ISSN 1533-7928.

A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *Advances in Neural Information Processing Systems*, 17, 2004.

A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000. ISSN 0885-6125.

W. Fan, H. Wang, and P.S. Yu. Mining extremely skewed trading anomalies. In *Proceedings of the 9th International Conference on Extending Database Technology*, pages 801–810, 2004.

U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41:174–211, 2001.

J.L. Fleiss. *Statistical Methods for Rates and Proportions*. John Wiley & Sons, 1981.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996. URL `citeseer.nj.nec.com/freund96experiments.html`.

M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite prgramming. *Journal of ACM*, 42: 1115–1145, 1995.

A.J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *AAAI/IAAI*, pages 692–699, 1998.

Q. Han, Y. Ye, and J. Zhang. An improved rounding method and semidefinite programming relaxation for graph partition. *Mathematical Programming*, pages 509–535, 2002.

L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. ISSN 0162-8828.

S. Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1997. ISSN 0893-6080.

T.K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Y. Kim, N.W. Street, and F. Menczer. Meta-evolutionary ensembles. In *IEEE International Joint Conference on Neural Networks*, pages 2791–2796, 2002.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5: 27–72, 2004.

D.D. Margineantu and T.G. Dietterich. Pruning adaptive boosting. In *14th International Conference on Machine Learning*, pages 211–218, 1997.

L. Mason, P. Bartlett, and J. Baxter. Direct optimization of margins improves generalization in combined classifier. *Advances in Neural Information Processing Systems*, 11:288–294, 1999.

A. McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI'99 Workshop on Text Learning*, 1999.

D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, pages 169–198, 1999.

M.P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman-Hall, 1993.

A. Prodromidis and P. Chan. Meta-learning in distributed data mining systems: Issues and approaches. In *Advances of Distributed Data Mining*. AAAI press, 2000.

A. Prodromidis and S. Stolfo. Pruning meta-classifiers in a distributed data mining system. In *Proc. of the First National Conference on New Information Technologies*, pages 151–160, Athens, Greece, October 1998.

R.J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Manteo, CA, 1993.

A.K. Rose and C. Engel. Currency unions and international integration. *Journal of Money, Credit and Banking*, 34(4):1067–89, 2002.

SDPLR, 2002. See the website: `http://dollar.biz.uiowa.edu/~sburer/software/SDPLR/`.

X. Shen, M. Boutell, J. Luo, and C. Brown. Multi-label machine learning and its application to semantic scene classification. In *International Symposium on Electronic Imaging*, San Jose, CA, January 2004.

W.N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 377–382, 2001.

H. Wang, W. Fan, P. Yun, and J. Han. Mining concept-drifting data streams using ensemble classifiers, 2003.

D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

G.U. Yule. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London, Ser. A*, 194:257–319, 1900.

Z. Zhou, J. Wu, Y. Jiang, and S. Chen. Genetic algorithm based selective neural network ensemble. In *17th International Joint Conference on Artificial Intelligence*, pages 797–802, 2001.

Journal of Machine Learning Research 7 (2006) 1339–1355 Submitted 9/05; Revised 2/06; Published 7/06

Linear Programs for Hypotheses Selection in Probabilistic Inference Models

Anders Bergkvist ABK@MOLBIO.GU.SE
Department of Molecular Biology
Göteborg University
P.O. Box 462
40530 Göteborg, Sweden

Peter Damaschke PTR@CS.CHALMERS.SE
Department of Computer Science and Engineering
Chalmers University
41296 Göteborg, Sweden

Marcel Lüthi MARCEL.LUETHI@STUD.UNIBAS.CH
Department of Computer Science
University of Basel
4056 Basel, Switzerland

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

We consider an optimization problem in probabilistic inference: Given n hypotheses H_j, m possible observations O_k, their conditional probabilities p_{kj}, and a particular O_k, select a possibly small subset of hypotheses excluding the true target only with some error probability ε. After specifying the optimization goal we show that this problem can be solved through a linear program in mn variables that indicate the probabilities to discard a hypothesis given an observation. Moreover, we can compute optimal strategies where only $O(m+n)$ of these variables get fractional values. The manageable size of the linear programs and the mostly deterministic shape of optimal strategies makes the method practicable. We interpret the dual variables as worst-case distributions of hypotheses, and we point out some counterintuitive nonmonotonic behaviour of the variables as a function of the error bound ε. One of the open problems is the existence of a purely combinatorial algorithm that is faster than generic linear programming.

Keywords: probabilistic inference, error probability, linear programming, cycle-free graphs, network flows

1. Introduction

Suppose that we are given one of m possible *observations* O_k, $k = 1,\ldots,m$, and n *hypotheses* H_j, $j = 1,\ldots,n$, each of which might have caused the observed O_k. Moreover we know the conditional probabilities $p_{kj} = P(O_k|H_j)$ to observe O_k if H_j is the true hypothesis, also called the *target*. Since exactly one O_k is observed, the p_{kj} must satisfy $\sum_{k=1}^{m} p_{kj} = 1$ for every j. The p_{kj} may come from background knowledge of causal relations, or they may be estimated from statistical data.

Our aim is to devise a strategy that, for any observed O_k, selects a subset of hypotheses so as to minimize two conflicting parameters at the same time: the probability to discard (that is, not to select) the target, and the size of the selection. We imagine that the selected hypotheses are

then examined closer, in order to identify the target, whereas we would come back to discarded hypotheses only if we missed the target in our selection. In Section 2 we will state this problem formally as an optimization problem, namely, the minimization of the expected weight of excluded hypotheses, given an error probability bound for each target. We think that the problem is very fundamental and its optimization view could be interesting for any setting where one has to guess hypotheses from data with known conditional distributions.

An obvious application scenario is diagnosis. The probabilities of various syndromes caused by any disease may be known from a database. In each particular case with a given syndrome, one wants to narrow down the set of suspects, that is, of possible diseases to be examined more carefully. But the true hypothesis should, with high probability, not be discarded in the beginning. See the discussion by Szolovits et al. (1988) which refers, however, to complex and structured models rather than "atomic" hypotheses and data.

Our particular motivation however came from a protein structure prediction project. Proteins are sequences of residues, each residue being derived from one of 20 possible amino acids. The 3D structure of the protein backbone is uniquely determined by its torsion angles. Since it is difficult and costly to determine them experimentally, various methods have been developed to infer torsion angles and other structure elements from easier measurable, correlated data, partly with help of sequence homology. Nuclear magnetic resonance (NMR) chemical shifts of nuclei in the amino acids are certain spectroscopic data influenced by the local molecular conformation, see Beger and Bolton (1997); Cornilescu et al. (1999); Wang and Jardetzky (2002); Xu and Case (2002) for more background information. Due to the correlations, it is a natural idea to infer torsion angles from measured chemical shifts. Torsion angle restraints that are narrow but still contain the (unknown) true torsion angle values in the majority of cases are important for correct 3D structure reconstruction of whole protein sequences. Since the correlations are complicated and can hardly be put in a neat formula, we have chosen a statistical approach based on large samples of data. That is, the "local" task of predicting single torsion angle restraints leads to instances of the optimization problem as considered here: Our hypotheses are torsion angles, our observations are measured chemical shifts, both discretized in finitely many intervals, and the p_{kj} are estimated from a database. Our raw data are scatterplots of chemical shift vs. torsion angle values from public databases. The discretization is done in a preprocessing phase, with the aim to partition the scatterplot into a coarse grid where the data points in each rectangle are approximately evenly distributed (so that further splitting would be meaningless). The current partitioning heuristic is described by Christin (2006). Then, we apply different prediction heuristics to the discretized scatterplots, that is, point count matrices. The role of optimization in this application is discussed after the main part of the paper, in Section 7. We have to treat in a semi-automated way a huge number of problem instances: for 6 different nuclei, 20 different amino acids, and 2 torsion angles we get nearly 240 data sets. (A few are empty.) Actually, the number of instances is a multiple of this number when we consider several error probability bounds and their combinations, maybe several discretizations, different sources of raw data, etc. In this sense our application is large-scale, even though the single problem instances are not. On the contrary, we need to reduce every instance to some efficiently solvable optimization problem in order to keep the project feasible. In this paper we will show several beneficial properties of the optimization problem that comply with this goal:

- We end up with a linear program in mn variables, which is a manageable size (see Section 2). Note that, since a selection rule conditional on the observation can be randomized, the possi-

ble strategies are described in the first place by variables for the probabilities of all 2^n subsets of hypotheses. However, by the nature of our objective function and by linearity of expectation, we actually need only variables for the probability to exclude any single hypothesis under any observation.

- We can always find an optimal solution where at most $\min\{m,n\}+n$ of the mn variables are strictly between 0 and 1 (Section 4). We conjecture that the actual number of fractional variables is even somewhat smaller. Hence, most decisions are deterministic, which greatly simplifies the practical use of our approach.

- The linear program formulation is quite flexible. We can work, for example, with larger error probabilities for hypotheses that are unlikely to appear as target, or hard to discriminate from others. We will also assign a weight w_j to every hypothesis H_j. Our goal is then to minimize the total weight of selected hypotheses, under given error probability constraints. The weights are just coefficients that do not complicate the problem to solve, but give us further modelling options (see below).

- It is easy to combine predictions from several unrelated observations, if their conditional distributions are available for the considered set of hypotheses. This can further reduce the selected set of hypotheses, for prescribed bounds on the error probability. Since most variables in optimal strategies are 0 or 1, the necessary calculations are fast (Section 6).

- Since the predictors are just linear programs, it is straightforward to implement the approach using standard software packages.

Regarding the weights, in the simplest case all w_j are equal. Otherwise, weight w_j may be used, for example, to indicate the time needed to verify or falsify H_j, so that the total weight of the selected set corresponds to the time to actually identify the target. In the diagnosis example, weights may also be proportional to time or costs to check the hypotheses, however we may divide each by a factor for the seriousness of the disease. In interval prediction applications like protein torsion angle prediction, it is sensible to choose the weight of each interval proportional to the interval length.

In Section 3 we also connect a game-theoretic interpretation of the problem to some Lagrange dual giving the worst-case probability distribution of hypotheses (in the sense that the achievable exlusiveness is minimized). We discuss the use of the dual optimum. Moreover we disprove in Section 4 the tempting conjecture that the exclusion probabilities in optimal strategies are always monotone in the error bounds. Such counterintuitive behaviour suggests that our optimization problem does not exhibit a simple structure that would also allow a simpler algorithm. Despite the fact that linear programs are a standard task being well solvable in practice, it would be interesting to devise a faster, purely combinatorial algorithm for our special class. This would speed up applications with massive sets of instances. We must leave this as an open question. Hints may come from some relationship to flow problems in lossy networks (briefly discussed in Section 5) for which such algorithms exist.

We are not aware of earlier work where optimization has been used for inference in such frameworks. A number of other machine learning tasks, for example, in classification and neural network training, have been cast as linear programs, as in Bennett (1992); Bennett and Mangasarian (1992a,b, 1993); Bradley (1998); Glover (1990). Damaschke (2004) studied target search problems

in finite probabilistic inference models with the goal to minimize the expected search time, when switching between the hypotheses (preemptive scheduling of verification jobs) is possible.

One may compare our optimization to very common and simple heuristic inference rules such as the maximum likelihood (ML) and the maximum a-posteriori (MAP) rule: For an observed O_k, ML selects the desired number of hypotheses H_j with the highest p_{kj}. MAP proceeds similarly with the posterior probabilities of the H_j, for prior probabilities given along with the p_{kj}, whereas ML ignores prior probabilities. Both ML and MAP can easily exclude some potential targets H_j completely even though they appear considerably often. This happens if p_{kj} is not among the top values for any k. In our approach we explicitly take care of the probabilities to wrongly discard the target (below denoted ε_j). Similarly to ML we do not make explicit use of prior probabilities, but we can, for example, assign higher ε_j to rare H_j. Finally we optimize the specificity of our hypotheses selection for the desired prescribed error bounds.

2. Hypothesis Selection by Linear Programs

Now we treat our problem more formally. Recall that p_{kj} is the (known) probability to observe O_k if H_j is the target. Based on an observed O_k, a player wants to discard a set of hypotheses that should have large weight but should not contain the target. A *strategy* σ is completely characterized by a probability distribution on the set of subsets (power set) of $\{H_1,\ldots,H_n\}$, depending on O_k. It specifies the probability to discard any set. This is in fact the most general form of a strategy, since the selection can be randomized, and the player does not learn more than just O_k. Next, we also make our optimization goal explicit.

Definition 1 *Consider a fixed strategy σ. The* error probability *of σ for target H_j is the probability to discard the target. The* exclusiveness *of σ for any fixed target H_j is the expected total weight of the hypotheses discarded by σ. (Here, randomness comes from the choice of O_k according to the p_{kj} and from σ's randomized choices.) Finally, the exclusiveness of σ is defined to be the worst (smallest) exclusiveness for all H_j.*

HYPOTHESIS SELECTION WITH ERROR BOUNDS *is the following problem: Given an $m \times n$ matrix $P = (p_{kj})$ and error probabilities ε_j for all H_j, devise a strategy with maximum exclusiveness.*

Comments:

(1) By defining the exclusiveness as the minimum over all hypotheses we optimize the guaranteed exclusiveness (in the sense of an expectation in the long run), independently of the frequencies of hypotheses which may be unknown or subject to changes: In the diagnosis example, the relative frequencies of diseases can vary a lot in time, and in torsion angle prediction, the distribution of angles in a protein under consideration is not known in advance. We avoid the explicit use of questionable prior probabilities.

(2) In the simplest case, all ε_j may be equal to some global error probability ε. However, we also allow individual error probabilities. This will not make our problem more complicated, but it gives us the option to assign higher error probabilities to certain hypotheses, and thus to raise the exclusiveness. The choice of the ε_j is up to the application, but, generally speaking, higher ε_j are advisable if H_j is considered unlikely, or if the vector of the p_{kj} for H_j (jth column of P) is in the convex hull of other columns of P, so that none of the O_k is characteristic for H_j alone.

(3) For entries with $p_{kj} = 0$ we would immediately discard hypothesis H_j upon observation O_k. Alternatively we may forbid zero entries and consider only instances with positive conditional

probabilities. In applications, typically the p_{kj} are estimated from statistical data, and instead of setting $p_{kj} = 0$ in the absence of cases, it is common in statistical learning methods to apply some correction rules that yield small positive values.

Note that a strategy is described by as many as $m2^n$ variables. However, for maximizing exclusiveness we actually need only mn variables, and this makes the approach feasible. Namely, let x_{kj} be the probability that σ discards hypothesis H_j if O_k has been observed. Let X be the $m \times n$ matrix $X = (x_{kj})$. Matrix X is well-defined, and X is uniquely determined by σ. (The converse is not true: The same X can be "realized" by many different σ, we come back to this point later.)

