OPTIMIZATION FOR SPARSE AND ACCURATE CLASSIFIERS

BY NOAM GOLDBERG

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Professor Jonathan Eckstein

and approved by

_____

_____

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2010

ABSTRACT OF THE DISSERTATION

Optimization for sparse and accurate classifiers

by Noam Goldberg

Dissertation Director: Professor Jonathan Eckstein

Classification and supervised learning problems in general aim to choose a function that best describes a relation between a set of observed attributes and their corresponding outputs. We focus on binary classification, where the output is a binary response variable. In this dissertation, we seek motivation within statistical learning theory, which attempts to estimate how well a classification function generalizes with respect to unseen data in a probabilistic setting.

We study linear programming formulations for finding a hyperplane that separates two sets of points. Such formulations were initially given by Mangasarian [52] for the separable case, and more recently extended by "soft margin" formulations that maximize the margin of separation subject to a penalty proportional to the sum of margin violations. LP-Boost is a boosting algorithm which solves the large scale linear program in a high dimensional space of features using a column generation solution technique.

While many authors have developed different boosting algorithms that assume the existence of effective base learning algorithms for generating a feature or base classifier (*e.g.*, solving the pricing problem with a column generation approach), there has not been much work done to propose such algorithms. In this dissertation, we propose a branch-and-bound algorithm for finding a Boolean monomial that best agrees with the given set of data. This problem has been known as *maximum monomial agreement* and has been mostly studied

in the computational complexity and learning communities in the context of negative computational complexity results. Here we propose an algorithm that finds the monomial that best agrees with the given data and show that it is computationally efficient in practice for UCI datasets.

Revisiting the problem of finding weighted voting classifiers, we consider the problem of balancing the sparsity of the vector of weights and accuracy with respect to the given the training data. It has been suggested by several authors to minimize the L_1-norm of the normal vector of the separating hyperplane in order to find sparse solutions. In contrast, we formulate the discrete optimization problem of minimizing the sum of disagreements of a weighted voting classifier and a penalty proportional to the number of nonzero components of the hyperplane's normal vector. The problem that we propose extends the $\mathcal{NP}$-hard Minimum Disagreement Halfspace problem studied in computational learning theory. We are able to formulate this problem as a Mixed Integer Linear Program (MILP), and show that the continuous relaxation of the MILP is equivalent to the well-known L_1-norm minimizing LP for finding weighted voting classifiers. We proceed by suggesting novel cutting planes to tighten the continuous relaxation. Finally, we propose and test a novel boosting algorithm that solves the relaxation by dynamic generation of columns and cuts. The algorithm used for generating the columns in this procedure generalizes our algorithm for maximum monomial agreement. In our experiments with the novel MILP relaxation, we demonstrate that our formulation, as well as solution technique, provide an effective approach for constructing sparse and accurate classifiers, and balancing the tradeoff between the two.

Acknowledgements

I would like to thank my advisor, Professor Jonathan Eckstein; without his guidance, this work would not have been possible. I enjoyed our conversation and the ideas that came up during and following these conversations. I would also like thank the members of the committee who I had the opportunity to consult with about problems discussed within this dissertation as well as other research problems.

During the course of my PhD studies I have spoken to several Professors, who I also looked up to as mentors. I have not always persisted and followed through with results, nevertheless, I believe that these interactions helped me to develop my "identity" as a researcher. I need to thank Professor Peter Hammer with whom I had the honor to discuss research possibilities before he passed away suddenly. I would like to thank Professor Chung-chieh (Ken) Shan for helping me to get started on a path to doing good research, including work that led to this dissertation. I would like to thank Professor Robert Schapire for an inspiring talk at a DIMACS conference that motivated me to try to address the machine learning and statistical learning communities at large, at an early stage of this work, and for discussing the machine learning aspects of this dissertation. I also thank Professors Endre Boros and Paul Kantor who funded and guided me in their Homeland Security funded research, which I believe helped me to further evolve and improve my skills as a researcher. I would like to thank Professor Fred Roberts for finding time to discuss an interesting research problem in voting theory, and also to invite me to get involved in the DyDAn center. I have also enjoyed speaking on occasion to Professors Michael Grigoriadis, András Prékopa, and Dan Stratila about different research projects or ideas.

I would like to thank Professor Kristin Bennett for general comments, and Martin Milanič for comments about a feature selection problem, both related to work done in Chapter 3 of this dissertation. I am also grateful for being able to use the computing resources

of the computer science department, which was instrumental in the computational work of Chapter 3. Finally, the dissertation is based upon work partially supported by DyDAn and the U.S. Department of Homeland Security under Grant Award Number 2008-DN-077-ARI001-02.

I would also like to thank Tami Carpenter, Paul Kantor, Tamara G. Kolda, Hanan Luss and Fred Roberts for career planning and job search advice.

At a personal level first and foremost I need to thank my wife and best friend Michal, for her love and support during the ups and downs of the past five years. I also need to thank friends at RUTCOR that helped with some good laughs and interesting conversation, to name a few: Ricardo Collado, Aritanan Gruber, Martin Milanič, Olga Myndyuk, Mathew Oster, David Papp, and Anupama Reddy.

Dedication

To my grandparents.

Table of Contents

Chapter 1

Introduction

This dissertation explores new formulations and algorithms for selecting data classification models. A classification model is a function that maps from some given domain of observed attributes to a predicted class of interest. We will assume here that there are only two classes in our data: a "positive" class and a "negative" one. In a medical diagnosis application the class may be either "ill" or "healthy". The data may include attributes of patients such as weight and blood pressure. Based on a set of observed attribute values of a newly examined patient, we may wish to predict whether the patient has heart disease or is healthy. We note, however, that the case of more than two classes has been reduced to one or more problems of binary classification, with the error rate of the multi-class problem bounded in terms of the error of the binary problem [1, 31].

By optimally *sparse* and accurate classifiers we mean to consider the problem of how to strike an ideal balance between sparsity and accuracy of a classifier on the given set of data. In this dissertation, we consider *weighted voting classifiers* that are linear combinations of the outputs of simple classifiers. The classification is made based on comparing the linear combination to a threshold which we may assume to be zero by including a constant base classifier, without loss of generality. The sparsity of a weighted voting classifier can be defined as its proportion of zero weighted base classifiers. The motivation for sparsity, which will be elaborated on below, is to avoid *overfitting*; selecting classifiers which perform poorly on new data but are highly accurate with respect to the given training data. This phenomenon is even more likely to occur when the samples contain noise or outliers that may cause us to select highly complex models that fail to apply to the "true" data. Generally speaking, sparse classification models tend to correspond to hyperplanes that do not necessarily separate all of the positive points from the negative ones, but are less likely to

overfit than complex models. Finally, a relatively sparse model's compact set of rules may be easier to interpret and comprehend.

Classification problems and weighted voting classifiers have been investigated in several fields and communities. In statistics, classification problems and specifically the problem of selecting weighted voting classifiers, traditionally have been solved by logistic regression. Logistic regression involves applying a transformation to the binary response variable in order to apply general linear model techniques. These techniques are not explored further in this dissertation, but the reader may refer to Cox and Snell [24] for more detail. In the field of operations research, linear and nonlinear programs have been suggested for finding separating hyperplanes for data classification applications by Mangasarian [52, 53]. Bennett and Mangasarian have also suggested robust linear programming formulations for data that may be non-separable [12]. Hammer's *logical Analysis of data* (LAD) [25, 15, 16] applies the theory of Boolean functions to data analysis. LAD consists of a systematic approach to enumerating monomials (or interactions of variables) and using them to model cause-and-effect relationships [25].

In machine learning, and now statistical learning, which lies in the intersection of machine learning, statistics and optimization, weighted voting classifiers have been an especially active area of research. Boosting was first considered in the theoretic *provably approximately correct* (PAC) learning framework by Kearns and Valiant [46, 47], and the first algorithms were suggested by Schapire [61], and later Freund [34] and Freund and Schapire [36]. Support Vector Machines (SVMs) were first suggested by Vapnik and his colleagues [69], and together with boosting they have been the most widely used algorithms for selecting weighted voting classifiers.

1.1 Weighted voting classification

In *binary classification* problems we are given a set of training data $A \in \mathbb{R}^{M \times N}$, whose rows correspond to observations and whose columns correspond to attributes. We are also given a vector of labels $y \in \{-1, 1\}^M$, defining a partition of the observations into a "positive" class $\Omega^+ = \{i \in \{1, \ldots, M\} \mid y_i = 1\}$, and a "negative" class $\Omega^- = \{i \in \{1, \ldots, M\} \mid$

$y_i = -1\} = \{1, \ldots, M\} \setminus \Omega^+$. Using these input data, we would like to train a *classifier*, $g : \mathbb{R}^N \to [-1, 1]$, in order to classify any $x \in \mathbb{R}^N$ as either positive or negative based on $\mathrm{sgn}(g(x))$.

Suppose we have a potentially large set of base classifiers $h_u : \mathbb{R}^N \to \{-1, 0, 1\}$, indexed by the set $\mathcal{U} = \{1, \ldots, U\}$. A *weighted voting classifier* (with 0 threshold) is a function of the form:

$$g(x) = \sum_{u \in \mathcal{U}} \lambda_u h_u(x). \tag{1.1}$$

Each h_u is a simple base classifier, a map from observations in the original space of attributes to labels assigned in the space of *features* given by the index set $\mathcal{U} = \{1, \ldots, U\}$.