Theorem 2 *Matrix X of an optimal strategy for* HYPOTHESIS SELECTION WITH ERROR BOUNDS *is the solution to the linear program written below.*

$$\max u \tag{1}$$

$$\forall j : \sum_{k=1}^{m} p_{kj} x_{kj} \leq \varepsilon_j \tag{2}$$

$$\forall j : \sum_{k=1}^{m} p_{kj} \sum_{i=1}^{n} w_i x_{ki} \geq u \tag{3}$$

$$\forall k, j : 0 \leq x_{kj} \leq 1 \tag{4}$$

Proof The left-hand side of (2) is obviously the probability to discard H_j if H_j is the target. The left-hand side of (3) is the exclusiveness for H_j, hence (3) says that the exclusiveness for every H_j is at least some u that is maximized in (1). That is, we are maximizing the exclusiveness of the strategy as desired. Constraint (4) just ensures that the x_{kj} are probabilities. ■

Corollary 3 *We can compute an optimal strategy σ for* HYPOTHESIS SELECTION WITH ERROR BOUNDS *through a linear program in only mn variables.*

In particular, it follows that the problem has polynomial time complexity in n, m. We remark that, because of (3), the exclusiveness actually depends only on the weighted sum of variables in each row of X, defined by $x_k := \sum_{i=1}^{n} w_i x_{ki}$. Corollary 3 needs some discussion. Strategy σ is not uniquely determined by X, but it is easy to obtain some σ. To mention only two natural options: We may take a random number *rand* uniformly from interval $[0,1]$ and discard all H_j with $x_{kj} \geq rand$, or we may discard the H_j independently with probabilities x_{kj}. This arbitrariness is not an issue here. Firstly, all σ with the same X have also the same exclusiveness. Thus we will henceforth consider the exclusion probabilities x_{kj} as the strategy variables. Accordingly, we also call a matrix X a *strategy*. Secondly, we will show later that there always exist optimal strategies where only a limited number of variables in X is fractional, so that most decisions are in fact deterministic.

Some applications may prefer hypotheses of some guaranteed weight for every O_k (although this can be rather unnatural, especially when rows of P contain very different numbers of safely discarded small entries p_{kj}). Then, a similar linear program where constraint (3) is replaced with $\forall j : \sum_{i=1}^{n} w_i x_{ki} \geq u$ can be applied.

3. Game-theoretic Interpretation, Knapsack Strategies, and the Dual

Our linear program from Theorem 2 is equivalent to a matrix game between a player who selects hypotheses and an adversary ("Nature") which tries to make a successful choice as difficult as possible. More precisely, the player can choose a strategy X that respects (2) and (4), the adversary chooses a hypothesis, and the payoff to the player is exclusiveness u in (3). The set of possible strategies X is infinite, but we can turn the game into an equivalent finite game, by observing that (a) exclusiveness is linear in X, and (b) all feasible X build a polytope with finitely many vertices. Hence it suffices to consider only these vertices as the player's pure strategies, all other X are convex linear combinations of them. Claim (a) is obvious from the left-hand side of (3), and (b) is clear since the X form a (bounded) feasible region of a linear program. The adversary's mixed strategies can be interpreted as prior probabilities q_j of the H_j. In the following, $q = (q_1, \ldots, q_n)$ denotes a vector of prior probabilities. By von Neumann's minmax theorem, there exists a pair X^*, q^* of optimal mixed strategies for both opponents, and the expected payoff for X^*, q^* is the value of the game.

For the moment assume that the player knows $q = (q_1, \ldots, q_n)$. The optimal solutions against q are easy to characterize by means of the following definitions. In every column j of X we set up an instance of the fractional knapsack problem, with capacity ε_j and items $k = 1, \ldots, m$ having sizes p_{kj} and utilities $w_j \sum_{i=1}^{n} p_{ki} q_i$; see Martello and Toth (1990) for an introduction to knapsack problems. Note that the fractional knapsack problem is trivially solved by a greedy algorithm: Start from $x_{kj} := 0$ for all k, and then set $x_{kj} := 1$ for k with decreasing utility-to-size ratio $r_{kj} := w_j \sum_{i=1}^{n} p_{ki} q_i / p_{kj}$, until the capacity is exhausted. The last $x_{kj} > 0$ can be fractional. (Possible division by 0 does not cause problems, cf. Comment (3) below Definition 1. If $p_{kj} = 0$, we get $r_{kj} = \infty$ and also $x_{kj} = 1$. If the whole kth row of P is zero, we can even ignore it right from the beginning.)

Now, we call a matrix X a *knapsack strategy against prior q* if each column of X is an optimal solution to the fractional knapsack problem introduced above.

Proposition 4 *The optimal strategies X against a prior q are exactly the knapsack strategies against that prior q. In particular, if X^* is optimal then X^* is a knapsack strategy against every optimal q^*.*

Proof The first assertion is obvious, since the utility term $w_j \sum_{i=1}^{n} p_{ki} q_i$ is the coefficient of x_{kj} in the exclusiveness. Let q^* be any optimal strategy of the adversary. A player's strategy achieving the value of the game must be optimal under prior q^*. But since the latter strategies are knapsack strategies against q^*, the second assertion follows. ■

We remark that the converse cannot be concluded: A knapsack strategy against the optimal q^* is not necessarily optimal in the whole game, since it may be worse against other priors. Optimality requires an additional condition that we can get from duality theory of linear programs. The fact that a worst-case prior q^* corresponds to a certain Lagrangian dual might be an interesting structural property in itself:

Proposition 5 *When we dualize constraints (3), then the vector of the n Lagrange multipliers $\lambda_j \geq 0$ in the dual optimal solution is a worst-case prior q^*.*

Proof The Lagrange function is given by

$$L(X,u,\lambda) = u + \sum_{j=1}^{n} \lambda_j \left(\sum_{k=1}^{m} p_{kj} \sum_{i=1}^{n} w_i x_{ki} - u \right).$$

For any fixed vector $\lambda = (\lambda_1, \ldots, \lambda_n)$, the Lagrangian subproblem $\theta(\lambda) = \max_{X,u} L(X,u,\lambda)$ can be separated for u and X:

$$\theta(\lambda) = \max_{u} u(1 - \sum_{j=1}^{n} \lambda_j) + \max_{X} \sum_{j=1}^{n} \lambda_j \sum_{k=1}^{m} p_{kj} \sum_{i=1}^{n} w_i x_{ki}.$$

The Lagrangian dual is $\min_\lambda \theta(\lambda)$. We observe that $\sum_{j=1}^{n} \lambda_j \geq 1$, otherwise $\theta(\lambda)$ is unbounded. Since the X term is increasing in the λ_j, and the same matrices X give the maximum when vector λ is multiplied with any positive factor, $\theta(\lambda)$ attains its minimum for some λ with $\sum_{j=1}^{n} \lambda_j = 1$. Thus the Lagrangian dual simplifies to

$$\min_{\lambda} \theta(\lambda) = \min_{\lambda} \max_{X} \sum_{j=1}^{n} \lambda_j \sum_{k=1}^{m} p_{kj} \sum_{i=1}^{n} w_i x_{ki}$$

subject to $\sum_{j=1}^{n} \lambda_j = 1$ and the original constraints (2),(4). Note also that $\theta(\lambda)$ is precisely the exclusiveness for prior λ, thus $\lambda = q^*$. ■

Theorem 6 *(X,q) is a pair of optimal solutions if and only if: X is a knapsack strategy against q, and X has its lowest exclusiveness for all H_j where $q_j > 0$.*

Proof As we have dualized constraints (3), we get from the complementary slackness conditions that (X,q) is optimal if and only if X has optimal exclusiveness against q, and the following alternative holds true for every j: Variable q_j is zero, or the slackness in constraint (3) is zero, which means that X's exclusiveness for target H_j is exactly u (and not larger). Together with Proposition 4 the criterion follows. ■

Note that this optimality criterion can be checked in $O(mn)$ time for given X and q: One just has to solve the fractional knapsack instances for all columns j and to compare the left-hand sides of constraints (3). Since optimality is that easy to check, and the Lagrangian subproblem (fractional knapsack) is trivial, a gradient descent method for the Lagrangian dual is efficient in every step. Therefore it would be interesting to study whether some gradient descent heuristic approaches q^* already in a few iterations. This would be valuable for applications with many instances like our torsion angle prediction project.

Calculating q^* appears to be useful also in another respect: Although we did not explicitly use prior probabilities of the H_j, we know in general which H_j appear frequently or rarely. Now, if q_j^* is large for some rare hypothesis H_j, this indicates that X^* has been optimized for an unlikely distribution of targets. (Recall that X^* has the worst exclusiveness even for all H_j with positive q_j^*.) We may then drop constraint (3) for such indices j and optimize again, in order to raise the exclusiveness for the more frequent targets only. Such modifications are natural and easy to implement, and they may improve the global results in, for example, protein structure prediction. This has to be tested more extensively within the particular applications.

4. Structural Properties of Optimal Solutions

In the following we consider, for simplicity, a special case of our linear program from Theorem 2 where all ε_j are equal to some ε. Regarding the dependency of u from this parameter we have:

Proposition 7 *For any fixed likelihood matrix P, the optimal exclusiveness u is a monotone increasing and concave function in ε.*

Proof *Monotonicity:* Parameter ε appears only in constraints (2). If one raises ε then, obviously, the set of feasible solutions becomes only larger, and since we have a maximization problem, the optimal u increases.

Concavity: Consider the $(mn+2)$-dimensional space with the mn variables x_{kj} and, additionally, ε and u as coordinates. Let F be the feasible region of our linear program in this space, that is, the set of $(mn+2)$-vectors that fulfill constraints (2),(3),(4). Clearly, F is convex. Hence the projection $F|_{\varepsilon,u}$ of F to the ε vs. u plane is convex, too. ($F|_{\varepsilon,u}$ is the set of all pairs (ε,u) for which there exist values of the x_{kj} so that the constraints are satisfied.) Remember that we have to maximize u for a given ε. Geometrically this means to take the point at the upper boundary of $F|_{\varepsilon,u}$ at abscissa ε. Since $F|_{\varepsilon,u}$ is convex, the upper boundary is the graph of a (piecewise linear) concave function. (Figure 1.) ■

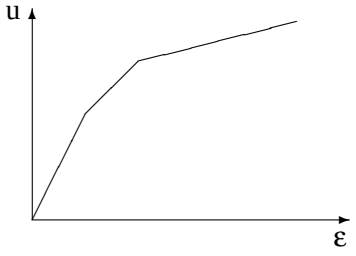


Figure 1. u is monotone and concave in ε. The graph limits the the feasible region from above.

One might expect that also every single variable x_{kj} in the strategy matrix X is monotone in the error bound ε, but this is not true in general. A small example demonstrates the reason. Recall the notations p_{kj} for the probability to observe O_k given H_j, the weighted row sums $x_k := \sum_{i=1}^{n} w_i x_{ki}$, and the utility-to-size ratios $r_{kj} := w_j \sum_{i=1}^{n} p_{ki} q_i / p_{kj}$ from the fractional knapsack problems.

Example 1 *Suppose that all hypotheses have unit weights $w_j = 1$. Consider the following matrix P of conditional probabilities p_{kj}:*

$$P = \begin{bmatrix} 0.1 & 0.5 & 0.8 \\ 0.9 & 0.5 & 0.2 \end{bmatrix}.$$

First let $\varepsilon = 0.1$. For the prior $(q_1,q_2,q_3) = (1,0,0)$ it is easy to check that any knapsack solution has exclusiveness $u = 0.73$. Moreover, against this prior, every knapsack solution with $x_1 \geq x_2$

satisfies the criterion in Theorem 6. Hence

$$X^* = \begin{bmatrix} 0.73+x & 0 & 0 \\ 0.03-x & 0.2 & 0.5 \end{bmatrix}$$

with $0 \leq x \leq 0.03$ *(arbitrary) is optimal, and* $(1,0,0)$ *is a worst prior. This in turn implies that every optimal X must be a knapsack solution against* $(1,0,0)$*. In particular,* $x_{22} = 0.2$ *is enforced.*

Now let $\varepsilon = 0.2$ *instead. Then, every knapsack solution against* $(1,0,0)$ *has* $x_1 < x_2$*, so that prior* $(0,0,1)$ *would be worse. But, similarly, every knapsack solution against* $(0,0,1)$ *has* $x_1 > x_2$*, so that prior* $(1,0,0)$ *would be worse. It follows that* $x_1 = x_2$ *holds in every optimal X, and that an optimal* q^* *differs from these two priors. But each prior except the mentioned two gives* $r_{11} > r_{21}$ *and* $r_{23} > r_{13}$*, which determines column 1 and 3 of* X^* *uniquely. Together with* $x_1 = x_2$ *this finally yields (matrix entries rounded to three decimals):*

$$X^* = \begin{bmatrix} 1 & 0.256 & 0 \\ 0.111 & 0.144 & 1 \end{bmatrix}.$$

Note that x_{22} *is smaller than before! The explanation is that* x_{11} *reached 1, thus only* x_{21} *could increase, and* x_{22} *decreased in favour of* x_{12}*, in order to keep* x_1 *and* x_2 *balanced.*

Next we consider arbitrary individual error bounds ε_j again. As announced, we show that our linear programs from Theorem 2 have optimal solutions where only a minority of the *mn* variables x_{kj} is fractional, that is, properly between 0 and 1. It means that these selection strategies are to a large extent deterministic, which makes them much easier to handle in practice.

Theorem 8 *Any optimal solution being a vertex of the feasible region has at most* $2n$ *fractional variables.*

Proof Some optimal solution X of a linear program is always a vertex of the feasible region. Constraints (4) describe the hypercube in *mn*-dimensional space where all vertices have coordinates 0 or 1. Furthermore, the number of binding constraints in a vertex X is at least the dimension *mn*, but only one of any two constraints $x_{kj} \geq 0$, $x_{kj} \leq 1$ can be binding. Thus, in a vertex X with more than $2n$ fractional coordinates, more than $2n$ other constraints must be binding. Since we have only $2n$ constraints (2),(3), the assertion follows. ■

We can also say something about the *positions* of fractional entries in optimal strategy matrices X and get a better bound in case that $m \leq n$. Let $B(X)$ be the bipartite graph with vertices r_k for all rows k, and vertices c_j for all columns j, where an edge between r_k and c_j exists iff x_{kj} is a fractional value.

Theorem 9 *There exists an optimal solution X with cycle-free* $B(X)$*, and thus at most* $m+n-1$ *fractional entries.*

Proof Suppose that $B(X)$ contains a cycle C with vertices $r_1, c_1, r_2, c_2, \ldots, r_l, c_l$ (in this cyclic order). That is, edges $r_i c_i$, $c_i r_{i+1}$ and $c_l r_1$ exist, where $2l$ is the length of the cycle. Note that the indexing of

rows and columns in X is arbitrary, hence we may rename them such that, without loss of generality, indices in C are $1,2,\ldots,l$ as defined above.

Let d be some real number that we fix later. We change the matrix entries corresponding to the edges in C by the following procedure. First, define $d_{11} = d$ and replace x_{11} with $x_{11} - d_{11}$. Define $d_{21} = \frac{p_{11}}{p_{21}} d_{11}$ and replace x_{21} with $x_{21} + d_{21}$. Obviously, the error bound constraint (2) remains valid for column $j = 1$. Next, define $d_{22} = \frac{w_1}{w_2} d_{21}$ and replace x_{22} with $x_{22} - d_{22}$. The effect is that $x_k := \sum_{i=1}^{n} w_i x_{ki}$ remains unchanged for row $k = 2$. We walk the cycle C and continue in this way. The general step is: Define $d_{ii} = \frac{w_{i-1}}{w_i} d_{i,i-1}$ and replace x_{ii} with $x_{ii} - d_{ii}$, then define $d_{i+1,i} = \frac{p_{ii}}{p_{i+1,1}} d_{ii}$ and replace $x_{i+1,i}$ with $x_{i+1,i} + d_{i+1,1}$. Following this scheme we finally we update x_{1l}, according to $l + 1 \bmod l = 1$.

Note that all these changes neither affect the left-hand sides of constraints (2) nor the weighted row sums x_k defined above, with x_1 as the only exception. If x_1 has not decreased, constraints (3) remain satisfied, too. If x_1 has decreased, we use $-d$ instead of d, so that x_1 now increases. Since all x_{kj} on edges of C are fractional, constraints (4) also remain valid for small enough $|d|$. Hence we get a new feasible solution for any d which has the suitable sign and small enough absolute value. Finally we adjust our d so that some entry in C becomes exactly 0 or 1.

Hence we can destroy some cycle C of fractional entries. Since the optimal value u is monotone in the x_k, the new solution X is no worse. Applying the same procedure repeatedly we destroy all such cycles. Since every step also properly decreases the number of fractional entries, the process terminates with an X as desired. Since a cycle-free graph has fewer edges than vertices, the bound $m + n - 1$ follows.

We remark that this proof gives also a polynomial algorithm that computes a cycle-free optimal solution. ■

The 3×2 instances from Example 1 admit optimal solutions with $n = 3$ fractional entries.

An obvious question is whether our combinatorial bounds are already tight. More precisely: Given numbers m, n, let $f(m,n)$ denote the largest number such that there exists an $m \times n$ instance P of HYPOTHESIS SELECTION WITH ERROR BOUNDS where every optimal solution X needs at least $f(m,n)$ fractional variables. We have shown $f(m,n) \leq \min\{m,n\} + n$, and it is trivial to give general examples where the number of fractional variables must be n, so that $f(m,n) \geq n$. On the other hand, note that the "fractional knapsack" property of optimal solutions does not imply $f(m,n) \leq n$: Knapsack solutions are not always unique and may allow several fractional variables in a column j (namely if several r_{kj} are equal), and since a knapsack solution against a dual optimal q^* is not necessarily already optimal, we may have to take a solution with more fractional variables. We must leave the exact $f(m,n)$ as an open problem.

5. Is There a Faster Algorithm?

In this more informal section we briefly discuss another open problem: to devise a purely combinatorial algorithm for our class of linear programs that is faster than a generic linear program solver. We point out two ways, but also the reasons why these attempts have not been successful so far.

(1) Example 1 in the previous section shows (besides non-monotonicity of the x_{kj} in the error bounds) that, in an optimal solution, the x_{kj} in a row k are in general not simply filled up to 1 in increasing order of the p_{kj}. This is an effect of the columnwise error constraints. Nevertheless, intuition tells that larger x_{kj} are mostly assigned to smaller p_{kj}. Exceptions are structurally limited,

due to the following discussion. Let us call two matrix entries in the same row or column a *monotone pair* if the values of these entries in X and P stand in the *same* relation (larger or smaller). For an input P and a strategy X, define a directed graph $C(X)$ whose vertices are the columns, with a directed arc from i to j if $p_{ki} > p_{kj}$ and $x_{ki} > x_{kj}$ holds for some k. The directed graph $R(X)$ whose vertices are rows is defined similarly. By an argument similar to the proof of Theorem 9 we can show the existence of an optimal solution X where $B(X)$ is cycle-free and also $C(X)$ and $R(X)$ are free of *directed* cycles. Hence there is some topological order of the rows and columns such that all monotone pairs in rows and columns decrease in the same direction, for example, to the right and downwards, respectively. However this does not limit the *number* of monotone pairs. Moreover, the topological orders are not obvious from P, and even if we knew them, we could not compute the optimal X from them in a simple way. In summary, the observation above did not lead us to an efficient algorithm.

(2) Another idea is a reduction to flow problems in bipartite lossy networks. For that problem which has many other applications in transportation and finance (for example, currency exchange), purely combinatorial polynomial-time algorithms have been given by Tardos and Wayne (1998); Wayne (2002). However, the idea works only for a variant of HYPOTHESIS SELECTION WITH ERROR BOUNDS with "observation-wise" exclusiveness demands instead of a global exclusiveness objective: Recall again the weighted row sums $x_k := \sum_{i=1}^{n} w_i x_{ki}$. For given parameters ε_j and y_k for all j and k, respectively, we may raise the following existence problem: Is there a solution X with error probabilities at most ε_j for all H_j, and $x_k \geq y_k$ for all O_k? This problem is easily seen to be a flow problem in a bipartite lossy network with arc capacities 1 and gain factors $1/p_{kj}$; see Tardos and Wayne (1998); Wayne (2002) for the definitions. In contrast, a reduction from HYPOTHESIS SELECTION WITH ERROR BOUNDS does not seem to exist, for the intuitive reason that flow variables cannot be "copied" in order to "participate" in several linear combinations of the x_k. Still, algorithmic techniques similar to those used for flows in lossy networks might be applicable. We have to leave this subject for future research.

6. Combining Data

Suppose that we have several matrices $P^{(1)}, P^{(2)}, \ldots$ of conditional probabilities for the same set of hypotheses but for different types of observations, such as different groups of symptomes in diagnosis, or chemical shifts of several nuclei in protein torsion angle prediction. We do not assume that the joint distribution of vectors of observations is known: Since the number of vectors is the product of $m^{(1)}, m^{(2)}, \ldots$, there may be not enough cases in the database that would allow meaningful probability estimates for all these vectors. Still, combining these data sets can further narrow down the selected hypotheses (if the observations "complement each other" well), and at the same time preserve guaranteed error bounds. For ease of presentation we describe the method for two matrices, but it can be readily extended to any number.

Proposition 10 *Let P and P' be the conditional probability matrices of size $m \times n$ and $m' \times n$, respectively, and $\varepsilon_j, \varepsilon'_j$ the error bounds of two instances of* HYPOTHESIS SELECTION WITH ERROR BOUNDS *for the same set of hypotheses H_j, $j = 1, \ldots, n$. Furthermore let X and X' be strategies for these two instances that respect the given error bounds. For any pair of observations O_k, O'_l from both instances (where $k = 1, \ldots, m$ and $l = 1, \ldots, m'$), we define for all H_j the exclusion probabilities $x_{kj} + x'_{lj} - x_{kj} x'_{lj}$. Then the resulting $mm' \times n$ matrix is a strategy for combined observations with upper bound $\varepsilon_j + \varepsilon'_j$ on the probability to wrongly discard target H_j.*

Proof In fact, the combined strategy is designed so that we discard any H_j if at least one of the predictors X or X' does. The decision to discard a hypothesis is taken independently in both instances. Hence, if O_k, O'_l are observed, we keep H_j with probability

$$(1-x_{kj})(1-x'_{lj}) = 1-(x_{kj}+x'_{lj}-x_{kj}x'_{lj}).$$

On the other hand, since the probability of a union of events is at most the sum of probabilities of the single events, we discard target H_j with *at most* the probability $\varepsilon_j+\varepsilon'_j$. ■

Proposition 10 gives only a guarantee on the error probabilities. However, concavity of exclusiveness (see Proposition 7) suggests that combining two predictors with half error bound in general improves the exclusiveness. For concrete instances and a desired total error probability we may try various partitions into summands, with some resonable step length, and take the combination that works best. We also remark that, since by Theorem 8 and 9 most strategy variables x_{kj}, x'_{lj} are 0 or 1, the calculations are fast.

If several $P^{(i)}$ are available (for example, in our protein structure application, the chemical shifts of 6 nuclei, and also from neighbored residues), then exhaustive search is expensive, but we may choose to combine only the most informative data, that is, only those $P^{(i)}$ with largest exclusiveness.

Finally, a deliberately very simple, symmetric toy example with two hypotheses of equal weight illustrates the principle of combining predictions.

Example 2

$$P = \begin{bmatrix} 1 & 0.5 \\ 0 & 0.5 \end{bmatrix} \; P' = \begin{bmatrix} 0.5 & 1 \\ 0.5 & 0 \end{bmatrix}$$

We choose $\varepsilon = 0.2$ *for both hypotheses in both instances. Then the optimal solutions for the separate instances are, rather obviously:*

$$X = \begin{bmatrix} 0.2 & 0.4 \\ 1 & 0 \end{bmatrix} \; X' = \begin{bmatrix} 0.4 & 0.2 \\ 0 & 1 \end{bmatrix}.$$

In the first instance, the exclusiveness is only 0.6 if H_1 *is the target (since always* O_1 *is observed), and for* H_2 *we get exclusiveness 0.8 (average of both rows), the second instance is symmetric. If we use the information from both instances, we can improve the exclusiveness for the same* $\varepsilon = 0.2$*. First we optimize both instances separately, but now with half error bound* 0.1*:*

$$X = \begin{bmatrix} 0.1 & 0.2 \\ 1 & 0 \end{bmatrix} \; X' = \begin{bmatrix} 0.2 & 0.1 \\ 0 & 1 \end{bmatrix}.$$

For the four combinations $(k,l) = (1,1),(1,2),(2,1),(2,2)$ *we compute new exclusion probabilities as specified above:*

$$\begin{bmatrix} 0.28 & 0.28 \\ 0.1 & 1 \\ 1 & 0.1 \\ 1 & 1 \end{bmatrix}.$$

Since target H_1 *causes the pair of observations* $(1,1)$ *or* $(1,2)$*, each with probability* 0.5*, the exclusiveness is* $0.5 \cdot (0.56+1.1) = 0.83$*, and target* H_2 *yields the same exclusiveness by symmetry. This is considerably better than the worst case above, and even slightly larger than the best case above.*

7. Application Note: Protein Torsion Angle Prediction

We studied properties of a class of linear programs for hypotheses selection in probabilistic inference which is hopefully of fundamental interest. We were led to the problem class by a concrete challenge: a project where we are comparing different methods for predicting protein torsion angles from NMR chemical shifts, see Section 1. Characteristic features of our scatterplot data are large empty regions with almost no data points, in them clouds of data points with a variety of shapes and different densities. Optimization assists in the creation of a predictor:

Any prediction heuristic has to take a measured chemical shift value and output predicted torsion angle values. In a statistical approach it is sensible to precompute the predictions, based on the sampled data. The actual application is then a simple table look-up, done by an auxiliary program. The main relevant question for spectroscopists is the achievable confidence when predicting torsion angle intervals of a prescribed length (error probability vs. exclusiveness, in our terminology). Then they can make their specific decisions using this tradeoff. Besides the actual predictions, the optimization results also quantify how informative the chemical shifts of different nuclei (or their combinations) are for this purpose.

Basic heuristics working purely "row-wise" (MAP, ML, or similar) do not pay attention to error probabilities for specific hypothesis intervals and easily discard certain torsion angles completely, despite a considerable frequency of occurrence. Hence such heuristics generate systematically misleading predictions when these neglected ranges of torsion angles appear. Even worse, they can appear more frequently in a protein under consideration than in the database: Recall that the scatterplots are sampled from a large collection of various proteins so that we know only average torsion angle frequencies. A more even distribution of errors to different torsion angles gives more robustness against varying torsion angle frequencies. We can also expect that the global structure reconstruction process itself works smoother if the local restraints have balanced errors: Most wrong sequences of torsion angles, that is, sequences with errors injected, are already geometrically impossible, which gives us a chance to correct such occasional errors.[1] Since the precise effects are hard to know beforehand, free parameters ε_j seem to be a valuable feature.

A simple MAP heuristics, for example, would take the measured chemical shift value and select the torsion angle ranges (columns) with highest point densities in the row containing the measured value. Other preferences may be taken into account, for instance, one interval is easier to handle as a restraint than a union of several intervals. In either case, a selection yields a matrix X and a vector of error probabilities ε_j which are typically low (high) in densely (sparsely) populated columns. Now we can adjust error probabilities for individual torsion angle intervals in any desired direction and re-optimize.

As an illustration we discuss an arbitrary example (point count matrix) from the real data: Aspartic Acid, nucleus C^α, and torsion angle ϕ, partitioned into homogeneous regions using the method from Christin (2006):

1. As a linguistic analogy, typos scattered in a text can be erased promptly, whereas systematic errors make words unrecognizable, or even smuggle in other words that fit in the context but were not intended.

$$\begin{bmatrix} 1 & 0 & 7 & 12 & 4 & 1 & 1 & 0 \\ 0 & 0 & 0 & 19 & 18 & 1 & 2 & 0 \\ 5 & 4 & 22 & 212 & 116 & 16 & 4 & 0 \\ 2 & 3 & 21 & 90 & 32 & 3 & 5 & 0 \\ 10 & 6 & 38 & 93 & 28 & 7 & 39 & 3 \\ 98 & 86 & 304 & 193 & 39 & 11 & 63 & 5 \\ 34 & 43 & 86 & 27 & 4 & 0 & 7 & 3 \\ 22 & 60 & 67 & 18 & 1 & 1 & 2 & 2 \\ 3 & 12 & 19 & 6 & 4 & 0 & 2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 40 & 30 & 30 & 20 & 15 & 80 & 45 & 100 \end{bmatrix}.$$

The bottom line gives the torsion angle interval lengths in degrees, that is, our weights. The frequencies of hypotheses in the database are (in percent, rounded):

$$\begin{bmatrix} 8.5 & 10.5 & 27.5 & 33.0 & 12.0 & 2.0 & 6.0 & 0.5 \end{bmatrix}.$$

Suppose we want to predict torsion angle intervals of about 60 degrees and start with a naive MAP heuristic that takes the intervals of exactly 60 degrees with maximum density in each row. It leads to the following matrix X (entries indicate the discarded fractions of intervals, values are rounded):

$$\begin{bmatrix} 1 & 1 & 0.17 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0.69 & 1 & 1 \\ 1 & 1 & 0.17 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0.17 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0.17 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0.67 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 100.0 & 33.0 & 2.5 & 7.5 & 19.5 & 99.0 & 100.0 & 100.0 \end{bmatrix}.$$

The bottom line indicates the error bounds ε_j in percent (rounded). For the prior probabilities from the database, the overall error probability would be about 26%. Optimization with the same ε_j yields only marginal changes:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0.69 & 1 & 1 \\ 1 & 1 & 0.18 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0.16 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0.17 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0.67 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0.06 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Columns 6 and 8 are completely discarded, which is reasonable because only 2.5% of cases are to be expected in the corresponding large intervals. The separate cluster in column 7 which appeared with 6% is always discarded, too. We may accept this error when we prefer a single predicted interval to a union of two (see the remark above). The most relevant part is the dense region in columns 1 to 5. Observe that also column 1 is completely discarded, even though it contains a considerable cluster of points. This makes up for 8.5 of the 26% total error. Let us reduce ε_1 and see how this affects the predictions. For instance, after changing ε_1 to 0.4, optimization (followed by raising some sporadic $x_{kj} < 1$ to 1 when p_{kj} is small) yields this X:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0.69 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0.76 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0.26 & 0.7 & 0 & 0.03 & 1 & 1 & 1 & 1 \\ 0.16 & 0 & 0.16 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

We remark that the optimal dual solution moved from $q_3 = 1$ to $q_1 = 1$. The (expected) length of predicted intervals increased to 84 degrees. On the other hand, the global error went down to 21%, and we can afford to raise the very small initial ε_3. For instance, with $\varepsilon_3 = 0.2$ we are back to the initial total error of 26%, now with an expected hypothesis length of 77 degrees and the following X:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0.69 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0.06 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0.46 & 0.7 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0.16 & 0 & 0.4 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Interestingly, this step pressed the predictions in rows 7,8 more to the lower left corner, while x_{61} became higher again. The apparent reason is that, in row 6, the element in column 1 has strong competitors in columns 3 and 4. Hence it is not predicted definitely, even though $p_{61} > p_{71}, p_{81}$. The result in row 6 suggests to choose either columns 1-2 or 3-4. In order to avoid predicting intervals of excessive lengths in some rows, we may cut the longest intervals down, for example, at the end with the smallest increase of error. In our example, the longest predicted interval, with 93 degrees, appears in row 4. Changing x_{42} to 1 increases ε_2 marginally to 0.345 but shortens this interval to 65 degrees. In row 8 we may cut at column 3, etc.