The *margin* of a set of observations $\{1, \ldots, M\}$ is defined as the quantity

$$\min_{i \in \{1, \ldots, M\}} y_i g(A_i).$$

Accordingly, the margin of observation i is just $y_i g(A_i)$.

We refer to a *boosting* algorithm as one that iteratively selects some $u \in \mathcal{U}$ in order to assign a nonzero weight to λ_u, and finally outputs the weighted voting (also known as the *strong*) classifier $g(\cdot)$ that either exactly or approximately maximizes the L_1 margin (equivalently the L_∞ distance to the closest point) [63, 60, 73]. The base classifier h_u, for $u \in \mathcal{U}$, is selected by a "black box" base (also known as *weak*) learning algorithm. A potential difficulty with this approach is that U can be very large and often exponential in N. In fact, as we will see in the following, the base learning problem can be $\mathcal{NP}$-hard for many interesting classes of base classifiers, such as monomials of arbitrary order.

Learning algorithms for classification, in general, have considered a variety of different *loss functions* as performance measures. Common loss functions, with respect to a single observation, defined in the following as $\ell : \mathbb{R}^N \times \{-1, 1\} \times [-1, 1] \to \mathbb{R}$, include the predictive zero-one loss:

$$\ell(A_i, y_i, g(A_i)) = \mathbf{I}(y_i g(A_i) \leq 0), \tag{1.2}$$

where $I(\cdot)$ is the 0/1 indicator function, the accuracy loss,

$$\ell(A_i, y_i, g(A_i)) = |y_i - g(A_i)|/2, \tag{1.3}$$

the exponential loss,

$$\ell(A_i, y_i, g(A_i)) = e^{-y_i g(A_i)}, \tag{1.4}$$

the logistic loss,

$$\ell(A_i, y_i, g(A_i)) = \log(1 + e^{-y_i g(A_i)}), \tag{1.5}$$

or the *soft margin* loss (with margin fixed to 1):

$$\ell(A_i, y_i, g(A_i)) = (1 - y g(A_i))_+, \text{ where } (\cdot)_+ = \max\{\cdot, 0\}. \tag{1.6}$$

The objective for minimization, in order to determine λ, is usually the sum, or equivalently the average, of losses over all observations. The original AdaBoost algorithm by Fruend and Schapire [36] has been shown to minimize the sum of (1.4) over the training observations [21], for certain optimal base learners. Logistic regression, which is a common classification model selection procedure in statistics, minimizes the sum of (1.5).

The problem of minimizing (1.2) has been studied in the computational learning theory community and theoretical computer science. Computational learning theory involves the complexity-theoretic study of key problems in machine and statistical learning, as well as theoretic estimates of the quality of the resulting models. In the following, we are mostly concerned with classification model selection schemes that attempt to minimize the sum of the "hard" zero-one loss (1.2) and its "relaxation", the "soft margin" loss (1.6).

Finally, we are interested in classification models that are robust; although the input data may be noisy, we would like to perform well with respect to the unseen test datasets. The definition of the objective function for finding an optimal classifier g can be addressed within the framework of *regularization theory* or *statistical learning theory* (for an introduction and comparison of both approaches in the context of supervised learning, see Evgeniou *et al.* [32]). Using either theory as motivation, robust optimization-based model selection procedures for classification would minimize an objective function consisting of the sum

of losses plus a regularization term or complexity penalty. In this dissertation, we seek motivation in statistical learning theory, which defines the optimality of a classifier with respect to its ability to "generalize" to unseen data in a probabilistic setting. Statistical learning theory may provide insight into the choice of penalty parameters that otherwise become a matter of trial and error. First, we appeal to an intuitive concept of our preference for simple models, which we will later relate to statistical learning theory and code lengths. Other things being equal, classification models are generally preferred to be sparse and less "complex".

1.2 Sparse models

Sparse linear models are loosely defined as linear functionals such as (1.1), for which the support of λ is small. The zero norm of a vector, which we denote by $||\cdot||_0$, is defined as the size of its support (and is not a proper L_p norm). Meir and Rätsch [54] more specifically define λ to be sparse if $||\lambda||_0 \in O(M)$ (rather than $||\lambda||_0 \in O(U)$). Nevertheless, if our model is to generalize well for unseen data, we expect $||\lambda||_0$ to be sublinear in M.

Sparse classification models are desirable for two main reasons: for computational purposes when U is large, and to prevent overfitting. Overfitting occurs when the classifier is accurate with respect to the input data A but is significantly less accurate with respect to new data to be classified. Occam's Razor principle also suggests choosing the simplest possible model that explains the data. Thus, everything else being the same, according to Occam's principle, we would select models for which $\lambda_u > 0$ for the smallest subset of $\mathcal{U}$.

Occam's principle, however, is quite subjective as a model selection criterion. *Minimum description length* (MDL) is a notion that attempts to quantify the tradeoff between model complexity and accuracy with respect to the input data. MDL treats the problem of model selection as a data compression optimization problem: the model is used to compress the data. The optimal model minimizes the sum of the size of an efficient encoding of the model and the size of encoding the observations which the model misclassifies. Exact MDL is difficult to apply in practice, due to the difficulty of finding and proving the efficiency of encoding schemes for a model. A comprehensive account of MDH, its information theoretic

foundations, and the equivalence of different variants of MDL to well known principles and methods of statistical inference is given by Grünwald [42]. Here we focus on the more intuitive aspects of MDL, and the related compression interpretation of learning within statistical learning theory.

In signal detection and compressed sensing, one faces a related problem of solving an under-determined linear system (see [72, 18]). Optimally sparse solutions correspond to solutions with minimum L_0 norm, defined as the number of nonzero components of the solution vector. Finding a solution with minimum L_0 norm is known to be $\mathcal{NP}$-hard [55]. In practice, the common approach is to minimize the L_1 or L_2 norms. Greedy heuristics for minimizing L_0 have also been suggested by several authors [66]. Natarajan proposes a greedy algorithm with an approximation guarantee in terms of an L_p-norm of the input matrix and right-hand side [55].

Similarly, in classification model selection, overfitting is typically mitigated by assigning a complexity (or density) penalty proportional to various norms of λ. Optimally sparse models would be obtained by directly minimizing or penalizing the L_0 norm of λ. In order to avoid a hard combinatorial optimization problem, the authors of various methods such as LP-Boost, Lasso and Support Vector Machines (SVMs) [26, 37, 23, 11] instead suggest using the L_1 or L_2 norms of λ. Minimizing any one of the common loss functions plus a complexity penalty proportional to an L_1 or L_2 norms of λ has the computational advantage of being a convex optimization problem.

In our approach to model selection, we would like to more directly attack the combinatorial optimization problem of minimizing the predictive loss subject to a weighted L_0 complexity penalty. Penalizing the L_0 norm of λ would help to construct classifiers that are "optimally sparse". We also believe that L_0 penalty parameters are easier to interpret than the penalty parameters applied to other p-norms, in the context of the MDL principle. When our model makes an error, we "transmit" and "pay for" identifying the observation which should be relabeled among the given M samples. Otherwise, in the following, we initially assume that all base classifiers are equally complex, so that the complexity of our model is linear in the number of nonzero coefficients in λ. We then generalize our model by taking into account that some features are more complex than others, and therefore need

to be assigned a greater complexity penalty. We expect the MDL principle to be helpful in choosing penalty parameters that otherwise could only be determined by experiment and cross-validation.

1.3 Risk minimization and risk bounds

Assuming that the data are sampled from an unknown underlying distribution $P(x, y)$, the learning problem is to minimize the *risk*:

$$R[g] = \int_{\mathbb{R}^N \times \{-1,1\}} \ell(x, y, g(x)) dP(x, y).$$

The *empirical risk* is:

$$R_{\text{emp}}[g] = \frac{1}{M} \sum_{i=1}^{M} \ell(A_i, y_i, g(A_i)). \tag{1.7}$$

Empirical risk minimization calls for minimizing (1.7), for some loss function ℓ such as (1.2), which is essentially averaged over all input observations. Empirical risk minimization may result in overfitting: significantly larger losses on the test (unseen) data than occurred on the training data.

Risk bounds are probabilistic statements that bound the risk of a model without making any assumption about the distribution, *i.e.*, without knowledge of P. In order to bound the expected risk of a model that is selected from a set of functions $\mathcal{F} \subset \left(\mathbb{R}^N\right)^{\{-1,1\}}$, statistical learning theory attempts to measure the *capacity* of a set of functions. We describe a number of different measures of the capacity of a set of classification functions. Let $\mathcal{N}(\mathcal{F}, A)$ denote the cardinality of $\mathcal{F}$ when restricted to $\{A_1, \ldots, A_M\}$, that is, the number of functions of $\mathcal{F}$ that can be distinguished by their values on $\{A_1, \ldots, A_M\} \subset \mathbb{R}^N$. Finally, let

$$\mathcal{N}(\mathcal{F}, M) = \max_{A \in \mathbb{R}^{M \times N}} \left| \ \{\{i \in \{1, \ldots, M\} \ \mid \ f(A_i) = 1\} \ \mid \ f \in \mathcal{F}\} \ \right|$$

denote the maximum number of ways functions in $\mathcal{F}$ can partition a set of M observations into two classes. The function $\mathcal{N}(\mathcal{F}, M)$ is known as the *shattering coefficient* of the class of functions $\mathcal{F}$. The logarithm of the expectation of the shattering coefficient over all samples of size M, that is $\ln E[\mathcal{N}(\mathcal{F}, Z_M)]$, is also known as the *annealed entropy*. A risk bound can

be stated with respect to the annealed entropy, specifically that with probability at least $1-\delta$ (see [66]):

$$R[g] \leq R_{\text{emp}}[g] + \sqrt{\frac{16}{M}\left(\ln E[\mathcal{N}(\mathcal{F}, A)] + \ln\frac{4}{\delta}\right)}. \tag{1.8}$$