This example merely served to demonstrate that desirable improvements can be made after quick manual checking, while the main calculations are left to any linear programming tool. For processing hundreds of instances with different desired interval lengths we fix the initial ε_j (subject to a proportional factor which is used for the error vs. exclusiveness tradeoff) also by other

plausible heuristics: ε_j proportional to the weight-by-frequency ratio, equal ε_j, and combinations of them. However, since no simple automatic rule seems to be satisfactory for *all* diverse shapes of scatterplots (apparently bad examples exist for each), some minor intervention as shown above is required.

Acknowledgments

The first author has been supported by the Knut and Alice Wallenberg Foundation, Carl Tryggers Foundation, and Assar Gabrielsson Foundation. The second author has been partially supported by the Swedish Research Council (Vetenskapsrådet), grant no. 621-2002-4574.

We thank the anonymous referees for their helpful suggestions.

References

R. D. Beger and P. H. Bolton. Protein ϕ and ψ dihedral restraints determined from multidimensional hypersurface correlations of backbone chemical shifts and their use in the determination of protein tertiary structures. *Journal of Biomolecular NMR*, 10:129–142, 1997.

K. P. Bennett. Decision tree construction via linear programming. In *Proceedings of the 4th Midwest AI and Cognitive Science Sociecty Conference*, pages 97–101, 1992.

K. P. Bennett and O. L. Mangasarian. Neural network training via linear programming. In *Advances in Optimization and Parallel Computing*, pages 56–67. North-Holland, 1992a.

K. P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992b.

K. P. Bennett and O. L. Mangasarian. Multicategory separation via linear programming. *Optimization Methods and Software*, 3:27–39, 1993.

P. S. Bradley. *Mathematical Programming Approaches to Machine Learning and Data Mining*. PhD thesis, University of Wisconsin, 1998.

C. Christin. Scatterplot partitioning algorithm for LETA-NMR. Master's thesis, International Master's Programme in Bioinformatics, Chalmers University, Göteborg (Sweden), 2006.

G. Cornilescu, F. Delaglio, and A. Bax. Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *Journal of Biomolecular NMR*, 13:289–302, 1999.

P. Damaschke. Scheduling search procedures. *Journal of Scheduling*, 7:349–364, 2004.

F. Glover. Improved linear programming models for discriminant analysis. *Decision Sciences*, 21: 771–785, 1990.

S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.

P. Szolovits, R. S. Patil, and W. B. Schwartz. Artificial intelligence in medical diagnosis. *Annals of Internal Medicine*, 108:80–87, 1988.

E. Tardos and K. D. Wayne. Simple generalized maximum flow algorithms. In *Proceedings of the 6th Integer Programming and Combinatorial Optimization Conference, Lecture Notes in Computer Science*, volume 1412, pages 310–324, 1998.

Y. Wang and O. Jardetzky. Probability-based protein secondary structure identification using combined NMR chemical-shift data. *Protein Science*, 11:852–861, 2002.

K. D. Wayne. A polynomial combinatorial algorithm for generalized minimum cost flow. *Mathematical Operations Research*, 27:445–459, 2002.

X. P. Xu and D. A. Case. Probing multiple effects on ^{15}N, $^{13}C^{\alpha}$, $^{13}C^{\beta}$ and ^{13}C' chemical shifts in peptides using density functional theory. *Biopolymers*, 65:408–423, 2002.

Journal of Machine Learning Research 7 (2006) 1357–1383 Submitted 10/05; Revised 5/06; Published 7/06

Bayesian Network Learning with Parameter Constraints

Radu Stefan Niculescu STEFAN.NICULESCU@SIEMENS.COM
Computer Aided Diagnosis and Therapy Group
Siemens Medical Solutions
51 Valley Stream Parkway
Malvern, PA, 19355, USA

Tom M. Mitchell TOM.MITCHELL@CMU.EDU
Center for Automated Learning and Discovery
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213, USA

R. Bharat Rao BHARAT.RAO@SIEMENS.COM
Computer Aided Diagnosis and Therapy Group
Siemens Medical Solutions
51 Valley Stream Parkway
Malvern, PA, 19355, USA

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

The task of learning models for many real-world problems requires incorporating domain knowledge into learning algorithms, to enable accurate learning from a realistic volume of training data. This paper considers a variety of types of domain knowledge for constraining parameter estimates when learning Bayesian networks. In particular, we consider domain knowledge that constrains the values or relationships among subsets of parameters in a Bayesian network with known structure.

We incorporate a wide variety of parameter constraints into learning procedures for Bayesian networks, by formulating this task as a constrained optimization problem. The assumptions made in module networks, dynamic Bayes nets and context specific independence models can be viewed as particular cases of such parameter constraints. We present closed form solutions or fast iterative algorithms for estimating parameters subject to several specific classes of parameter constraints, including equalities and inequalities among parameters, constraints on individual parameters, and constraints on sums and ratios of parameters, for discrete and continuous variables. Our methods cover learning from both frequentist and Bayesian points of view, from both complete and incomplete data.

We present formal guarantees for our estimators, as well as methods for automatically learning useful parameter constraints from data. To validate our approach, we apply it to the domain of fMRI brain image analysis. Here we demonstrate the ability of our system to first learn useful relationships among parameters, and then to use them to constrain the training of the Bayesian network, resulting in improved cross-validated accuracy of the learned model. Experiments on synthetic data are also presented.

Keywords: Bayesian networks, constrained optimization, domain knowledge

1. Introduction

Probabilistic models have become increasingly popular in the last decade because of their ability to capture non-deterministic relationships among variables describing many real world domains. Among these models, graphical models have received significant attention because of their ability to compactly encode conditional independence assumptions over random variables and because of the development of effective algorithms for inference and learning based on these representations.

A Bayesian network (Heckerman, 1999) is a particular case of a graphical model that compactly represents the joint probability distribution over a set of random variables. It consists of two components: a structure and a set of parameters. The structure is a directed acyclic graph with one node per variable. This structure encodes the cocal Markov assumption: a variable is conditionally independent of its non-descendants in the network, given the value of its parents. The parameters describe how each variable relates probabilistically to its parents. A Bayesian network encodes a unique joint probability distribution, which can be easily computed using the chain rule.

When learning Bayesian networks, the correctness of the learned network of course depends on the amount of training data available. When training data is scarce, it is useful to employ various forms of prior knowledge about the domain to improve the accuracy of learned models. For example, a domain expert might provide prior knowledge specifying conditional independencies among variables, constraining or even fully specifying the network structure of the Bayesian network. In addition to helping specify the network structure, the domain expert might also provide prior knowledge about the values of certain parameters in the conditional probability tables (CPTs) of the network, or knowledge in the form of prior distributions over these parameters. While previous research has examined a number of approaches to representing and utilizing prior knowledge about Bayesian network parameters, the type of prior knowledge that can be utilized by current learning methods remains limited, and is insufficient to capture many types of knowledge that may be readily available from experts.

One contribution of our previous work (Niculescu, 2005) was the development of a general framework to perform parameter estimation in Bayesian networks in the presence of any parameter constraints that obey certain differentiability assumptions, by formulating this as a constrained maximization problem. In this framework, algorithms were developed from both a frequentist and a Bayesian point of view, for both complete and incomplete data. The optimization methods used by our algorithms are not new and therefore this general learning approach has serious limitations given by these methods, especially given arbitrary constraints. However, this approach constitutes the basis for the efficient learning procedures for specific classes of parameter constraints described in this paper. Applying these efficient methods allows us to take advantage of parameter constraints provided by experts or learned by the program, to perform more accurate learning of very large Bayesian networks (thousands of variables) based on very few (tens) examples, as we will see later in the paper.

The main contribution of our paper consists of efficient algorithms (closed form solutions, fast iterative algorithms) for several classes of parameter constraints which current methods can not accommodate. We show how widely used models including hidden Markov models, dynamic Bayesian networks, module networks and context specific independence are special cases of one of our constraint types, described in subsection 4.3. Our framework is able to represent parameter sharing assumptions at the level of granularity of individual parameters. While prior work on parameter sharing and Dirichlet priors can only accommodate simple equality constraints between

parameters, our work extends to provide closed form solutions for classes of parameter constraints that involve relationships between groups of parameters (sum sharing, ratio sharing). Moreover, we provide closed form maximum likelihood estimators when constraints come in the form of several types of inequality constraints. With our estimators come a series of formal guarantees: we formally prove the benefits of taking advantage of parameter constraints to reduce the variance in parameter estimators and we also study the performance in the case when the domain knowledge represented by the parameter constraints might not be entirely accurate. Finally, we present a method for automatically learning parameter constraints, which we illustrate on a complex task of modelling the fMRI brain image signal during a cognitive task.

The next section of the paper describes related research on constraining parameter estimates for Bayesian networks. Section 3 presents the problem and describes our prior work on a framework for incorporating parameter constraints to perform estimation of parameters of Bayesian networks. Section 4 presents the main contribution of this paper: very efficient ways (closed form solutions, fast iterative algorithms) to compute parameter estimates for several important classes of parameter constraints. There we show how learning in current models that use parameter sharing assumptions can be viewed as a special case of our approach. In section 5, experiments on both real world and synthetic data demonstrate the benefits of taking advantage of parameter constraints when compared to baseline models. Some formal guarantees about our estimators are presented in section 6. We conclude with a brief summary of this research along with several directions for future work.

2. Related Work

The main methods to represent relationships among parameters fall into two main categories: Dirichlet priors and their variants (including smoothing techniques) and parameter sharing of several kinds.

In Geiger and Heckerman (1997), it is shown that Dirichlet priors are the only possible priors for discrete Bayes nets, provided certain assumptions hold. One can think of a Dirichlet prior as an expert's guess for the parameters in a discrete Bayes net, allowing room for some variance around the guess. One of the main problems with Dirichlet priors and related models is that it is impossible to represent even simple equality constraints between parameters (for example the constraint: $\theta_{111} = \theta_{121}$ where $\theta_{ijk} = P(X_i = x_{ij}|Parents(X_i) = pa_{ik})$) without using priors on the hyperparameters of the Dirichelet prior, in which case the marginal likelihood can no longer be computed in closed form, and expensive approximate methods are required to perform parameter estimation. A second problem is that it is often beyond the expert's ability to specify a full Dirichlet prior over the parameters of a Bayesian network.

Extensions of Dirichlet priors include Dirichlet tree priors (Minka, 1999) and dependent Dirichlet priors (Hooper, 2004). Although these priors allow for more correlation between the parameters of the model than standard Dirichlet priors, essentially they face the same issues. Moreover, in the case of dependent Dirichlet priors, parameter estimators can not be computed in closed form, although Hooper (2004) presents a method to compute approximate estimators, which are linear rational fractions of the observed counts and Dirichlet parameters, by minimizing a certain mean square error measure. Dirichlet priors can be considered to be part of a broader category of methods that employ parameter domain knowledge, called smoothing methods. A comparison of several smoothing methods can be found in Zhai and Lafferty (2001).

A widely used form of parameter constraints employed by Bayesian networks is *parameter sharing*. Models that use different types of parameter sharing include: dynamic Bayesian networks

(Murphy, 2002) and their special case hidden Markov models (Rabiner, 1989), module networks (Segal et al., 2003), context specific independence models (Boutilier et al., 1996) such as Bayesian multinetworks, recursive multinetworks and dynamic multinetworks (Geiger and Heckerman, 1996; Pena et al., 2002; Bilmes, 2000), probabilistic relational models (Friedman et al., 1999), object oriented Bayes nets (Koller and Pfeffer, 1997), Kalman filters (Welch and Bishop, 1995) and bilinear models (Tenenbaum and Freeman, 2000). Parameter sharing methods constrain parameters to share the same value, but do not capture more complicated constraints among parameters such as inequality constraints or constraints on sums of parameter values. The above methods are restricted to sharing parameters at either the level of sharing a conditional probability table (CPT) (module networks, HMMs), at the level of sharing a conditional probability distribution within a single CPT (context specific independence), at the level of sharing a state-to-state transition matrix (Kalman filters) or at the level of sharing a style matrix (bilinear models). None of the prior models allow sharing at the level of granularity of individual parameters.

One additional type of parameter constraints is described by *probabilistic rules*. This kind of domain knowledge was used in Rao et al. (2003) to assign values to certain parameters of a Bayesian network. We are not aware of probabilistic rules being used beyond that purpose for estimating the parameters of a Bayesian network.

3. Problem Definition and Approach

Here we define the problem and describe our previous work on a general optimization based approach to solve it. This approach has serious limitations when the constraints are arbitrary. However, it constitutes the basis for the very efficient learning procedures for the classes of parameter constraints described in section 4. While the optimization methods we use are not new, applying them to our task allows us to take advantage of expert parameter constraints to perform more accurate learning of very large Bayesian networks (thousands of variables) based on very few (tens) examples, as we will see in subsection 5.2. We begin by describing the problem and state several assumptions that we make when deriving our estimators.

3.1 The Problem

Our task here is to perform parameter estimation in a Bayesian network where the structure is known in advance. To accomplish this task, we assume a data set of examples is available. In addition, a set of parameter equality and/or inequality constraints is provided by a domain expert. The equality constraints are of the form $g_i(\theta) = 0$ for $1 \leq i \leq m$ and the inequality constraints are of the form $h_j(\theta) \leq 0$ for $1 \leq j \leq k$, where θ represents the set of parameters of the Bayesian network.

Initially we will assume the domain knowledge provided by the expert is correct. Later, we investigate what happens if this knowledge is not completely correct. Next we enumerate several assumptions that must be satisfied for our methods to work. These are similar to common assumptions made when learning parameters in standard Bayesian networks.

First, we assume that the examples in the training data set are drawn independently from the underlying distribution. In other words, examples are conditionally independent given the parameters of the graphical model.

Second, we assume that all the variables in the Bayesian network can take on at least two different values. This is a safe assumption since there is no uncertainty in a random variable with

only one possible value. Any such variables in our Bayesian network can be deleted, along with all arcs into and out of the nodes corresponding to those variables.

When computing parameter estimators in the discrete case, we additionally assume that all observed counts corresponding to parameters in the Bayesian network are strictly positive. We enforce this condition in order to avoid potential divisions by zero, which may impact inference negatively. In the real world it is expected there will be observed counts which are zero. This problem can be solved by using priors on parameters, that essentially have the effect of adding a positive quantity to the observed counts and essentially create strictly positive virtual counts.

Finally, the functions $g_1, \ldots, g_m$ and $h_1, \ldots, h_k$ must be twice differentiable, with continuous second derivatives. This assumption justifies the formulation of our problem as a constrained maximization problem that can be solved using standard optimization methods.