The difficulty with this bound is that it may require knowledge of P in order to estimate the quantity $E[\mathcal{N}(\mathcal{F}, A)]$. In order to upper bound this expectation, we need to describe another useful combinatorial construct due to Vapnik and Chervonenkis. A set of M points is *shattered* by $\mathcal{F}$ if it can be labelled in all of the 2^M possible ways, *i.e.*, $\mathcal{N}(\mathcal{F}, M) = 2^M$. The *Vapnik-Chervonenkis (VC) dimension* of $\mathcal{F}$ is the maximum number of points that can be shattered by $\mathcal{F}$ (and is ∞ if no such set exists). Let d denote the VC dimension of $\mathcal{F}$. For $M > d$, the annealed entropy can be bounded in terms of the VC dimension [69, 66]:

$$\ln E[\mathcal{N}(\mathcal{F}, Z_{2M})] \leq d\left(\ln\frac{M}{d} + 1\right) \tag{1.9}$$

Freund and Schapire motivate the original AdaBoost with a risk (also known as generalization) bound in terms of the number of rounds, T, that the AdaBoost algorithm is run. In particular, they extend a previous result of Baum and Haussler for Neural Networks [8] to show that the *VC-dimension* of the classifier can be bounded in terms of the sum of VC-dimension of the base classifiers and T.

Theorem 1.3.1 (Freund and Schapire)**.** *The VC-dimension of a set of functions of the form* $\operatorname{sgn}(g(x)) = \operatorname{sgn}\left(\sum_{u\in\mathcal{U}} \lambda_u h_u(x)\right)$*, with* $||\lambda||_0 \leq T$ *is at most*

$$2(\hat{d}+1)(T+1)\log_2(e(T+1)),$$

where $\hat{d} \geq 2$ *is the VC-dimension of the set of base classifiers* $\{h_u \mid u \in \mathcal{U}\}$*.*

Theorem 1.3.1 implies that the VC dimension of a weighted voting classifier can be bounded in terms of a product of the VC dimension of its base classifiers, and $||\lambda||_0$. As a consequence of this result, along with the bounds (1.8) and (1.9), the risk of a weighted voting classifier can also be bounded in terms of this product. Thus, Theorem 1.3.1 also provides motivation for finding sparse weighted voting classifiers.

The VC dimension of a set of functions may seem dependent on the dimension of the

space. For example, the VC dimension of halfspaces in dimension N is known to be $N+1$; in two dimensions, for example, it is easy to see that any three points can be separated in all 2^3 possible ways using separating hyperplanes, while four points cannot be separated in all 2^4 possible ways. The following theorem, by Vapnik, shows that, in fact, the VC dimension of a possibly infinite set of functions can be bounded independently of the dimension of the space of a given set of points.

Theorem 1.3.2 (Vapnik [66], section 5.5.6)**.** *Let $\mathcal{F}$ denote the set of functions $f_\lambda(x) = \operatorname{sgn}\left(\sum_{u=1}^{U} \lambda_u h_u(x)\right)$, for which $\min_{i=1,\ldots,M}\left\{\left|\sum_{u=1}^{U} \lambda_u h(A_{iu})\right|\right\} = 1$, and $||\lambda||_2 \leq b$ for some $b > 0$. Then the VC dimension of $\mathcal{F}$ is at most r^2b^2, where*

$$r = max_{i=1,\ldots,M}\left\{\left|\left|\begin{pmatrix} h_1(A_i) & \ldots & h_U(A_i) \end{pmatrix}\right|\right|\right\}.$$

The theorem implies that the VC dimension can be controlled irrespective of the dimension U. Thus, a risk bound that does not depend on the dimension can be stated [66]:

Theorem 1.3.3 (Bartlett and Shawe-Taylor)**.** *Assume $||\lambda||_2 \leq b$ and $||x||_2 \leq r$, for some $r, b > 0$. Let $\rho > 0$ and ν denote the fraction of training observations with margin less than $\rho/||\lambda||_2$. For all distributions P generating the data, with probability at least $1-\delta$:*

$$R[g] \leq \nu + \sqrt{\frac{\kappa}{M}\left(\frac{r^2b^2\ln^2(M)}{\rho^2} + \ln\left(\frac{1}{\delta}\right)\right)},$$

where κ is a universal constant.

Similar bounds in terms of the L_1-norm have been developed for boosting algorithms by Schapire, Freund, Bartlett and Lee [63] and Demiriz, Bennett and Shawe-Taylor [26].

A risk bound may also justify a model selection procedure that minimizes the same bound. Minimizing the entire right hand side of the bound instead of just the empirical risk, $R_{\text{emp}}[g]$, for a given structure of $\mathcal{F}$, is suggested by Vapnik [69] as a part of *structural risk minimization* (SRM). In SRM, the function class $\mathcal{F}$ is decomposed into a sequence of subsets of increasing size and "capacity": $S_1 \subset S_2 \subset \ldots \subset \mathcal{F}$. We pick g from the subset S_j for some $j \geq 1$ that minimizes the risk bound. Vapnik's version of SRM assumes some predefined (*i.e.*, defined prior to the appearance of the data) hierarchy of complexity classes

$S_1, S_2, \ldots$ A choice of penalties based on a decomposition into classes of risk based on the actual data is known as a *(un)luckiness function* [68], which is used in general to encode bias in favor of certain classifiers, for example those corresponding to hyperplanes with a larger margin over others.

A different kind of risk bound that we will use to motivate the optimization formulations in this dissertation is in terms of *two-part code* length. The code is used by an imaginary sender and receiver that share the training observations, but only the sender has the actual labels y of the observations. The code is used to efficiently transmit the labels of the training data to the receiver. Vapnik [69] derives the following upper bound on the risk of a classifier $g \in \mathcal{F}$, with $|\mathcal{F}| < \infty$: with probability $1 - \delta$,

$$R[g] \leq 2\left(\ln 2\bar{L}[y, g]/M - \frac{\ln \delta}{M}\right),$$

where $\bar{L}[y, g]$ is an upper bound on the length of the code that encodes y using g, for a fixed A. Blum and Langford [13] suggest a different risk bound in terms of code length that applies also for infinite sets of functions $\mathcal{F}$, as long as the code of length $\bar{L}[y, g]$ can be used to encode both the training labels y and test labels $y' \in \{-1, 1\}^{M'}$, and transmits the training labels without error. Let us denote the corresponding observable attributes of the test data by $A' \in \mathbb{R}^{M' \times N}$. The resulting bound on the risk of $g \in \mathcal{F}$ holds with probability $1 - \delta$:

$$\tilde{R}[g] = \sum_{i=1}^{M'} \mathbf{I}(y'_i g(A'_i) \leq 0)/M' \leq \frac{\bar{L}[y, g] \ln 2 + \ln \frac{1}{\delta}}{M' \ln(1 + M/M')}, \tag{1.10}$$

where M' is the number of test observations.

By setting the number of test observations M' to equal M in (1.10) they also obtain the following simple risk bound in terms of the code length $\bar{L}[y, g]$ [13], with probability $1 - \delta$,

$$\tilde{R}[g] \leq \frac{\bar{L}[y, g]}{M} + \frac{\ln \delta}{M}. \tag{1.11}$$

1.4 Robust linear programming formulations for classification and LP-Boost

The current methodology of robust voting classification methods is to allow for some outliers, observations which do not necessarily lie on the correct side of the hyperplane (or the *decision boundary*), and to use regularization; penalizing either the L_1 or L_2 norms of λ. Robust linear separation formulations have been suggested by several authors such as Bennett and Mangasarian [12] and Cortes and Vapnik [23]. Cortes and Vapnik suggest maximizing a "soft margin" (1.13) in one of their papers first introducing Support Vector Machines (SVMs). SVM models find $g(\cdot)$ by maximizing the L_2 margin of separation on the space of features corresponding to $\mathcal{U}$. By "soft margin" it is meant that not all observations need to be separated and satisfy the same requirement of distance from the hyperplane. Graepel *et al.* [41] and Ratsch *et al.* [58] adapted the quadratic optimization formulation of SVMs using "soft margins" to a linear programming formulation.