3.2 A General Approach

In order to solve the problem described above, here we briefly mention our previous approach (Niculescu, 2005) based on already existing optimization techniques. The idea is to formulate our problem as a constrained maximization problem where the objective function is either the data log-likelihood $\log P(D|\theta)$ (for maximum likelihood estimation) or the log-posterior $\log P(\theta|D)$ (for maximum aposteriori estimation) and the constraints are given by $g_i(\theta) = 0$ for $1 \leq i \leq m$ and $h_j(\theta) \leq 0$ for $1 \leq j \leq k$. It is easy to see that, applying the Karush-Kuhn-Tucker conditions theorem (Kuhn and Tucker, 1951), the maximum must satisfy a system with the same number of equations as variables. To solve this system, one can use any of several already existing methods (for example the Newton-Raphson method (Press et al., 1993)).

Based on this approach, in Niculescu (2005) we develop methods to perform learning from both a frequentist and a Bayesian point of view, from both fully and partially observable data (via an extended EM algorithm). While it is well known that finding a solution for the system given by the KKT conditions is not enough to determine the optimum point, in Niculescu (2005) we also discuss when our estimators meet the sufficiency criteria to be optimum solutions for the learning problem. There we also describe how to use *constrained conjugate parameter priors* for the MAP estimation and Bayesian model Averaging. A sampling algorithm was devised to address the challenging issue of computing the normalization constant for these priors. Furthermore, procedures that allow the automatic learning of useful parameter constraints were also derived.

Unfortunately, the above methods have a very serious shortcoming in the general case. With a large number of parameters in the Bayesian network, they can be extremely expensive because they involve potentially multiple runs of the Newton-Raphson method and each such run requires several expensive matrix inversions. Other methods for finding the solutions of a system of equations can be employed, but, as noted in Press et al. (1993), all these methods have limitations in the case when the constraints are arbitrary, non-linear functions. The worst case happens when there exists a constraint that explicitly uses all parameters in the Bayesian network.

Because of this shortcoming and because the optimization methods we use to derive our algorithms are not new, we choose not to go into details here. We mention them to show how learning in the presence of parameter constraints can be formulated as a general constrained maximization problem. This general framework also provides the starting point for the efficient learning methods for the particular classes of parameter constraints presented in the next section.

4. Parameter Constraints Classes

In the previous section we mentioned the existence of general methods to perform parameter learning in Bayesian networks given a set of parameter constraints. While these methods can deal with arbitrary parameter constraints that obey several smoothness assumptions, they can be very slow since they involve expensive iterative and sampling procedures.

Fortunately, in practice, parameter constraints usually involve only a small fraction of the total number of parameters. Also, the data log-likelihood can be nicely decomposed over examples, variables and values of the parents of each variable (in the case of discrete variables). Therefore, the maximum likelihood optimization problem can be split into a set of many independent, more manageable, optimization subproblems, which can either be solved in closed form or for which very efficient algorithms can be derived. For example, in standard maximum likelihood estimation of the parameters of a Bayesian network, each such subproblem is defined over one single conditional probability distribution. In general, in the discrete case, each optimization subproblem will span its own set of conditional probability distributions. The set of maximum likelihood parameters will be the union of the solutions of these subproblems.

This section shows that for several classes of parameter constraints the system of equations given by the Karush-Kuhn-Tucker theorem can be solved in an efficient way (closed form or fast iterative algorithm). In some of these cases we are also able to find a closed form formula for the normalization constant of the corresponding constrained parameter prior.

4.1 Parameter Sharing within One Distribution

This class of parameter constraints allows asserting that specific user-selected parameters within a single conditional probability distribution must be shared. This type of constraint allows representing statements such as: *"Given this combination of causes, several effects are equally likely."* Since the scope of this constraint type does not go beyond the level of a single conditional probability distribution within a single CPT, the problem of maximizing the data likelihood can be split into a set of independent optimization subproblems, one for each such conditional probability distribution. Let us consider one of these subproblems (for a variable X and a specific value $PA(X) = pa$ of the parents). Assume the parameter constraint asserts that several parameters are equal by asserting that the parameter θ_i appears in k_i different positions in the conditional distribution. Denote by N_i the cumulative observed count corresponding to θ_i. The cumulative observed count is the sum of all the observed counts corresponding to the k_i positions where θ_i appears in the distribution. Let $N = \sum_i N_i$ be the sum of all observed counts in this conditional probability distribution i.e. the total number of observed cases with $PA(X) = pa$.

At first it may appear that we can develop maximum likelihood estimates for θ_i and the other network parameters using standard methods, by introducing new variables that capture the groups of shared parameters. To see that this is not the case, consider the following example. Assume a variable X with values $\{1,2,3,4\}$ depends on Y. Moreover, assume the parameter constraint states that $P(X = 1|Y = 0) = P(X = 2|Y = 0)$ and $P(X = 3|Y = 0) = P(X = 4|Y = 0)$. Then one can introduce variable X_{12} which is 1 if $X \in \{1,2\}$ and 0 otherwise. This variable is assumed dependent on Y and added as a parent of X. It is easy to see that $P(X|X_{12} = 0, Y = 0)$ must be equal to the distribution on $\{1,2,3,4\}$ that assigns half probability to each of 3 and 4. Therefore, if Y takes only one value, the task of finding maximum likelihood estimators with parameter sharing is reduced to the one of finding standard maximum likelihood estimators for $X_{12}|Y = 0$. However, if Y takes only

one value, then we can safely remove it as a parent of X. When Y can take two values, 0 and 1, assume the expert states the additional assumption that $P(X=1|Y=1)=P(X=3|Y=1)=P(X=4|Y=1)$. Now we need to introduce a new variable X_{134} that depends on Y and add it as a parent of X. There must be an edge between X_{12} and X_{134} because, otherwise, the structural assumption that X_{12} and X_{134} are conditionally independent given Y is obviously not true. Assuming X_{12} is the parent of X_{134}, the constraints given by the expert need to be modelled in the distribution $P(X_{134}|X_{12},Y)$ instead. Not only did we fail to encode the constraints in the new structure, but we also complicated the problem by adding two nodes in our network. A similar argument holds for all discrete types of parameter constraints presented in this section.

Below we present closed form solutions for the maximum likelihood estimators from complete data and for the normalization constant for the corresponding constrained Dirichlet priors used to perform maximum aposteriori estimation. These priors are similar to the standard Dirichlet priors, but they assign zero probability over the space where the expert's constraints are not satisfied. The normalization constant for the constrained Dirichlet prior can be computed over the scope of a certain constraint and then all such constants are multiplied to obtain the normalization constant for the prior over the whole set of parameters of the Bayesian network. We also present an EM algorithm to approach learning from incomplete data under this type of parameter sharing.

4.1.1 MAXIMUM LIKELIHOOD ESTIMATION FROM COMPLETE DATA

Theorem 1 *The maximum likelihood estimators for the parameters in the above conditional probability distribution are given by:*

$$\hat{\theta}_i = \frac{N_i}{k_i \cdot N}$$

Proof The problem of maximizing the data log-likelihood subject to the parameter sharing constraints can be broken down in subproblems, one for each conditional probability distribution. One such subproblem can be restated as:

$$P : argmax \{h(\theta) \mid g(\theta) = 0\}$$

$$\text{where } h(\theta) = \sum_i N_i \log \theta_i \text{ and } g(\theta) = (\sum_i k_i \cdot \theta_i) - 1 = 0$$

When all counts are positive, it can be easily proved that P has a global maximum which is achieved in the interior of the region determined by the constraints. In this case the solution of P can be found using Lagrange multipliers. Introduce Lagrange multiplier λ for the constraint in P. Let $LM(\theta,\lambda) = h(\theta) - \lambda \cdot g(\theta)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\theta,\lambda) = 0$. Let $(\hat{\theta},\lambda)$ be a solution of this system. We have: $0 = \frac{\partial LM}{\partial \theta_i} = \frac{N_i}{\theta_i} - \lambda \cdot k_i$ for all i. Therefore, $k_i \cdot \hat{\theta}_i = \frac{N_i}{\lambda}$. Summing up for all values of i, we obtain:

$$0 = \frac{\partial LM}{\partial \lambda} = (\sum_i k_i \cdot \hat{\theta}_i) - 1 = (\sum_i \frac{N_i}{\lambda}) - 1 = \frac{N}{\lambda} - 1$$

From the last equation we compute the value of $\lambda = N$. This gives us: $\hat{\theta}_i = \frac{N_i}{k_i \cdot N}$. The fact that $\hat{\theta}$ is the set of maximum likelihood estimators follows because the objective function is concave and because the constraint is a linear equality. ■

4.1.2 CONSTRAINED DIRICHLET PRIORS

From a Bayesian point of view, each choice of parameters can occur with a certain probability. To make learning easier, for this type of parameter constraints, we employ conjugate constrained Dirichlet priors that have the following form for a given conditional probability distribution in the Bayesian network:

$$P(\theta) = \begin{cases} \frac{1}{Z}\prod_{i=1}^{n}\theta_i^{\alpha_i-1} & \text{if } \theta \geq 0, \sum k_i \cdot \theta_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

maximum aposteriori estimation can be now performed in exactly the same way as maximum likelihood estimation (see Theorem 1), with the only difference that the objective function becomes $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$. The normalization constant Z can be computed by integration and depends on the elimination order. If θ_n is eliminated first, we obtain:

$$Z_n = \frac{k_n}{\prod_{i=1}^{n} k_i^{\alpha_i}} \cdot \frac{\prod_{i=1}^{n}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n}\alpha_i)}$$

The above normalization should be thought of as corresponding to $P(\theta_1, \ldots, \theta_{n-1})$. If we eliminate a different parameter first when computing the integral, then we obtain a different normalization constant which corresponds to a different $(n-1)$-tuple of parameters. Note that having different constants is not an inconsistency, because the corresponding probability distributions over $n-1$ remaining parameters can be obtained from each other by a variable substitution based on the constraint $\sum k_i \cdot \theta_i = 1$. It is easy to see (Niculescu, 2005) that learning procedures are not affected in any way by which parameter is eliminated. In the case of no parameter sharing (that is $k_i = 1$ for all i), all these normalization constants are equal and we obtain the standard Dirichlet prior.

4.1.3 MAXIMUM LIKELIHOOD ESTIMATION FROM INCOMPLETE DATA

It can be easily proved that learning with incomplete data can be achieved via a modified version of the standard expectation maximization algorithm used to train Bayesian networks, where in the E-Step the expected counts are estimated and in the M-Step parameters are re-estimated using these expected counts based on Theorem 1.

Algorithm 1 (Expectation maximization for discrete Bayesian networks) *Randomly initialize the network parameters with a value $\hat{\theta}^0$. Repeat the following two steps until convergence is reached:*

E-Step*: At iteration $t+1$, use any inference algorithm to compute expected counts $E[N_i|\hat{\theta}^t]$ and $E[N|\hat{\theta}^t]$ for each distribution in the network under the current parameter estimates $\hat{\theta}^t$.*

M-Step*: Re-estimate the parameters $\hat{\theta}^{t+1}$ using Theorem 1, assuming that the observed counts are equal to the expected counts given by the E-Step.*

4.2 Parameter Sharing in Hidden Process Models

A *hidden process model (HPM)* is a probabilistic framework for modelling time series data (Hutchinson et al., 2005, 2006) which predicts the value of a *target variable X* at a given point in time as the

sum of the values of certain *hidden processes* that are active. The HPM model is inspired by our interest in modelling hidden cognitive processes in the brain, given a time series of observed fMRI images of brain activation. One can think of the observed image feature X as the value of the fMRI signal in one small cube inside the brain (also called a voxel). A hidden process may be thought of as a mental process that generates fMRI activity at various locations in the brain, in response to an external *stimulus*. For example, a *"ComprehendPicture"* process may describe the fMRI signal that happens in the brain starting when the subject is presented with a picture. A *"ComprehendSentence"* process may provide the same characterization for the situation when a subject is reading a sentence. HPMs assume several cognitive processes may be active at the some point in time, and assume in such cases the observed fMRI signal is the sum of the corresponding processes, translated according to their starting times. Hidden process models can be viewed as a subclass of dynamic Bayesian networks, as described in Hutchinson et al. (2006).

Formally, a *hidden process model* is defined by a collection of time series (also called hidden processes): $P_1, \ldots, P_K$. For each process P_k with $1 \leq k \leq K$, denote by P_{kt} the value of its corresponding time series at time t after the process starts. Also, let X_t be the value of the target variable X at time t. If process P_k starts at time t_k, then a hidden process model predicts the random variable X_t will follow the distribution:

$$X_t \sim N(\sum_k P_{k(t-t_k+1)}, \sigma^2)$$

where σ^2 is considered to be the variance in the measurement and is kept constant across time. For the above formula to make sense, we consider $P_{kt} = 0$ if $t < 0$. Figure 1 shows an example of a hidden process model for the fMRI activity in a voxel in the brain during a cognitive task involving reading a sentence and looking at a picture.

In general HPMs allow modeling uncertainty about the timing of hidden processes, allow uncertainty about the types of the processes, and allow for multiple instances of the same process to be active simultaneously (Hutchinson et al., 2006). However, in the treatment and experiments in this paper we make three simplifying assumptions. We assume the times at which the hidden processes occur are known, that the types of the processes are known, and that two instances of the same types of process may not be active simultaneously. These three simplifying assumptions lead to a formulation of HPMs that is equivalent to the analysis approach of Dale (1999) based on multivariate regression within the general linear model.

In a typical fMRI experiment, the subject often performs the same cognitive task multiple times, on multiple *trials*, providing multiple observation sequences of the variable X. In our framework we denote by X_{nt} the value of X_t during trial n, and by t_{nk} the starting point of process P_k during trial n. Let N be the total number of observations. We can now write:

$$X_{nt} \sim N(\sum_k P_{k(t-t_{nk}+1)}, \sigma^2)$$

While not entirely necessary for our method to work, we assume that X is tracked for the same length of time in each trial. Let T be the length of every such trial (observation). Since we are not modelling what happens when $t > T$, we can also assume that each process has length T.