Demiriz, Bennett and Shawe-Taylor [26] use a linear programming formulation based on the formulation of Graepel *et al.* [41] and Ratsch *et al.* [58], shown below as (1.12), in their LP-Boost algorithm:

$$\max \quad \rho - D\sum_{i=1}^{M}\xi_i \tag{1.12a}$$

$$\text{s.t.} \quad y_i H_i \lambda + \xi_i \geq \rho \qquad i = 1, ..., M \tag{1.12b}$$

$$\sum_{u=1}^{U}\lambda_u = 1 \tag{1.12c}$$

$$\xi_i, \lambda_u \geq 0 \qquad i = 1, ..., M,\ u = 1, ..., U \tag{1.12d}$$

In this formulation, each observation $i \in \{1, ..., M\}$ has a variable ξ_i which allows it to have a margin smaller than ρ. The margin of an observation i is equal to $y_i H_i \lambda$, where H_i is the i^{th} row of H, an $M \times U$ matrix, whose elements are $h_{iu} = h_u(A_i)$. The penalty parameter D penalizes the size of each deviation from the margin, $\xi_i = \rho - y_i H_i \lambda$. These margin deviations are illustrated in Figure 1.1. The matrix H, in which each column corresponds to a column of labels ($-1, 0, 1$ in our case) assigned by a base classifier $u \in \mathcal{U}$,

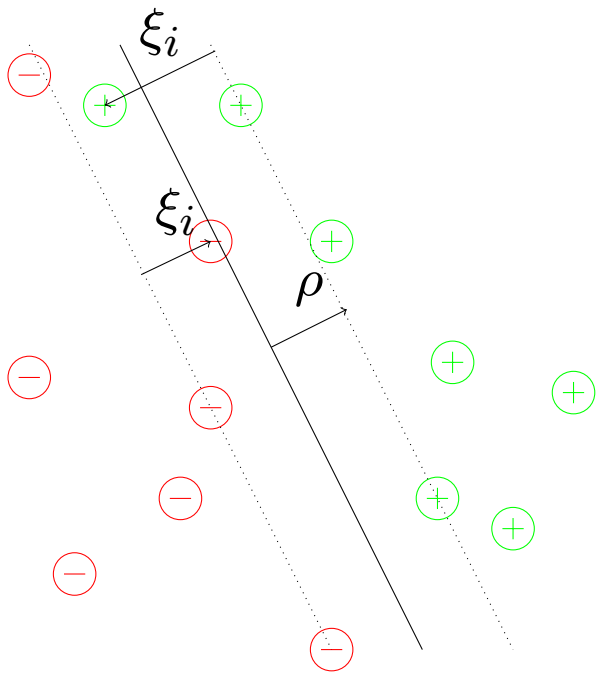


Figure 1.1: The margin ρ and margin deviations ξ_i illustrated in the "soft margin" LP formulation. The separating hyperplane is shown as a solid line, and the maximum margin planes as the parallel dotted lines. The points that lie in between the maximum margin lines are margin errors.

usually has too many columns to be written in full as a part of the *LP*. Instead Demiriz *et al.* [26] propose a column generation procedure as a solution technique of the LP-Boost algorithm (to be exact, they in fact propose to generate cuts for the dual formulation).

Column generation is an approach for solving a large linear programming formulation without having to explicitly include all of its variables. The method starts by solving a *restricted master problem* (RMP) that includes only a subset of the variables, and uses dual variable information to iteratively generate additional (primal) variables by solving the *pricing* subproblem. The algorithm appends the newly generated variable(s) to the RMP and re-solves until no additional variables can be generated. Column generation has been used to solve large LPs since the early 1960s, with a wide variety of applications starting with the seminal application of Gilmore and Gomory to the cutting stock problem [39, 51]. The method is successful in practice when there are many more columns than rows and the number of variables that need to be generated tends to be small.

In the formulation (1.12), the objective function is to maximize the L_1 margin of separation less a misclassification penalty. It has been shown by Bennett and Bredensteiner [11] that formulation (1.12) is equivalent to:

$$\min \left\{ \sum_{u=1}^{U} \lambda_u + D' \sum_{i=1}^{M} \xi_i \;\mid\; \mathrm{diag}(y) H \lambda + \xi \geq \mathbf{1}, \text{ and } \lambda, \xi \geq 0 \right\}, \tag{1.13}$$

for appropriately chosen constants D and D':

Theorem 1.4.1 (Bennett and Bredensteiner [11] and Demiriz, Bennett and Shawe-Taylor [26])**.** *If LP* (1.12) *has an optimal solution* (λ, ρ, ξ) *with parameter* D *and* $\rho \neq 0$*, then* $(\lambda/\rho, \xi/\rho)$ *is an optimal solution of* (1.13) *with parameter* $D' = D/\rho$*. Similarly, if LP* (1.13) *has an optimal solution* $(\hat{\lambda}, \hat{\xi})$ *with parameter* D' *and* $\hat{\lambda} \neq 0$*, then* $(\hat{\lambda}/||\hat{\lambda}||_1, 1/||\hat{\lambda}||_1, \hat{\xi}/||\hat{\lambda}||_1)$ *is an optimal solution of LP* (1.12) *with* $D = D'/||\hat{\lambda}||_1$*.*

Thus, the formulation (1.12) can be considered to find sparse solutions by minimizing the L_1 norm of λ. The dual formulation of (1.12) is the following:

$$\begin{aligned}
\min \quad & \alpha & & & \text{(1.14a)} \\
\text{s.t.} \quad & \sum_{i=1}^{M} w_i = 1 & & & \text{(1.14b)} \\
& \sum_{i=1}^{M} y_i H_{ij} w_i \leq \alpha & & \text{for } j = 1, \ldots, U & \text{(1.14c)} \\
& 0 \leq w_i \leq D & & & \text{(1.14d)}
\end{aligned}$$

In the dual formulation, the parameter D corresponds to an upper bound on the "weight" of an observation. Another way to view the parameter D is by substituting $D = \frac{1}{\nu M}$, in which case the parameter ν can be interpreted as the maximum fraction of *margin errors*. A margin error is an observation $i \in \{1, \ldots, M\}$, for which $\xi_i = (\rho - y_i H_i \lambda)_+ > 0$.

Graepel *et al.* [41] and Rätsch *et al.* [59] state a theorem that provides an interpretation for the parameter ν: for any optimal solution (λ, ρ), ν is an upper bound on the fraction of margin errors, and $1 - \nu$ as an upper bound on the fraction of points with margin larger than ρ; they provide a variational and graphical sketch of a proof, respectively. Schölkopf *et*

al. [65] prove a similar theorem for an SVM quadratic programming formulation where the L_2-norm of λ is being minimized; in particular they show that ν is an upper bound on the fraction of errors and a lower bound on the fraction of support vectors. In the following, we show a simple algebraic proof, based on complementary slackness conditions for the LP (1.12), of the theorem by Graepel *et al.* [41] and Rätsch *et al.* [59].

Theorem 1.4.2 (Graepel *et al.* [41] and Rätsch *et al.* [59])**.** *In any optimal solution of* (1.13) (ξ, λ, ρ)*:*

1. ν *upper-bounds the fraction of margin errors.*

2. $1 - \nu$ *upper-bounds the fraction of points with margin larger than* ρ*.*

Proof. To prove the first claim, note that, by dual complementary slackness,

$$\xi_i > 0 \Rightarrow w_i = D = \frac{1}{\nu M}.$$

Thus,

$$\sum_{i:\xi_i>0} w_i = \frac{|\{i : \xi_i > 0\}|}{\nu M} \leq \sum_{i=1}^{M} w_i = 1,$$

where the last inequality follows from dual feasibility, and constraint (1.14b) being satisfied. It follows that $\frac{|\{i:\xi_i>0\}|}{M} \leq \nu$.

To establish the second claim, we make use of the primal complementary slackness conditions:

$$y_i H_i \lambda > \rho \Rightarrow w_i = 0.$$

With dual feasibility and constraint (1.14b), these imply

$$\sum_{i=1}^{M} w_i = \sum_{i:y_i H_i \lambda > \rho} w_i + \sum_{i:y_i H_i \lambda \leq \rho} w_i = \sum_{i:y_i H_i \lambda \leq \rho} w_i = 1.$$

Thus, summing the upper bounds (1.14d) and multiplying by ν,

$$\frac{|\{i \mid y_i H_i \lambda \leq \rho\}|}{M} \geq \nu.$$

So that

$$1 - \frac{|\{i \mid y_i H_i \lambda \leq \rho\}|}{M} = \frac{|\{i \mid y_i H_i \lambda > \rho\}|}{M} \leq 1 - \nu. \qquad \square$$

Additional observations can be made about the parameter D in formulation (1.12) and its dual LP (1.14). First we note that $D \geq 1/M$, otherwise (1.12) becomes unbounded (and its dual (1.14) becomes infeasible). If D is small, then λ tends to be sparse, but if it is too small then we get degenerate solutions with $\rho = 1$ in which all observations are assigned to be in the same class, *i.e.*, λ corresponds to a non-separating hyperplane. If D is made large enough, the optimal solution misclassifies very few observations, or none, if the data are linearly separable. But such a solution may correspond to a dense λ. If the data are not linearly separable, with a large enough D, the formulation will effectively minimize the margin deviations rather than maximize the margin. The straightforward margin deviation minimization formulation where all ξ_i have equal coefficients in the objective function, often fails to find meaningful hyperplane solutions in practice. In our case if $h_1(A_i) = 1$ for $i = 1, \ldots, M$ in order to assume the role of a constant in $\sum_{u=1}^{\mathcal{U}} \lambda_u h_u(x) = 0$, then $\lambda_1 > 0$ and $\lambda_u = 0$ for $u = 2, \ldots, \mathcal{U}$, corresponds to such a "null" hyperplane solution. Bennett and Mangasarian [12] prove conditions under which the simple margin deviation minimization results in an optimal "null" hyperplane solution.

1.5 The base learning and maximum agreement problems - the basic formulation and brief survey of literature

First, we consider the problem of the base learning algorithm pricing the best column within the column generation framework: that is, how to find the best feature in each iteration of a boosting algorithm. While much of the boosting literature (see [36, 64, 59, 58, 26, 54, 62]) has focused on run-time analysis as well as learning generalization bounds with respect to the boosting algorithm, not much work has been done in the area of developing new base learning algorithms for boosting.

The original AdaBoost algorithm [36] requires that the base learning algorithm be able to find a base classifier u, given a weight vector $w \in \mathbb{R}_+^M$ and some $\epsilon > 0$, so that

$\sum_{i=1}^{M} y_i H_{iu} w_i / \sum_{i=1}^{M} w_i \geq 1/2 + \epsilon$. Once this condition is satisfied, it guarantees exponential convergence of AdaBoost to zero training loss, *i.e.*, when the data is linearly separable and the sum of (1.2) over all observations can be zero (see Freund and Schapire [36] and the survey by Meir and Rätsch [54]). The pricing problem in LP-Boost [26], on the other hand, is to find the column with minimum reduced cost, that is:

$$\min_{u \in \mathcal{U}} \left\{ -\sum_{i=1}^{M} y_i H_{iu} w_i - \alpha \right\} \Leftrightarrow \max_{u \in \mathcal{U}} \left\{ \sum_{i=1}^{M} w_i y_i H_{iu} \right\} = \max_{u \in \mathcal{U}} \left\{ \sum_{\substack{i \in \{1,\ldots,M\}: \\ y_i H_{iu} = 1}} w_i - \sum_{\substack{i \in \{1,\ldots,M\}: \\ y_i H_{iu} = -1}} w_i \right\}, \tag{1.15}$$

where w_i and α are the dual variables corresponding to constraints (1.12b) and (1.12c), respectively.

The same objective, in fact, has been long known in machine learning and computational geometry communities as *maximum agreement* [48, 9, 40] or *maximum bi-chromatic discrepancy* [27]. A probabilistic argument for arriving at the same objective can be made for abstaining base classifiers. A base classifier (or hypothesis) is said to *abstain* on $i \in \{1, \ldots, M\}$ if $h_u(A_i) = 0$ (see also [63]). When $h_u(x) = 0$, *i.e.* it abstains, we may consider it as having an error rate of $\frac{1}{2}$, the same as a coin toss. We can then write the empirical error or weighted sum of the accuracy loss (1.3) for a set of i.i.d. observations, with a prior probability distribution w, as:

$$\begin{aligned}
\sum_{i=1}^{M} \ell(A_i, y_i, h_u(A_i)) w_i &= \sum_{i: y_i H_{iu} = -1} w_i + \frac{1}{2} \sum_{i: y_i H_{iu} = 0} w_i \\
&= \sum_{i: y_i H_{iu} = -1} w_i + \frac{1}{2} \left(\sum_{i \in \Omega^+} w_i - \sum_{i: y_i H_{iu} = 1} w_i + \sum_{i \in \Omega^-} w_i - \sum_{i: y_i H_{iu} = -1} w_i \right) \\
&= \frac{1}{2} \left(\sum_{i: y_i H_{iu} = -1} w_i - \sum_{i: y_i H_{iu} = 1} w_i \right) + \frac{1}{2}
\end{aligned} \tag{1.16}$$

We note that the maximum agreement problem (1.15) is equivalent, from an exact

optimization point of view, to a minimum disagreement problem:

$$\min_{u \in \{1,\ldots,U\}} \left\{ \sum_{i:y_i H_{iu}=-1} w_i - \sum_{i:y_i H_{iu}=1} w_i \right\}.$$

The computational complexity of maximum agreement problems depends on the properties of the space corresponding to the columns of the matrix H. Trivially, if U is polynomial in the size of M, then the problem is polynomially solvable by simple enumeration. The computational complexity of more interesting maximum agreement problems was first investigated by Pitt and Valiant, who proved that it is $\mathcal{NP}$-hard to decide if there is a 2-term Disjunctive Normal Form that agrees with all observations of a dataset [57]. Kearns, Schapire and Sellie [48], following work by Kearns and Li [45], investigated minimum disagreement with Boolean monomials and showed that the problem is $\mathcal{NP}$-hard.

The inapproximability of maximum agreement problems has been investigated by Ben David, Eiron and Long [9], Bshouty and Burroughs [19], and Feldman [33]. Ben-David *et al.* show that it is $\mathcal{NP}$-hard to approximate the maximum agreement problem to within various fixed constants for the class of monomials, axis-aligned hyper-rectangles, closed balls and monotone monomials. Bshouty and Burroughs [19] improve on some of these inapproximability factors and derive new ones for decision lists, exclusive-or, Boolean halfspaces, 2-term DNFs and 2-term multivariate polynomials. Feldman [33] shows optimal inapproximability results for maximum monomial agreement[1].

In Chapter 2 we proceed to formally introduce the maximum monomial agreement problem and suggest an algorithm for efficiently solving the problem in practice. In Chapter 3 we revisit the problem of weighted voting classifier model selection: we formulate a discrete optimization problem for choosing λ, *sparse minimum disagreement hyperplane* (SMDH), which achieves an optimal balance between empirical (*i.e.*, 0/1) loss minimization and the density (*i.e.*, lack of sparsity) of λ. We show that the resulting discrete optimization problem is hard. We then consider the continuous relaxation of a mixed integer program for

[1]However, this result applies to an objective function that differs by a constant and excludes the absolute value of the objective function (1.15), which we consider.

solving the SMDH problem. We are able to show that the continuous relaxation is equivalent to the existing, and well known, robust LP formulation for choosing λ (1.12). We then propose novel cutting planes for tightening the relaxation and suggest solution techniques for dynamically generating columns and cuts in order to solve the tightened LP relxation. In Chapter 4, we investigate the new base learning (or pricing) problem which arises with the novel LP relaxation. In particular, we suggest an abstract formulation and then focus on the problem with Boolean monomial base classifiers by extending the solution techniques of maximum monomial agreement considered in Chapter 2.

Chapter 2

Maximum Monomial Agreement

2.1 Problem Statement and Introduction

The *maximum monomial agreement* (MMA) problem has applications to machine learning, data mining [48, 40], as well as computational geometry and computer graphics [27], where it is known as the *maximum bi-chromatic discrepancy* problem. The problem can be motivated by applications to classification independently of weighted voting classifiers and boosting. For example, we may have M patients, with Ω^+ corresponding to those having a particular disease, and Ω^- to those who do not. Each patient has N binary attributes corresponding to personal traits or results of medical tests, and we would like to find the interaction of attributes which best predicts presence or absence of the disease. For example, a monomial might represent a rule or hypothesis such as ('age' ≥ 50) $\wedge$ ('blood pressure' = 'high') $\rightarrow$ 'heart disease'.

However, monomial hypotheses have been found especially useful for constructing weighted voting classifiers. In particular, boosting of monomial hypotheses has been suggested by several authors [20, 38, 40]. The experimental results in [40] show that boosting optimal monomial hypotheses, as opposed to heuristically generated monomials (*e.g.* as in SLIPPER [20]), can improve the classification performance when using a sufficiently robust boosting algorithm, such as Servedio's SmoothBoost [67]. Monomial hypotheses (also called logical patterns) are also a basic building block in the *logical analysis of data* (LAD) methodology [16], where linear programming as well as other techniques are suggested for computing the discriminant function as a linear combination of monomials.

As input, we now assume that the given input data $A \in \mathbb{R}^{M \times N}$ is a 0/1 matrix. In this chapter, we consider only binary datasets; however, for general datasets in $\mathbb{R}^N$, we note that there is a corresponding "binarization" with dimension that is at most polynomially larger

than M, and in practice does not tend to be much larger than N [15, 40]. As before we are also given a partition of its rows into positive observations $\Omega^+ \subset \{1, ..., M\}$ and negative observations $\Omega^- = \{1, ..., M\} \backslash \Omega^+$. Finally, we are given a function $w : \{1, \ldots, M\} \to [0, \infty)$; $w(i)$ specifies the weight of observation i. In the context of algorithms such as LP-Boost $w(i)$ corresponds to the dual variable associated with observation i.

A monomial on $\{0,1\}^N$ is simply a function $p : \{0,1\}^N \to \{0,1\}$ of the form

$$m_{J,C}(x) = \prod_{j \in J} x_j \prod_{c \in C} (1 - x_c), \tag{2.1}$$

where J and C are (usually disjoint) subsets of $\{1, \ldots, N\}$. The monomial $m_{J,C}$ is said to *cover* a vector $x \in \{0,1\}^N$ if $m_{J,C}(x) = 1$. We define the *coverage* of a monomial $m_{J,C}$ to be

$$\text{Cover}(J, C) = \{ i \in \{1, ..., M\} \mid m_{J,C}(A_i) = 1 \},$$

where A_i denotes the i^{th} row of A. Defining the weight of a set $S \subseteq \{1, \ldots, M\}$ to be $w(S) = \sum_{i \in S} w(i)$, the MMA problem is to find disjoint sets $J, C \subseteq \{1, \ldots, N\}$ maximizing

$$f(J, C) = \left| w(\text{Cover}(J, C) \cap \Omega^+) - w(\text{Cover}(J, C) \cap \Omega^-) \right|. \tag{2.2}$$

As mentioned in Chapter 1, and as is the case for other interesting cases of maximum agreement problems, this problem is known to be $\mathcal{NP}$-hard [48]. Furthermore, it has been shown [33] that even if the weights $w(i)$ are all equal, a related problem with the slightly different maximization objective

$$\begin{aligned} \tilde{f}(J, C) &= \left(w(\text{Cover}(J, C) \cap \Omega^+) + w(\Omega^- \setminus \text{Cover}(J, C)) \right) / M \\ &= \left(w(\text{Cover}(J, C) \cap \Omega^+) - w(\text{Cover}(J, C) \cap \Omega^-) + w(\Omega^-) \right) / M, \end{aligned}$$

is $\mathcal{NP}$-hard to approximate to any factor less than 2. Approximation of the latter objective to a factor of 2 is trivial, however, by simply considering only the constant monomials such that either $m_{J,C}(A_i) = 0$ for all $i \in \{1, \ldots, M\}$, or $m_{J,C}(A_i) = 1$ for all $i \in \{1, \ldots, M\}$.