The natural constraints of this domain lead to an opportunity to specify prior knowledge in the form of parameter constraints, as follows: an external stimulus will typically influence the activity in multiple voxels of the brain during one cognitive task. For example, looking at a picture may activate

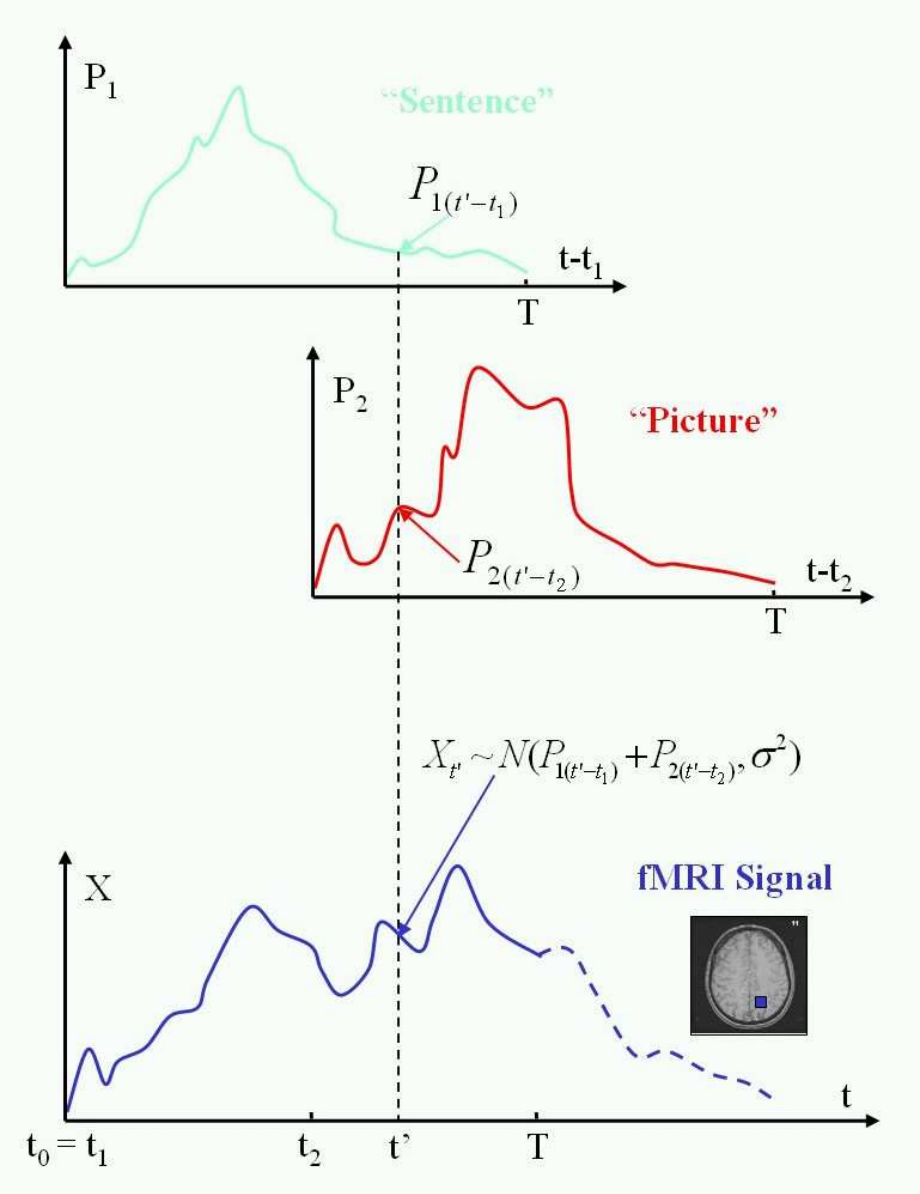


Figure 1: A hidden process model to model a human subject who is asked to read a sentence and to look at a picture. In half of the observations, the sentence is presented first, then the picture is shown. In the other half of the observations, the picture is presented first. The activity in a given voxel X in the brain is modelled as a hidden process model with two processes: "Sentence" (P_1) and "Picture" (P_2). Each observation has length $T = 32$ fMRI snapshots (16 seconds) and the same holds for both processes. This figure shows an observation where the sentence is presented at time $t_1 = 1$ and the picture is shown at $t_2 = 17$ (8 seconds after t_1). After time t_2, the two processes overlap and the fMRI signal $X_{t'}$ is the sum of the corresponding values of the two processes plus $N(0, \sigma^2)$ measurement variance. The blue dotted line represents the fMRI activity that would happen after time T.

many voxels in the visual cortex. The activation in these voxels may be different at each given point in time. Intuitively, that means the same stimulus may produce different hidden processes in different voxels. However, certain groups of voxels that are close together often have similarly shaped time series, but with different amplitude. In this case, we believe it is reasonable to assume that the underlying hidden processes corresponding to these voxels are proportional to one another. Experiments performed in Section 5 will prove that this assumption will help learn better models than the ones that choose to ignore it.

In the above paragraph we explained intuitively that sometimes it makes sense to share the same base processes across several time-varying random variables, but allow for different scaling factors. Formally, we say that time-varying random variables $X^1,\ldots,X^V$ share their corresponding *hidden process models* if there exist base processes $P_1,\ldots,P_K$ and constants c_k^v for $1 \le v \le V$ such that:

$$X_{nt}^v \sim N(\sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)}, \sigma^2)$$

and the values of different variables X^v are independent given the parameters of the model. Here σ^2 represents the variance in measurement which is also shared across these variables.

We now consider how to efficiently perform maximum likelihood estimation of the parameters of the variables $X^1,\ldots,X^V$, assuming that they share their corresponding hidden process model parameters as described above. The parameters to be estimated are the base process parameters P_{kt} where $1 \le k \le K$ and $1 \le t \le T$, the scaling constants c_k^v (one for each variable V and process k) where $1 \le v \le V$ and the common measurement variance σ^2. Let $P = \{P_{kt} \mid 1 \le k \le K,\ 1 \le t \le T\}$ be the set of all parameters involved in the base processes and let $C = \{c_k^v \mid 1 \le k \le K,\ 1 \le v \le V\}$ be the set of scaling constants. Subsequently, we will think of these sets as column vectors. Recall that N represents the number of observations. After incorporating the parameter sharing constraints in the log-likelihood function, our optimization problem becomes:

$$P : argmax\ \ l(P,C,\sigma)$$

where

$$l(P,C,\sigma) \quad = \quad -\frac{NTV}{2} \cdot \log(2\pi) - NTV \cdot \log(\sigma) - \frac{1}{2 \cdot \sigma^2} \cdot \sum_{n,t,v} (x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)})^2$$

It is easy to see that the value of (P,C) that maximizes l is the same for all values of σ. Therefore, to maximize l, we can first minimize $l'(P,C) = \sum_{n,t,v}(x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)})^2$ with respect to (P,C) and then maximize l with respect to σ based on the minimum point for l'. One may notice that l' is a sum of squares, where the quantity inside each square can be seen as a linear function in both P and C. Therefore one can imagine an iterative procedure that first minimizes with respect to P, then with respect to C using the Least Squares method. Once we find $M = \min l'(P,C) = l'(\hat{P},\hat{C})$, the value of σ that maximizes l is given by $\hat{\sigma}^2 = \frac{M}{NVT}$. This can be derived in a straightforward fashion by enforcing $\frac{\partial l}{\partial \sigma}(\hat{P},\hat{C},\hat{\sigma}) = 0$. With these considerations, we are now ready to present an algorithm to compute maximum likelihood estimators $(\hat{P},\hat{C},\hat{\sigma})$ of the parameters in the shared hidden process model:

Algorithm 2 (Maximum likelihood estimators in a shared hidden process model) *Let $\bar{X}$ be the column vector of values x_{nt}^v. Start with a random guess $(\hat{P},\hat{C})$ and then repeat Steps 1 and 2 until*

they converge to the minimum of the function $l'(P,C)$.

STEP 1. *Write $l'(\hat{P},\hat{C}) = ||A \cdot \hat{P} - \bar{X}||^2$ where A is a NTV by KT matrix that depends on the current estimate $\hat{C}$ of the scaling constants. More specifically, each row of A corresponds to one of the squares from l' and each column corresponds to one of the KT parameters of the base processes (the column number associated with such a parameter must coincide with its position in column vector P). Minimize with respect to $\hat{P}$ using ordinary Least Squares to get a new estimate $\hat{P} = (A^T \cdot A)^{-1} \cdot A^T \cdot \bar{X}$.*

STEP 2.*Write $l'(\hat{P},\hat{C}) = ||B \cdot \hat{C} - \bar{X}||^2$ where B is a NTV by KV matrix that depends on the current estimate $\hat{P}$ of the base processes. More specifically, each row of B corresponds to one of the squares from l' and each column corresponds to one of the KV scaling constants (the column number associated with such a constant must coincide with its position in column vector C). Minimize with respect to $\hat{C}$ using ordinary Least Squares to get a new estimate $\hat{C} = (B^T \cdot B)^{-1} \cdot B^T \cdot \bar{X}$.*

STEP 3. *Once convergence is reached by repeating the above two steps, let $\hat{\sigma}^2 = \frac{l'(\hat{P},\hat{C})}{NVT}$.*

It might seem that this is a very expensive algorithm because it is an iterative method. However, we found that when applied to fMRI data in our experiments, it usually converges in 3-5 repetitions of Steps 1 and 2. We believe that the main reason why this happens is because at each partial step during the iteration we compute a closed form global minimizer on either $\hat{P}$ or $\hat{C}$ instead of using a potentially expensive gradient descent algorithm. In Section 5 we will experimentally prove the benefits of this algorithm over methods that do not take advantage of parameter sharing assumptions.

One may suspect that it is easy to learn the parameters of the above model because it is a particular case of bilinear model. However, this is not the case. In the bilinear model representation (Tenenbaum and Freeman, 2000), the style matrices will correspond to process parameters *P* and the content vectors will correspond to scaling constants. It is easy to see that in our case the style matrices have common pieces, depending on when the processes started in each example. Therefore, the SVD method presented in Tenenbaum and Freeman (2000) that assumes independence of these style matrices is not appropriate in our problem.

4.3 Other Classes of Parameter Constraints

In the above subsections we discussed efficient methods to perform parameter estimation for two types of parameter constraints: one for discrete variables and one for continuous variables. These methods bypass the need for the potentially expensive use of methods such as Newton-Raphson. There are a number of additional types of parameter constraints for which we have developed closed form maximum likelihood and maximum aposteriori estimators: equality and inequality constraints, on individual parameters as well as on sums and ratios of parameters, for discrete and continuous variables. Moreover, in some of these cases, we were able to compute the normalization constant in closed form for the corresponding constrained priors, which allows us to perform parameter learning from a Bayesian point of view. All these results can be found in Niculescu (2005). We briefly describe these types of parameter constraints below, and provide real-world examples of prior knowledge that can be expressed by each form of constraint.

- *Constraint Type 1: Known Parameter values, Discrete.* Example: If a patient has a heart attack (Disease = "Heart Attack"), then there is a 90% probability that the patient will experience chest pain.

- *Constraint Type 2: Parameter Sharing, One Distribution, Discrete.* Example: Given a combination of risk factors, several diseases are equally likely.

- *Constraint Type 3: Proportionality Constants, One Distribution, Discrete.* Example: Given a combination of risk factors, disease A is twice as likely to occur as disease B.

- *Constraint Type 4: Sum Sharing, One Distribution, Discrete.* Example: A patient who is a smoker has the same chance of having a Heart Disease (Heart Attack or Congestive Heart Failure) as having a Pulmonary Disease (Lung Cancer or Chronic Obstructive Pulmonary Disease).

- *Constraint Type 5: Ratio Sharing, One Distribution, Discrete.* Example: In a bilingual corpus, the relative frequencies of certain groups of words are the same, even though the aggregate frequencies of these groups may be different. Such groups of words can be: "words about computers" ("computer", "mouse", "monitor", "keyboard" in both languages) or "words about business", etc. In some countries computer use is more extensive than in others and one would expect the aggregate probability of "words about computers" to be different. However, it would be natural to assume that the relative proportions of the "words about computers" are the same within the different languages.

- *Constraint Type 6: General Parameter Sharing, Multiple Distributions, Discrete.* Example: The probability that a person will have a heart attack given that he is a smoker with a family history of heart attack is the same whether or not the patient lives in a polluted area.

- *Constraint Type 7: Hierarchical Parameter Sharing, Multiple Distributions, Discrete.* Example: The frequency of several *international words* (for instance "computer") may be shared across both Latin languages (Spanish, Italian) and Slavic languages (Russian, Bulgarian). Other Latin words will have the same frequency only across Latin languages and the same holds for Slavic Languages. Finally, other words will be language specific (for example names of country specific objects) and their frequencies will not be shared with any other language.

- *Constraint Type 8: Sum Sharing, Multiple Distributions, Discrete.* Example: The frequency of nouns in Italian is the same as the frequency of nouns in Spanish.

- *Constraint Type 9: Ratio Sharing, Multiple Distributions, Discrete.* Example: In two different countries (A and B), the relative frequency of Heart Attack to Angina Pectoris as the main diagnosis is the same, even though the the aggregate probability of Heart Disease (Heart Attack and Angina Pectoris) may be different because of differences in lifestyle in these countries.

- *Constraint Type 10: Inequalities between Sums of Parameters, One Distribution, Discrete.* Example: The aggregate probability mass of adverbs is no greater than the aggregate probability mass of the verbs in a given language.

- *Constraint Type 11: Upper Bounds on Sums of Parameters, One Distribution, Discrete.* Example: The aggregate probability of nouns in English is no greater than 0.4.
- *Constraint Type 12: Parameter Sharing, One Distribution, Continuous.* Example: The stock of computer maker *DELL* as a Gaussian whose mean is a weighted sum of the stocks of software maker *Microsoft (MSFT)* and chip maker *Intel (INTL)*. Parameter sharing corresponds to the statement that *MSFT* and *INTL* have the same importance (weight) for predicting the value of stock *DELL*.
- *Constraint Type 13: Proportionality Constants, One Distribution, Continuous.* Example: Suppose we also throw in the stock of a Power Supply maker (PSUPPLY) in the linear mix in the above example. The expert may give equal weights to INTL and MSFT, but five times lower to PSUPPLY.
- *Constraint Type 14: Parameter Sharing for Hidden Process Models.* Example: Several neighboring voxels in the brain exhibit similar activation patterns, but with different amplitudes when a subject is presented with a given stimulus.

Note that general parameter sharing (Constraint Type 6) encompasses models including HMMs, dynamic Bayesian networks, module networks and context specific independence as particular cases, but allows for much finer grained sharing, at the level of individual parameters, across different variables and across distributions of different lengths. Briefly, this general parameter sharing allows for a group of conditional probability distributions to share some parameters across all distributions in the group, but not share the remaining parameters. This type of parameter constraint is described in more detail in Niculescu et al. (2005) where we demonstrate our estimators on a task of modelling synthetic emails generated by different subpopulations.

It is also important to note that different types of parameter constraints can be mixed together when learning the parameters of a Bayesian network as long as the scopes of these constraints do not overlap.

5. Experiments

In this section we present experiments on both synthetic and real world data. Our experiments demonstrate that Bayesian network models that take advantage of prior knowledge in the form of parameter constraints outperform similar models which choose to ignore this kind of knowledge.

5.1 Synthetic Data - Estimating Parameters of a Discrete Variable

This section describes experiments involving one of the simplest forms of parameter constraint: parameter sharing within one distribution, presented in subsection 4.1. The purpose of these experiments is purely demonstrative and a more complicated scenario, on real world data, will be presented in subsection 5.2.

5.1.1 EXPERIMENTAL SETUP

Here, our task is to estimate the set of parameters of a Bayesian network which consists of one discrete variable X. We assume that prior knowledge is available that the distribution of X shares certain parameters. Without loss of generality, we consider that the parameter constraint states that

the parameters to estimate are given by $\theta = \{\theta_1, \ldots, \theta_n\}$ where θ_i appears in $k_i \geq 1$ known places in the distribution of X.

Our synthetic data set was created as follows: first, we randomly generated a distribution T (the "true distribution") that exhibits parameter sharing. This distribution described a variable X with 50 values, which had a total of roughly 50% shared parameters i.e. $\sum_{k_i>1} k_i \approx \sum_{k_i=1} k_i$. Each distinct parameter appeared at most 5 times. We start with an empty distribution and generate a uniformly random parameter v between 0 and 1. Then we generate a random integer s between 2 and 5 and share v in the first s places of the distribution. We continue to generate shared parameters until we reach 25 (50% of 50 parameters). After that, we generate the rest of parameters uniformly randomly between 0 and 1. After all 50 parameters are obtained using this procedure, we normalize to yield a valid probability distribution. Once this distribution was generated, we sampled it to obtain a data set of 1000 examples which were used subsequently to perform parameter estimation.