Dobkin *et al.* consider the maximum bi-chromatic discrepancy problem for axis-aligned

rectangles in $\mathbb{R}^2$ and propose an $O(M^2 \log M)$ algorithm. They also suggest a generalization of the algorithm for $\mathbb{R}^n$, which results in $O(M^{2n-2} \log M)$ running time. When the input data of the problem consists of points in $\{0,1\}^N$, then any axis-aligned rectangle corresponds to a monomial $m_{J,C}$. Another related (but not equivalent) problem for real-valued data is the *maximum box* problem [29]. For the special case of input data in $\{0,1\}^N$, the (weighted) maximum box problem can be stated as the problem of finding a box or subcube (J, C) that maximizes $w(\Omega^+ \cap \text{Cover}(J, C))$, and such that $\Omega^- \cap \text{Cover}(J, C) = \emptyset$.

If the space of possible weak learners consists of all monomial base classifiers, the column pricing problem (1.15) of a linear program such as (1.12) corresponds to finding a monomial (J, C) that maximizes (2.2).

2.2 Branch and bound

One possible approach to exactly solving the MMA is to formulate it as an equivalent integer linear program, and solve it with a standard integer programming solver. However, it is more efficient to solve the problem by a specialized branch-and-bound algorithm. The key elements of a branch-and-bound algorithm are the definition of subproblems that represent sets of possible solutions, a method for computing a bound on the objective value of all solutions represented by a subproblem, and a branching procedure for subdividing subproblems into smaller ones.

We characterize each subproblem as a partition (J, C, E, F) of $\{1, \ldots, N\}$. As in (2.1), J and C respectively represent the literals which must be in the monomial, and whose complements must be in the monomial. E indicates a set of "excluded" literals: neither they nor their complements may appear in the monomial. Finally, $F = \{1, \ldots, N\} \backslash (J \cup C \cup E)$ is the set of "free", undetermined literals. The set of monomials corresponding to subproblem (J, C, E, F) is given by

$$P(J, C, E) = \left\{ (J', C') \;\middle|\; J' \supseteq J,\, C' \supseteq C, \text{ and } J', C', E \text{ are disjoint} \right\}. \tag{2.3}$$

The search begins with a priority queue Q containing the single root subproblem

$(\emptyset, \emptyset, \emptyset, \{1 \ldots, N\})$, which corresponds to the set of all possible monomials. At each iteration, we remove a subproblem (J, C, E, F) from Q and, unless it is fathomed, that is, its upper bound $b(J, C, E) \leq f(J^*, C^*)$ for some known J^*, C^*, we further subdivide (branch) it into smaller subproblems. For the fathoming test to be valid, the upper bound function u should have the property

$$b(J, C, E) \geq f(J', C') \quad \forall\, (J', C') \in P(J, C, E). \tag{2.4}$$

2.2.1 The upper bound function

In the special case that $F = \emptyset$, the set $P(J, C, E)$ is a singleton consisting only of (J, C), and we can take $b(J, C, E) = f(J, C)$. If $F \neq \emptyset$, we must use some other function satisfying (2.4). In previous work with Shan [40], we suggested the simple bound $b(J, C, E) = b_{\text{gs}}(J, C)$, where

$$b_{\text{gs}}(J, C) = \max\{w(\text{Cover}(J, C) \cap \Omega^+), w(\text{Cover}(J, C) \cap \Omega^-)\} \tag{2.5}$$

Theorem 2.2.1. *The bound* (2.5) *satisfies property* (2.4)*.*

Proof. Let $(J', C') \in P(J, C, E)$. Then by definition $J' \supseteq J$ and $C' \supseteq C'$, and clearly $\text{Cover}(J', C') \subseteq \text{Cover}(J, C)$. Now,

$$\begin{aligned}
b_{\text{gs}}(J, C) &= \max\{w(\text{Cover}(J, C) \cap \Omega^+), w(\text{Cover}(J, C) \cap \Omega^-)\} \\
&\geq \max\{w(\text{Cover}(J', C') \cap \Omega^+), w(\text{Cover}(J', C') \cap \Omega^-)\} \\
&\geq \max \left\{ \begin{array}{l} w(\text{Cover}(J', C') \cap \Omega^+) - w(\text{Cover}(J', C') \cap \Omega^-), \\ w(\text{Cover}(J', C') \cap \Omega^-) - w(\text{Cover}(J', C') \cap \Omega^+) \end{array} \right\} \\
&= f(J', C').
\end{aligned}$$

□

However, this bound essentially ignores the information in the set E. To obtain a stronger bound, we now exploit the information in E:

Definition 2.2.1. *Two binary vectors* $x, y \in \{0, 1\}^N$ *are* inseparable *with respect to a set* $E \subseteq \{1, ..., N\}$*, if, for all* $j \in \{1, ..., N\} \setminus E$*, one has* $x_j = y_j$*.*

Inseparability is an equivalence relation: any set $E \subseteq \{1, \ldots, N\}$ thus partitions $\{1, ..., M\}$ into equivalence classes, i and i' being in the same class when the observation A_i is inseparable from the observation $A_{i'}$. Let us denote these classes by $V_1^E, V_2^E, ..., V_{q(E)}^E$, where $1 \leq q(E) \leq m$. Let

$$w_\eta^+(J, C, E) = w(V_\eta^E \cap \text{Cover}(J, C) \cap \Omega^+) \qquad w^+(J, C) = w(\text{Cover}(J, C) \cap \Omega^+)$$
$$w_\eta^-(J, C, E) = w(V_\eta^E \cap \text{Cover}(J, C) \cap \Omega^-) \qquad w^-(J, C) = w(\text{Cover}(J, C) \cap \Omega^-)$$

for $\eta = 1, \ldots, q(E)$, and note that

$$f(J, C) = \left|w^+(J, C) - w^-(J, C)\right| = \left|\sum_{\eta=1}^{q(E)} (w_\eta^+(J, C, E) - w_\eta^-(J, C, E))\right|. \tag{2.6}$$

We will call a monomial *positive* if $w^+(J, C) \geq w^-(J, C)$, and *negative* if $w^+(J, C) < w^-(J, C)$. Positive monomials cover more weight of positive than negative observations, and negative monomials the reverse; we include ties in the positive class.

Now consider some $(J', C') \in P(J, C, E)$, assume it is positive, and take $\eta \in \{1, \ldots, q(E)\}$. Because the observations in the equivalence class V_η^E are inseparable with respect to E, the monomial $m_{J',C'}(x)$ must either cover all of V_η^E, or cover none of it. Considering the η^{th} term in the second expression in (2.6), we see that the largest value of $f(J', C')$ would result if $m_{J',C'}(x)$ covers the entire class V_η^E whenever $w_\eta^+(J, C, E) \geq w_\eta^-(J, C, E)$, obtaining a value of $w_\eta^+(J, C, E) - w_\eta^-(J, C, E)$, and otherwise does not cover the entire class V_η^E, obtaining the value 0. Thus, if (J', C') is positive, we have

$$f(J', C') \leq \sum_{\eta=1}^{q(E)} \Big(w_\eta^+(J, C, E) - w_\eta^-(J, C, E)\Big)_+.$$

By nearly identical reasoning, we can obtain a similar relation for the case that (J', C') is negative, and combining the two cases, we derive the following upper bound function

satisfying (2.4):

$$b(J,C,E) = \max \left\{ \begin{array}{l} \sum_{\eta=1}^{q(E)} \left(w_\eta^+(J,C,E) - w_\eta^-(J,C,E) \right)_+, \\ \sum_{\eta=1}^{q(E)} \left(w_\eta^-(J,C,E) - w_\eta^+(J,C,E) \right)_+ \end{array} \right\}. \tag{2.7}$$

A similar use of equivalence classes to obtain tightened subproblem bounds has been used by De Bontridder *et al.* [14], in the context of the minimum test cover problem. Furthermore, we can also rewrite the upper bound $b(J,C,E)$ in terms of the simple upper bound (2.5).

Theorem 2.2.2. *The upper bound* (2.7) *can be equivalently written as,*

$$b(J,C,E) = b_{\text{gs}}(J,C) - \sum_{\eta=1}^{q(E)} \min\{w_\eta^+(J,C,E), w_\eta^-(J,C,E)\}. \tag{2.8}$$

Proof. Setting $w_\eta^+ = w_\eta^+(J,C,E)$, $w_\eta^- = w_\eta^-(J,C,E)$, and $q = q(E)$ for brevity,

$$\begin{aligned} &\max \left\{ \sum_{\eta=1}^{q} (w_\eta^+ - w_\eta^-)_+, \ \sum_{\eta=1}^{q} (w_\eta^- - w_\eta^+)_+ \right\} \ + \ \sum_{\eta=1}^{q} \min\{w_\eta^+, w_\eta^-\} \\ &= \max \left\{ \sum_{\eta=1}^{q} (w_\eta^+ - w_\eta^-)_+ + \sum_{\eta=1}^{q} \min\{w_\eta^+, w_\eta^-\}, \ \sum_{\eta=1}^{q} (w_\eta^- - w_\eta^+)_+ + \sum_{\eta=1}^{q} \min\{w_\eta^+, w_\eta^-\} \right\} \\ &= \max \left\{ \begin{array}{l} \sum_{\eta:\, w_\eta^+ \geq w_\eta^-} (w_\eta^+ - w_\eta^- + w_\eta^-) \ + \ \sum_{\eta:\, w_\eta^+ < w_\eta^-} w_\eta^+, \\ \sum_{\eta:\, w_\eta^- \geq w_\eta^+} (w_\eta^- - w_\eta^+ + w_\eta^+) \ + \ \sum_{\eta:\, w_\eta^- < w_\eta^+} w_\eta^- \end{array} \right\} \\ &= \max\{w^+(J,C), w^-(J,C)\} = b_{\text{gs}}(J,C). \end{aligned}$$

□

We define $\phi(J,C,E) = \sum_{\eta=1}^{q(E)} \min\{w_\eta^+(J,C,E), w_\eta^-(J,C,E)\}$, the second term in (2.8), to be the *inseparability* of E with respect to J and C. Unless $\phi(J,C,E) = 0$, the bound $b(J,C,E)$ is strictly tighter than $b_{\text{gs}}(J,C)$. If $\phi(J,C,E) = 0$, then all the sets $\text{Cover}(J,C) \cap V_\eta^E$, for $\eta = 1, \ldots, q(E)$, are homogeneous with respect to observations with nonzero weight, meaning that among the observations $i \in V_\eta^E$ with $w(i) > 0$, all are either in Ω^+, or all are in Ω^-.