In our experiments we compare two models that estimate the parameters of distribution T over X. One is a standard Bayesian network (STBN) that is learned using standard Bayesian networks maximum likelihood estimators with no parameter sharing. The second model (PDKBN) is a Bayesian network that is learned using the results in 4.1 assuming the correct parameter sharing was specified by an oracle. While STBN needs to estimate $\sum_{i=1}^{n} k_i$ parameters, PDKBN only needs to estimate n parameters. To deal with potentially zero observed counts, we used priors on the parameters of the two models and then performed maximum aposteriori estimation. For STBN we introduced a Dirichlet count of 2 for each parameter while for PDKBN we used a constrained Dirichlet count of $k_i + 1$ for each distinct parameter θ_i in the network. The role of these priors is simply to assure strictly positive counts.

5.1.2 RESULTS AND DISCUSSION

We performed parameter estimation for the STBN and PDKBN models, varying the number of examples in the training set from 1 to 1000. Since we were using synthetic data, we were able to assess performance by computing KL(T,STBN) and KL(T,PDKBN), the KL divergence from the true distribution T.

Figure 2 shows a graphical comparison of the performance of the two models. It can be seen that our model (PDKBN) that takes advantage of parameter constraints consistently outperforms the standard Bayesian network model which does not employ such constraints. The difference between the two models is greatest when the training data is most sparse. The highest observed difference between KL(T,STBN) and KL(T,PDKBN) was 0.05, which was observed when the two models were trained using 30 examples. As expected, when the amount of training data increases, the difference in performance between the two models decreases dramatically.

Training Examples	KL(T,PDKBN)	Examples needed by STBN
5	0.191	16
40	0.094	103
200	0.034	516
600	0.018	905
650	0.017	> 1000

Table 1: Equivalent training set size so that STBN achieves the same performance as PDKBN.

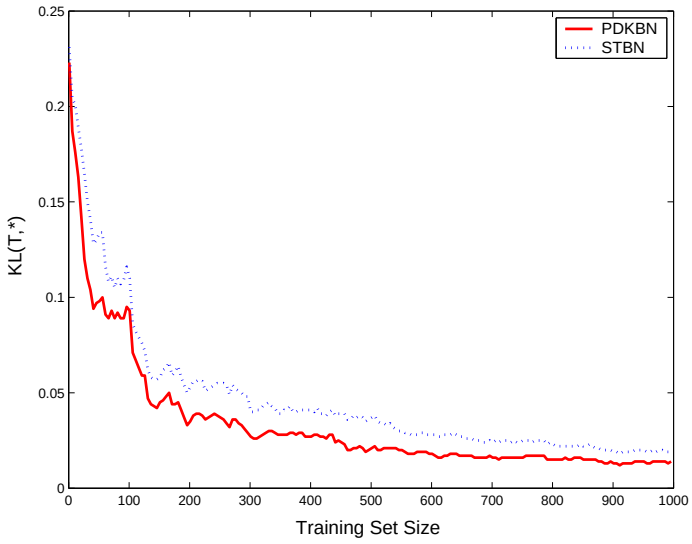


Figure 2: KL divergence of PDKBN and STBN with respect to correct model T.

To get a better idea of how beneficial the prior knowledge in these parameter constraints can be in this case, let us examine *"how far STBN is behind PDKBN"*. For a model PDKBN learned from a data set of a given size, this can be measured by the number of examples that STBN requires in order to achieve the same performance. Table 1 provides these numbers for several training set sizes for PDKBN. For example, STBN uses 16 examples to achieve the same KL divergence as PDKBN at 5 examples, which is a factor of 3.2 (the maximum observed) increase in the number of training samples required by STNB. On average, STBN needs 1.86 times more examples to perform as well as PDKBN.

As mentioned previously, this subsection was intended to be only a proof of concept. We next provide experimental results on a very complex task involving several thousand random variables and prior knowledge in the form of parameter constraints across many conditional probability distributions.

5.2 Real World Data - fMRI Experiments

As noted earlier, functional magnetic resonance imaging (fMRI) is a technique for obtaining three-dimensional images of activity in the brain, over time. Typically, there are ten to fifteen thousand voxels (three dimensional pixels) in each image, where each voxel covers a few tens of millimeters of brain tissue. Due to the nature of the signal, the fMRI activation observable due to neural activity extends for approximately 10 seconds after the neural activity, resulting in a temporally blurred response (see Mitchell et al. (2004) for a brief overview of machine learning approaches to fMRI analysis).

This section presents a generative model of the activity in the brain while a human subject performs a cognitive task, based on the hidden process model and parameter sharing approach discussed in section 4.2. In this experiment involving real fMRI data and a complex cognitive task, domain experts were unable to provide parameter sharing assumptions in advance. Therefore, we

have developed an algorithm to automatically discover clusters of voxels that can be more accurately learned with shared parameters. This section describes the algorithm for discovering these parameter sharing constraints, and shows that training under these parameter constraints leads to hidden process models that far outperform the baseline hidden process models learned in the absence of such parameter constraints.

5.2.1 EXPERIMENTAL SETUP

The experiments reported here are based on fMRI data collected in a study of sentence and picture comprehension (Carpenter et al., 1999). Subjects in this study were presented with a sequence of 40 trials. In 20 of these trials, the subject was first presented with a sentence for 4 seconds, such as "The plus sign is above the star sign.", then a blank screen for 4 seconds, and finally a picture such as

$$\frac{+}{*}$$

for another 4 seconds. During each trial, the subject was required to press a "yes" or "no" button to indicate whether the sentence correctly described the picture. During the remaining 20 trials the picture was presented first and the sentence presented second, using the same timing.

In this data set, the voxels were grouped into 24 anatomically defined spatial regions of interest (ROIs), each voxel having a resolution of 3 by 3 by 5 millimeters. An image of the brain was taken every half second. For each trial, we considered only the first 32 images (16 seconds) of brain activity. The results reported in this section are based on data from a single human subject (04847). For this particular subject, our data set tracked the activity of 4698 different voxels.

We model the activity in each voxel by a hidden process model with two processes, corresponding to the cognitive processes of comprehending a *Sentence* or a *Picture*. The start time of each processes is assumed to be known in advance (i.e., we assume the process begins immediately upon seeing the sentence or picture stimulus). We further assume that the activity in different voxels is independent given the hidden processes corresponding to these voxels. Since the true underlying distribution of voxel activation is not known, we use the average log-likelihood score (the log-likelihood of the test data divided by the number of test examples) to assess performance of the trained HPMs. Because data is scarce, we can not afford to keep a large held-out test set. Instead, we employ a leave-two-out cross-validation approach to estimate the performance of our models.

In our experiments we compare three HPM models. The first model *StHPM*, which we consider a baseline, consists of a standard hidden process model learned independently for each voxel. The second model *ShHPM* is a hidden process model, shared for all the voxels within an ROI. In other words, all voxels in a specific ROI share the same shape hidden processes, but with different amplitudes (see Section 4.2 for more details). *ShHPM* is learned using Algorithm 2. The third model (*HieHPM*) also learns a set of shared hidden process models, but instead of assuming a priori that a particular set of voxels should be grouped together, it chooses these voxel groupings itself, using a nested cross-validation hierarchical approach to both come up with a partition of the voxels in clusters that form a shared hidden process model. The algorithm is as follows:

Algorithm 3 (Hierarchical Partitioning and Hidden Process Models learning)

STEP 1. *Split the 40 examples into a set containing 20 folds $F = \{F_1, \ldots, F_{20}\}$, each fold containing one example where the sentence is presented first and one example where the picture is presented first.*

STEP 2. *For all $1 \leq k \leq 20$, keep fold F_k aside and learn a model from the remaining folds using Steps 3-5.*

STEP 3. *Start with a partition of all voxels in the brain by their ROIs and mark all subsets as* Not Final.

STEP 4. *While there are subsets in the partition that are* Not Final, *take any such subset and try to split it using equally spaced hyperplanes in all three directions (in our experiments we split each subset into 4 (2 by 2) smaller subsets. If the cross-validation average log score of the model learned from these new subsets using Algorithm 2 (based on folds $F \setminus F_k$) is lower than the cross-validation average log score of the initial subset for folds in $F \setminus F_k$, then mark the initial subset as* Final *and discard its subsets. Otherwise remove the initial subset from the partition and replace it with its subsets which then mark as* Not Final.

STEP 5. *Given the partition computed by STEPS 3 and 4, based on the 38 data points in $F \setminus F_k$, learn a hidden process model that is shared for all voxels inside each subset of the partition. Use this model to compute the log score for the examples/trials in F_k.*

STEP 6. *In Steps 2-4 we came up with a partition for each fold F_k. To come up with one single model, compute a partition using STEPS 3 and 4 based on all 20 folds, then, based on this partition learn a model as in STEP 5 using all 40 examples. The average log score of this last model can be estimated by averaging the numbers obtained in STEP 5.*

5.2.2 RESULTS AND DISCUSSION

We estimated the performance of our three models using the average log score, based on a leave two out cross-validation approach, where each fold contains one example in which the sentence is presented first, and one example in which the picture is presented first.

Our first set of experiments, summarized in Table 2, compared the three models based on their performance in the visual cortex (CALC). This is one of the ROIs actively involved in this cognitive task and it contains 318 voxels. The training set size was varied from 6 examples to all 40 examples, in multiples of two. Sharing the parameters of hidden process models proved very beneficial and the impact was observed best when the training set size was the smallest. With an increase in the number of examples, the performance of *ShHPM* starts to degrade because it makes the biased assumption that all voxels in CALC can be described by a single shared hidden process model. While this assumption paid off with small training set size because of the reduction in variance, it definitely hurt in terms of bias with larger sample size. Even though the bias was obvious in CALC, we will see in other experiments that in certain ROIs, this assumption holds and in those cases the gains in performance may be quite large.

As expected, the hierarchical model *HieHPM* performed better than both *StHPM* and *ShHPM* because it takes advantage of shared hidden process models while not making the restrictive as-

Training Trials	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells (HieHPM)
6	-30497	-24020	-24020	1
8	-26631	-23983	-23983	1
10	-25548	-24018	-24018	1
12	-25085	-24079	-24084	1
14	-24817	-24172	-24081	21
16	-24658	-24287	-24048	36
18	-24554	-24329	-24061	37
20	-24474	-24359	-24073	37
22	-24393	-24365	-24062	38
24	-24326	-24351	-24047	40
26	-24268	-24337	-24032	44
28	-24212	-24307	-24012	50
30	-24164	-24274	-23984	60
32	-24121	-24246	-23958	58
34	-24097	-24237	-23952	61
36	-24063	-24207	-23931	59
38	-24035	-24188	-23921	59
40	-24024	-24182	-23918	59

Table 2: The effect of training set size on the average log score of the three models in the visual cortex (CALC) region.

sumption of sharing across entire ROIs. The largest difference in performance between *HieHPM* and *StHPM* is observed at 6 examples, in which case *StHPM* basically fails to learn a reasonable model while the highest difference between *HieHPM* and *ShHPM* occurs at the maximum number of examples, presumably when the bias of *ShHPM* is most harmful. As the number of training examples increases, both *StHPM* and *HieHPM* tend to perform better and better and one can see that the marginal improvement in performance obtained by the addition of two new examples tends to shrink as both models approach convergence. While with an infinite amount of data, one would expect *StHPM* and *HieHPM* to converge to the true model, at 40 examples, *HieHPM* still outperforms the baseline model *StHPM* by a difference of 106 in terms of average log score, which is an improvement of e^{106} in terms of data likelihood.

Probably the measure that shows best the improvement of *HieHPM* over the baseline *StHPM* is the number of examples needed by *StHPM* to achieve the same performance as *HieHPM*. It turns out that on average, *StHPM* needs roughly 2.9 times the number of examples needed by *HieHPM* in order to achieve the same level of performance in the visual cortex (CALC).

The last column of Table 2 displays the number of clusters of voxels in which *HieHPM* partitioned CALC. As can be seen, at small sample size *HieHPM* draws its performance from reductions in variance by using only one cluster of voxels. However, as the number of examples increases, *HieHPM* improves by finding more and more refined partitions. This number of shared voxel sets tends to stabilize around 60 clusters once the number of examples reaches 30, which yields an av-

erage of more than 5 voxels per cluster given that CALC is made of 318 voxels. For a training set of 40 examples, the largest cluster has 41 voxels while many clusters consist of only one voxel.

ROI	Voxels	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells Hierarchical
CALC	318	-24024	-24182	-23918	59
LDLPFC	440	-32918	-32876	-32694	11
LFEF	109	-8346	-8299	-8281	6
LIPL	134	-9889	-9820	-9820	1
LIPS	236	-17305	-17187	-17180	8
LIT	287	-21545	-21387	-21387	1
LOPER	169	-12959	-12909	-12909	1
LPPREC	153	-11246	-11145	-11145	1
LSGA	6	-441	-441	-441	1
LSPL	308	-22637	-22735	-22516	4
LT	305	-22365	-22547	-22408	18
LTRIA	113	-8436	-8385	-8385	1
RDLPFC	349	-26390	-26401	-26272	40
RFEF	68	-5258	-5223	-5223	1
RIPL	92	-7311	-7315	-7296	11
RIPS	166	-12559	-12543	-12522	20
RIT	278	-21707	-21720	-21619	42
ROPER	181	-13661	-13584	-13584	1
RPPREC	144	-10623	-10558	-10560	1
RSGA	34	-2658	-2654	-2654	1
RSPL	252	-18572	-18511	-18434	35
RT	284	-21322	-21349	-21226	24
RTRIA	57	-4230	-4208	-4208	1
SMA	215	-15830	-15788	-15757	10
All Brain	4698	-352234	-351770	-350441	299

Table 3: Per ROI performance (average log score) of the three models when learned using all 40 examples.

The second set of experiments (see Table 3) describes the performance of the three models for each of the 24 individual ROIs of the brain, and trained over the entire brain. While we have seen that *ShHPM* was biased in CALC, we see here that there are several ROIs where it makes sense to characterize all of its voxels by a single shared hidden process model. In fact, in most of these regions, *HieHPM* finds only one cluster of voxels. Actually, *ShHPM* outperforms the baseline model *StHPM* in 18 out of 24 ROIs while *HieHPM* outperforms *StHPM* in 23 ROIs. One may ask how *StHPM* can possibly outperform *HieHPM* on a ROI, since *HieHPM* may also represent the case when there is no sharing. The explanation is that the hierarchical approach can get stuck in a local maximum of the data log-likelihood over the search space if it cannot improve by splitting at

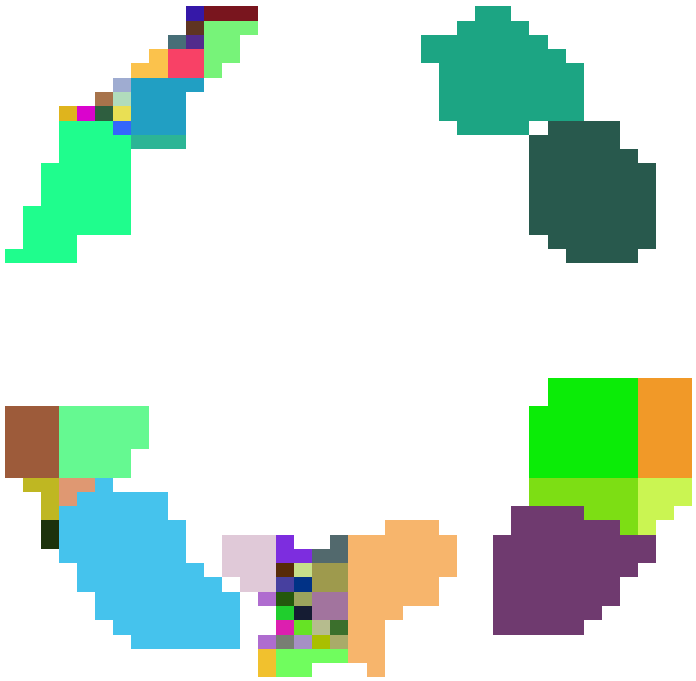


Figure 3: Parameter sharing found using model *HieHPM*. Slice five of the brain is showed here. Shared neighboring voxels have the same color.

a specific step, since it is a greedy process that does not look beyond that split for a finer grained partition. Fortunately, this problem appears to be rare in these experiments.

Over the whole brain, *HieHPM* outperforms *StHPM* by a factor 1792 in terms of log likelihood while *ShHPM* outperforms *StHPM* only by a factor of 464. The main drawback of the *ShHPM* is that it also makes a very restrictive sharing assumption and therefore we suggest *HieHPM* as the recommended approach. Next we give the reader a feel of what the learned *HieHPM* model looks like.

As mentioned above, *HieHPM* automatically learns clusters of voxels that can be represented using a shared hidden process model. Figure 3 shows the portions of these learned clusters in slice five of the eight vertical slices that make up the 3D brain image captured by the fMRI scanner. Neighboring voxels that were assigned by *HieHPM* to the same cluster are pictured with the same color. Note that there are several very large clusters in this picture. This may be because of the fact that it makes sense to represent an entire ROI using a single shared hidden process model if the cognitive process does not activate voxels in this ROI. However, large clusters are also found in areas like CALC, which we know is directly involved in visual processing.

In Figure 4 we can see the learned *Sentence* hidden process for the voxels in the visual cortex (CALC). Again, the graphs corresponding to voxels that belong to the same cluster have been painted in the same color, which is also the same as the color used in Figure 3. To make these graphs readable, we only plotted the base process, disregarding the scaling (amplitude) constants corresponding to each voxel within a given cluster (consult Section 4.2 for more details about shared

hidden process models).

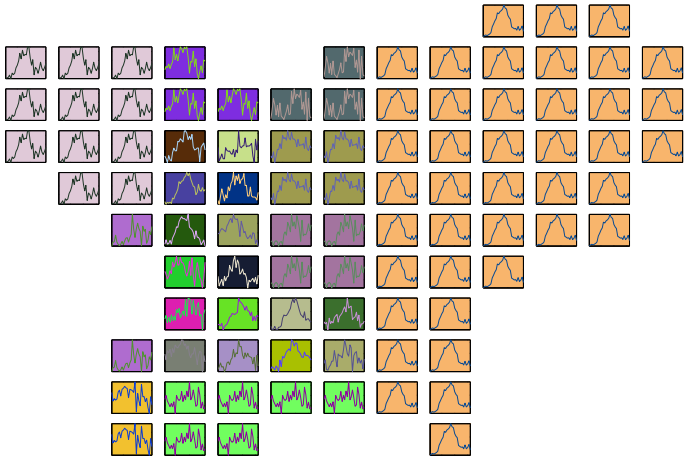


Figure 4: Per voxel base *Sentence* processes in the visual cortex (CALC).

To summarize, this subsection presented experiments training different generative models for the fMRI signal during a cognitive task, all based on hidden Process models. We demonstrated experimentally that parameter sharing for hidden Process models (as defined in Section 4.2) can greatly benefit learning, and that it is possible to automatically discover useful parameter sharing constraints in this domain using our hierarchical partitioning algorithm.

6. Formal Guarantees

Taking advantage of parameter constraints can be beneficial to learning because, intuitively, it has the effect of lowering the variance in parameter estimators by shrinking the degrees of freedom of the model. In this section we provide a formal proof of this fact. In order for our proof to work, we make the assumption that the true distribution factors according to the given Bayesian network structure and that it obeys the parameter constraints provided by the expert. The second interesting result presented in this section will give theoretical guarantees in the case when the constraints provided by the expert are not entirely accurate. While we only investigate this issue for one type of constraint, parameter sharing within one distribution (introduced in subsection 4.1), we believe similar formal guarantees describe all other types of parameter constraints presented in this paper.

6.1 Variance Reduction by Using Parameter Constraints

Assume we want to learn a Bayesian network in the case when a domain expert provides parameter constraints specifying that certain parameters appear multiple times (are shared) within a conditional probability distribution. Each conditional probability distribution in the Bayesian network can have its own such constraints. Also, the case when all parameters are distinct within one such distribution may be seen as a particular case of parameter sharing within one distribution, where each parameter is shared exactly once.

There are two ways to perform maximum likelihood parameter learning in the above Bayesian network. First, one may choose to ignore the constraints given by the expert and compute standard maximum likelihood estimators. A second option is to incorporate the constraints in the learning method, in which case we can use the results described in subsection 4.1. One would intuitively expect that taking advantage of the constraints provided by the expert would reduce the variance in parameter estimates when compared to the first approach. In Niculescu (2005) we prove the following result:

Theorem 2 *Assuming a domain expert can specify parameter sharing assumptions that take place inside the conditional probability distributions of a Bayesian network, the maximum likelihood estimators that use this domain knowledge as computed with Theorem 1 have lower variance than standard maximum likelihood estimators computed ignoring the domain knowledge. More specifically, for one parameter θ_{ijk} that is shared $s \geq 1$ times within $P(X_i|PA_i = pa_{ik})$, denote by $\hat{\theta}^{ML}_{ijk}$ the maximum likelihood estimator that ignores domain knowledge and by $\hat{\theta}^{PS}_{ijk}$ the maximum likelihood estimator that uses the parameter sharing assumptions specified by the expert. We have the following identity:*

$$Var[\hat{\theta}^{ML}_{ijk}] - Var[\hat{\theta}^{PS}_{ijk}] = \theta_{ijk} \cdot (1 - \frac{1}{s}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0] \geq 0$$

6.2 Performance with Potentially Inaccurate Constraints

Sometimes it may happen that the parameter constraints provided by an expert are not completely accurate. In all our methods so far, we assumed that the parameter constraints are correct and therefore errors in domain knowledge can prove detrimental to the performance of our learned models. In this section we investigate the relationship between the true, underlying distribution of the observed data and the distribution estimated using our methods based on parameter constraints. In particular, we come up with an upper bound on how well our estimated model can perform given a set of potentially incorrect parameter constraints.

Assume an expert provides a set of potentially incorrect parameter sharing assumptions as described in subsection 4.1. In other words, for each conditional probability distribution c in the Bayesian network, the expert is stating that parameter θ_{ic} is shared in k_{ic} given positions. We denote by N_{ic} the cumulative observed count corresponding to the presumably shared parameter θ_{ic} and by N_c the cumulative observed count corresponding to the conditional distribution c. Essentially, we follow the notations in subsection 4.1, to which we add an additional index corresponding to the conditional probability distribution that a parameter belongs to.

Let us introduce the notion of *true probabilistic counts (TPC)*. Suppose P is the true distribution from which data is sampled. If, for example, the expert states that θ_{ic} is the shared parameter that describes the set $\{P(X = x_1|PA(X) = pa), \ldots, P(X = x_{k_{ic}}|PA(X) = pa)\}$, let $TPC_{ic} = \sum_{i=1}^{k_{ic}} P(X = x_i, PA(X) = pa)$. Let P^* be the distribution that factorizes according to the structure provided by the expert and has parameters given by theorem 1 where the observed counts are replaced by the *true probabilistic counts*.

Theorem 3 *P^* is the closest distribution to P (in terms of $KL(P, \cdot)$) that factorizes according to the given structure and obeys the expert's parameter sharing assumptions.*

Proof Let Q be such a distribution. Minimizing $K(P,Q)$ is equivalent to maximizing $\sum_d P(d) \cdot \log Q(d)$. Let θ be the set of parameters that describe this distribution Q. After breaking the logarithms into sums of logarithms based on the factorization given by the provided structure, our optimization problem reduces to the maximization of $\sum TPC_{ic} \cdot \log \theta_{ic}$. This is exactly the objective function used in theorem 1. This is equivalent to the fact that P^*(see the definition above) minimizes $KL(P,\cdot)$ out of all the distributions that factorize according to the given structure and obey the expert's sharing assumptions. ■

Theorem 4 *With an infinite amount of data, the distribution $\hat{P}$ given by the maximum likelihood estimators in Theorem 1 converges to P^* with probability 1.*

Proof Assume the number of data points in a data set sampled from P is denoted by n. According to the law of large numbers, we have $lim_{n\to\infty} \frac{N_{ic}}{n} = TPC_{ic}$. This implies that $\hat{P}$ converges to P^* with probability 1. ■

Corollary 5 *If the true distribution P factorizes according to the given structure and if the parameter sharing provided by the expert is completely accurate, then the distribution $\hat{P}$ given by the estimators computed in Theorem 1 converges to P with probability 1.*

Again, we mention that we analyzed the formal guarantees presented in this section using only one type of parameter constraints. We are confident that these results can be extended to all other types of constraints for which we computed closed form solutions.

7. Conclusions and Future Work

Building accurate models from limited training data is possible only by using some form of prior knowledge to augment the data. In this paper we have demonstrated both theoretically and experimentally that the standard methods for parameter estimation in Bayesian networks can be naturally extended to accommodate parameter constraints capable of expressing a wide variety of prior domain knowledge.

We mentioned our previous work on methods for incorporating general parameter constraints into estimators for the parameters of a Bayesian network, by framing this task as a constrained optimization problem. In the general case, solving the resulting optimization problem may be very difficult. Fortunately, in practice the optimization problem can often be decomposed into a set of many smaller, independent, optimization subproblems. We have presented parameter estimators for several types of constraints, including constraints that force various types of parameter sharing, and constraints on sums and other relationships among groups of parameters. Subsection 4.3 provides a comprehensive list of the parameter constraint types we have studied, along with brief examples of each. We considered learning with both discrete and continuous variables, in the presence of both equality and inequality constraints. While for most of these types of parameter constraints we can derive closed form maximum likelihood estimators, we developed a very efficient iterative algorithm to perform the same task for shared hidden process models. In many of these cases, for discrete variables, we are also able to compute closed form normalization constants for the corresponding

constrained parameter priors, allowing one to perform closed form MAP and Bayesian estimation when the data is complete.

The general parameter sharing domain knowledge type (Constraint Type 6 defined in subsection 4.3) encompasses models including HMMs, dynamic Bayesian networks, module networks and context specific independence as particular cases, but allows for much finer grained sharing, at the parameter level, across different variables and across distributions of different lengths. It is also important to note that one can combine different types of parameter constraints when learning the parameters of a Bayesian network as long as the scopes of these constraints do not overlap.

Experimental results using an fMRI brain imaging application demonstrate that taking advantage of parameter constraints can be very beneficial for learning in this high-dimensional, sparse-data domain. In the context of this application we developed methods to automatically discover parameter sharing constraints. Using these methods our program discovered clusters of voxels whose parameters can be shared. Our results showed that the impact of these learned parameter constraints can be equivalent to almost tripling the size of the training set on this task. Experiments on synthetic data demonstrated the same beneficial effect of incorporating parameter constraints.

A basic theoretical result is that the estimators taking advantage of a simple form of parameter sharing achieve variance lower than that achieved by estimators that ignore such constraints. We conjecture that similar results hold for other types of parameter constraints, but their proof is left as future work. In addition, we proved that even when the asserted parameter constraints turn out to be incorrect, given an infinite amount of training data our maximum likelihood estimators converge to the best describable distribution; that is, the distribution closest in terms of KL distance from the true distribution, among all distributions that obey the parameter constraints and factor according to the given structure.

We see many useful directions for future work. In this paper we have considered only how to take advantage of deterministic parameter constraints when the structure of the Bayesian network is known in advance. It would be interesting to investigate methods to incorporate probabilistic constraints in learning algorithms for Bayesian networks. A second direction to explore is to also use parameter constraints to perform structure learning. This might be achieved by specifying an initial set of parameter constraints, then at each step of the hill climbing during structure search performing a change of variable to adapt the constraints to the new parameterization of the network. Finally, we would like to extend our results to undirected graphical models, to the extent that it is intuitive to acquire domain knowledge from an expert about the much harder to interpret parameters of such models.

Acknowledgments

We would like to thank the following people for their useful comments and suggestions during the development of this research: John Lafferty, Andrew Moore, Russ Greiner, Zoubin Ghahramani, Sathyakama Sandilya and Balaji Krishnapuram. As a student at Carnegie Mellon University, Radu Stefan Niculescu was sponsored the National Science Foundation under grant nos. CCR-0085982 and CCR-0122581, by the Darpa PAL program under contract NBCD030010, and by a generous gift from Siemens Medical Solutions.

References

J. Bilmes. Dynamic bayesian multinets. In *Proceedings of UAI*, pages 38–45, 2000.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *Proceedings of 12th UAI*, pages 115–123, 1996.

P. A. Carpenter, M. A. Just, T. A. Keller, W. F. Eddy, and K. R. Thulborn. Time course of fMRI-activation in language and spatial networks during sentence comprehension. *NeuroImage*, 10: 216–224, 1999.

A. M. Dale. Optimal experimental design for event-related fMRI. *Human Brain Mapping*, 8:109–114, 1999.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of 16th IJCAI*, pages 1300–1307, 1999.

D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

D. Geiger and D. Heckerman. A characterization of the dirichlet distribution through global and local parameter independence. *The Annals of Statistics*, 25:1344–1369, 1997.

D. Heckerman. A tutorial on learning with bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

P. Hooper. Dependent dirichlet priors and optimal linear estimators for belief net parameters. In AUAI Press, editor, *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 251–259, 2004.

R. Hutchinson, T.M. Mitchell, and I. Rustandi. Learning to identify overlapping and hidden cognitive processes from fMRI data. In *11th Conference on Human Brain Mapping*, June 2005.

R. Hutchinson, T.M. Mitchell, and I. Rustandi. Hidden process models. Technical Report CS-CALD-05-116, Carnegie Mellon University, February 2006.

D. Koller and A. Pfeffer. Object oriented bayesian networks. In *Proceedings of 13th UAI*, pages 302–313, 1997.

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

T. Minka. The dirichlet-tree distribution. This unpublished paper is available online at http://research.microsoft.com/∼minka/papers/dirichlet/minka-dirtree.pdf, 1999.

T. Mitchell, R. Hutchinson, M. Just, S. Newman, R. S. Niculescu, F. Pereira, and X. Wang. Learning to decode cognitive states from brain images. *Machine Learning*, 57(1-2):145–175, 2004.

K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.

R. S. Niculescu. Exploiting parameter domain knowledge for learning in bayesian networks. Technical Report CMU-TR-05-147, Carnegie Mellon University, 2005.

R. S. Niculescu, T. Mitchell, and R. B. Rao. Parameter related domain knowledge for learning in graphical models. In *Proceedings of SIAM Data Mining conference*, 2005.

J. M. Pena, J. A. Lozano, and P. Larranaga. Learning recursive bayesian multinets for data clustering by means of constructive induction. *Machine Learning*, 47(1):63–89, 2002.

W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

R. L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

R. B. Rao, S. Sandilya, R. S. Niculescu, C. Germond, and H. Rao. Clinical and financial outcomes analysis with existing hospital patient records. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 416–425, 2003.

E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. In *Proceedings of 19th UAI*, pages 525–534, 2003.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina, 1995.

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*, pages 334–342, 2001.