Both of the bounds (2.5) and (2.7) may be computed in $O(MN)$ time in the worst case.

i \ j	1	2	3	4
1	0	0	1	1
2	1	0	0	1
3	1	0	1	1
4	0	1	1	0
5	1	0	0	0

				E	
	j	1	3	2	4
η	i				
1	1	0	1	0	1
	4	0	1	1	0
2	2	1	0	0	1
	5	1	0	0	0
3	3	1	1	0	1

Figure 2.1: The given matrix A is shown on the left and its corresponding matrix with rows sorted lexicographically by the entries in the columns $\{1, \ldots, N\} \setminus E$ is given on the right. Each equivalence class η can be identified as a set of rows with identical values in these columns. In the example $\{1, \ldots, N\} \setminus E = \{1, 3\}$.

For (2.7), the equivalence classes $\{V_\eta^E\}$ need to be computed, which can be done in $O(MN)$ time, by applying a radix sort to the rows of a submatrix of A with columns $\{1, \ldots, N\} \setminus E$. A numerical example of the procedure for identifying the equivalence classes is shown in Figure 2.1. The radix sort algorithm, in fact, has $\Theta(MN)$ running time. Computing $b_{\text{gs}}(J, C)$, on the other hand, requires only the computation of the set $\text{Cover}(J, C)$, which involves $|J \cup C| \leq N$ set intersection operations, each of which tends to be much faster than $O(M)$ in practice. Thus, it may be more practically efficient to first compute $b_{\text{gs}}(J, C)$, and if that does not fathom the subproblem (J, C, E, F), then compute $\phi(J, C, E)$ and thus (2.7). Further, it is also possible to use any lower bound for $\phi(J, C, E)$ in order to obtain a tightening of the upper bound b_{gs}. Lower bounds of $\phi(J, C, E)$ can be iteratively computed as $\sum_{\eta=1}^{\bar{\eta}} \min\{w_\eta^+(J, C, E), w_\eta^-(J, C, E)\}$, for $\bar{\eta} = 1, \ldots, q(E)$, until either the subproblem (J, C, E, F) is fathomed, or $\bar{\eta} = q(E)$.

2.2.2 The branching procedure

Our branching procedure works as follows: given a subproblem (J, C, E, F), we select k distinct elements $j_1, \ldots, j_k$ of F, where $1 \leq k \leq |F|$. In our branching step, we create $2k+1$ smaller subproblems respectively representing the following subsets of $P(J, C, E)$:

- The subset of monomials in which none of $x_{j_1}, \ldots, x_{j_k}$ appear.
- For $t = 1, \ldots, k$, the subset of monomials in which x_{j_t} is the first in the sequence

$x_{j_1}, \ldots, x_{j_k}$ to appear, in the uncomplemented form; $x_{j_1}, \ldots, x_{j_{t-1}}$ are excluded from further consideration.

- For $t = 1, \ldots, k$, the subset of monomials in which x_{j_t} is the first in the sequence $x_{j_1}, \ldots, x_{j_k}$ to be used, and appears in the complemented form $(1 - x_{j_t})$; $x_{j_1}, \ldots, x_{j_{t-1}}$ are excluded from further consideration.

These subsets clearly form a partition of $P(J, C, E)$. In our notation, the corresponding subproblems are represented by the respective 4-tuples in lines 13, 15, and 16 of Algorithm 1 on page 29. We have significant latitude in choosing k for each subproblem. We have experimented with choosing $\alpha \in \{1/4, 1/2, 3/4, 1\}$, and setting $k = \lceil \alpha |F| \rceil$ for each subproblem; we have also experimented with fixing $k = 1$.

Motivated by (2.8), we attempt for a given k to choose $j_1, \ldots, j_k$ to maximize the inseparability $\phi(J, C, E \cup \{j_1, \ldots, j_k\})$. We now show that exactly maximizing $\phi(J, C, E \cup \{j_1, \ldots, j_k\})$ is itself $\mathcal{NP}$-hard. We will show that even the case $J = C = \emptyset$ and $w(i) = 1$ for all $i \in \{1, \ldots, M\}$ is $\mathcal{NP}$-hard.

Theorem 2.2.3. *The problem of finding a set $S \subseteq \{1, \ldots, N\}$, of size k, that maximizes $\hat{\phi}(S) = \phi(\emptyset, \emptyset, S)$ is $\mathcal{NP}$-hard.*

Proof. The proof follows by reduction of the clique problem in a graph. Given a graph $G = (V, E)$ and an integer parameter $k \leq |V|$, the clique problem is to find whether there exists a subset $S \subseteq V$, such that $|S| \geq k$ and every two vertices in S are joined by an edge in E. The dense k-subgraph problem is the problem of finding $S \subseteq V$, such that $|S| = k$ and that the maximum number of edges join vertices in S. The clique decision problem can be solved by the dense k-subgraph optimization problem by simply checking whether the optimal objective value equals $\binom{k}{2}$.

Let $N = |V|$. For each edge $(u, v) \in E$, we create one element $i \in \Omega^+$ and a vector A_i such that

$$A_{ij} = \begin{cases} 1, & j = u \text{ or } j = v \\ 0, & \text{otherwise} \end{cases},$$

and another element $i' \in \Omega^-$ such that $A_{i'} = 0$, letting $w(i) = w(i') = 1$.

Let $q(S)$ denote the number of equivalence classes induced by the set $S \subseteq \{1, \ldots, N\}$. Note that since $A_i = 0$ for all $i \in \Omega^-$, then either $w_\eta^-(\emptyset, \emptyset, S) = 0$ or $w_\eta^-(\emptyset, \emptyset, S) = |E|$. Without loss of generality, we assume that $\eta = 1$ denotes the class index corresponding to the zero vector, and hence $w_1^- = |E|$.

$$\begin{aligned}\max_{S \subseteq \{1,\ldots,N\}:|S|=k} \{\hat{\phi}(S)\} &= \max_{S \subseteq \{1,\ldots,N\}:|S|=k} \left\{ \sum_{\eta=1}^{q(S)} \min\{w_\eta^+(\emptyset, \emptyset, S), w_\eta^-(\emptyset, \emptyset, S)\} \right\} \\ &= \max_{S \subseteq \{1,\ldots,N\}:|S|=k} \{w_1^+(\emptyset, \emptyset, S)\}\end{aligned}$$

The last equality follows because $w_\eta^-(\emptyset, \emptyset, S) = 0$ for $2 \leq \eta \leq q(S)$, and $w_1^+(J, C, S) \leq w_1^-(J, C, S) = |E|$. Now,

$$w_1^+(\emptyset, \emptyset, S) = \left|\left\{ i \in \Omega^+ \mid \forall j \in \{1, \ldots, N\} \setminus S : A_{ij} = 0 \right\}\right| = \left|\{(u, v) \in E \mid u \in S \wedge v \in S\}\right|.$$

The equivalence class 1 consists of vectors indistinguishable from 0 when excluding S. For the observations in Ω^+, these are precisely the vectors with both their 1's in the vector position indices in S, that is, those corresponding to edges (u, v) with $u, v \in S$. Thus, maximizing $\hat{\phi}(S)$ subject to $|S| = k$ solves the dense k-subgraph problem, and

$$\max_{S \subseteq \{1,\ldots,N\}:|S|=k} \{w_1^+(\emptyset, \emptyset, S)\} = \binom{k}{2}$$

if and only if there exists a clique of size k in G. □

In the reduction of our proof, $\hat{\phi}(S)$ equals the number of edges joining the set of vertices S, implying that the reduction of the dense k-subgraph is approximation-preserving. Thus, based on an inapproximability result of Khot for the dense k-subgraph problem [49], we can also make the following claim:

Theorem 2.2.4. *There is no polynomial time approximation scheme (PTAS) for the problem* $\max_{S \subseteq \{1,\ldots,N\}:|S|=k} \hat{\phi}(S)$*, unless* $\mathcal{NP} \subseteq \bigcap_{\epsilon>0} BPTIME(2^{N^\epsilon})$.[1]

---

[1] BPTIME(t) is the class of decision problems solvable by an t-time probabilistic Turing machine such that:

1. If the answer is "yes" then at least 2/3 of the computation paths accept.

Given that the problem of maximizing $\phi(J, C, E)$ subject to a cardinality constraint on E is $\mathcal{NP}$-hard, we consider fast greedy heuristics to use within the branch-and-bound procedure for selecting the set $E \subset \{1, \ldots, N\}$ of size k. For any sets $S \subset E \subset \{1, \ldots, N\}$,

$$\begin{aligned} & \left|\left\{(i, i') \in \Omega^+ \times \Omega^- \;\middle|\; A_{ij} = A_{i'j}, \forall j \in \{1, \ldots, N\} \setminus (S \cup \{j'\})\right\}\right| \\ \leq \quad & \left|\left\{(i, i') \in \Omega^+ \times \Omega^- \;\middle|\; A_{ij} = A_{i'j}, \forall j \in \{1, \ldots, N\} \setminus (E \cup \{j'\})\right\}\right|. \end{aligned}$$

That is, excluding an additional variable j' may induce fewer inseparable observation pairs when only a subset of the variables E are excluded to begin with. In fact, when $E = \emptyset$, selecting any single variable to add to E is unlikely to make many pairs inseparable or to give any indication about the number of pairs that can be made inseparable with the "proper" combination with additional variables to include in E. Intuitively, this observation motivates our use of a reverse (also known as worst-out) greedy heuristic, instead of the forward greedy algorithm. Conceptually, we set $E' \leftarrow F$, and then remove elements one-by-one from E', greedily with respect to maximizing $\phi(J, C, E \cup E')$, until $|E'| = k$. We then take $\{j_1, \ldots, j_k\} = E'$, and determine the ordering $j_1, \ldots, j_k$ by a similar reverse greedy procedure. We found this algorithm much more effective in practice than the forward greedy algorithm.

We now consider the case $k = 1$, in which case our branching scheme generates three children. In this case, we implement an alternative, "strong" branching method: for each possible branching feature $j \in F$, we calculate the bounds $b(J \cup \{j\}, C, E)$, $b(J, C \cup \{j\}, E)$, and $b(J, C, E \cup \{j\})$ of the three resulting subproblems. For each j, we place these bounds in a triple $\mathbf{b}_j$ with elements sorted in descending order, and branch on some feature j that lexically minimizes $\mathbf{b}_j$. This approach is not only faster than the reverse greedy method, but also more practically effective at pruning search nodes. One reason for the efficiency of this alternative branching strategy is that it considers the bounds of all three prospective children, and not just the bound of the child $(J, C, E \cup \{j\}, F \setminus \{j\})$. Algorithm 1 outlines our entire branch-and-bound algorithm; there, our procedure for choosing $j_1, \ldots, j_k$ is called

2. If the answer is "no" then at most 1/3 of the computation paths accept.

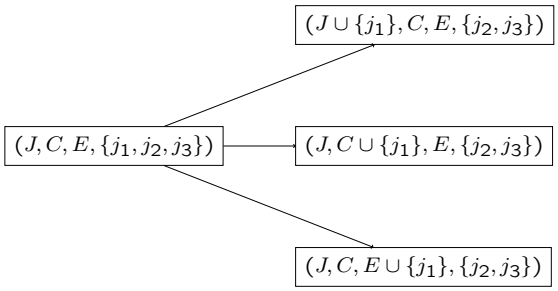
exclude.

Algorithm 1

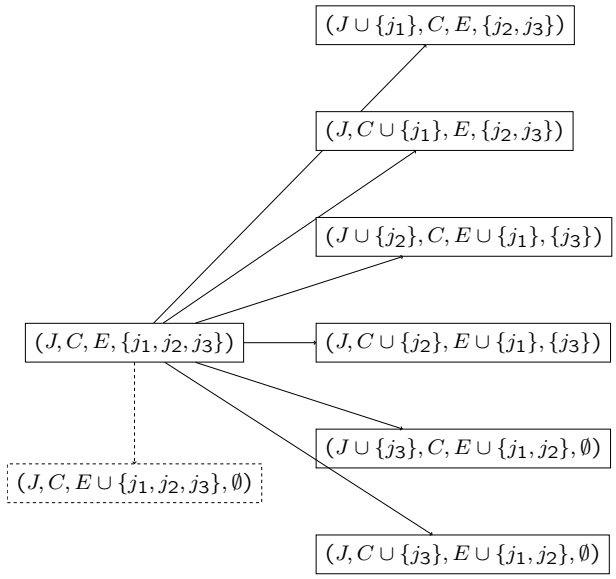
Input: Sets Ω^+, Ω^-, and $M \times N$ matrix A.
Output: Best solution value l and corresponding monomial given by (J^*, C^*).
$Q \leftarrow \{(\emptyset, \emptyset, \emptyset, \{1 \ldots, N\})\}$
$l \leftarrow -\infty$
while $Q \neq \emptyset$ **do**
  Remove subproblem $S = (J, C, E, F)$ from Q
  if $b(J, C, E) > l$ **then**
    if $f(J, C) > l$ **then**
      $(J^*, C^*) \leftarrow (J, C)$
      $l \leftarrow f(J^*, C^*)$
    end if
    $(j_1, ..., j_k) \leftarrow exclude(J, C, F)$
    Insert $(J, C, E \cup \{j_1, ..., j_k\}, F \setminus \{j_1, ..., j_k\})$ into Q
    for $t \in \{1, \ldots, k\}$ **do**
      Insert $(J \cup \{j_t\}, C, E \cup \{j_1, \ldots, j_{t-1}\}, F \setminus \{j_1, \ldots, j_t\})$ into Q
      Insert $(J, C \cup \{j_t\}, E \cup \{j_1, \ldots, j_{t-1}\}, F \setminus \{j_1, \ldots, j_t\})$ into Q
    end for
  end if
end while

The two most extreme branching strategies, with $k = 1$ and with $k = |F|$, are illustrated in Figure 2.2. We note that in the earlier branch-and-bound method of [40], the branching procedure is essentially the special case that k is always chosen as large as possible, that is, $k = |F|$, as illustrated in Figure 2.2(b), and $j_1, \ldots, j_k$ are ordered so that $j_1 < j_2 < \cdots < j_k$. When $k = |F|$, the $(J, C, E \cup \{j_1, \ldots, j_k\}, F \backslash \{j_1, \ldots, j_k\})$ child always represents exactly one monomial, allowing it to be immediately evaluated and implicitly dropped from the search tree. With this branching strategy, it can be shown that the maximum size of the search tree that can evolve, assuming no fathoming, is 3^N, which is exactly the number of possible monomials. Conceptually, this feature is a possible advantage of the method; on the other hand, this approach results in a very high branching factor, especially near the root, and the scheme is essentially static: in the previous branch-and-bound algorithm [40], the set of children developed from a given subproblem depends only on the parameter N.

Here, we make branching decisions that use the input data and are designed to tighten the bounds computed for the resulting child subproblems. Particularly, for $k = 1$, illustrated in Figure 2.2(a), the maximum tree size is $(3^{N+1} - 1)/2 > 3^N$, but the benefits from



(a) Ternary branching with $k = 1$.



(b) Branching with $k = |F|$ implies a $2K + 1$ branching factor. In the previous algorithm [40] the ordering j_1, j_2, j_3 is statically fixed to satisfy $j_1 < j_2 < j_3$.

Figure 2.2: A branching example for a subproblem (J, C, E, F) with $|F| = 3$, that is, three variables that are "free" and the two different extreme branching strategies: $k = 1$ and $k = |F|$.

the smaller branching factor and flexibility of variable selection will become clear in the experiments below.

In lines 15 and 16 of Algorithm 1, we apply a few simple pruning rules that provide further improvement in practical performance: first, we need not insert a subproblem into Q if it is already fathomed. Second, if $\text{Cover}(J \cup \{j_t\}, C) = \text{Cover}(J, C)$, this implies that $m_{J\cup\{j_t\},C}(A_i) = A_{ij_t} m_{J,C}(A_i) = m_{J,C}(A_i)$ for all $i \in \text{Cover}(J, C)$. Thus, $A_{ij_t} = 1$ for all $i \in \text{Cover}(J, C)$. Consider any $(J', C') \in P(J, C, E)$. Now, since $\text{Cover}(J', C') \subseteq \text{Cover}(J, C)$, we have $f(J', C') = f(J' \cup \{j_t\}, C')$. Thus, such a subproblem may be dropped from

consideration without insertion into Q. Finally, a similar result holds for the subproblem $(J, C \cup \{j_t\}, E \cup \{j_1, \ldots, j_{t-1}\}, F \setminus \{j_1, \ldots, j_t\})$ when $\text{Cover}(J, C \cup \{j_t\}) = \text{Cover}(J, C)$.

2.3 Experimental study

We implemented the procedure of Algorithm 1 in C++ using the open-source PEBBL branch-and-bound library [30]. We ran our experiments on a workstation with 3.00 GHz Intel x5400 Xeon processors and 800MHz memory (running only in serial, although PEBBL is capable of parallelism).

Table 2.1 and Figures 2.4-2.8 show our experimental results using Algorithm 1 as a weak learning algorithm inside the LP-Boost boosting procedure [26], applied to the UCI [3] binary dataset SPECTHRT and binarized versions of additional UCI datasets. We configured our binarization procedure to obtain a larger number of variables (as indicated by N in the table) than is customary for most binary feature selection methods (*e.g.* [15]).

The LP-Boost algorithm requires the specification of the "soft margin" penalty parameter D in (1.12). We selected $D = 3/M$, which in our computational experience provides good classification performance. We compare four different algorithm configurations:

- An algorithm equivalent to [40], in which $k = |F|$, using only the b_{gs} upper bound function
- Algorithm 1, with $k = |F|$ and the bound (2.7)
- Algorithm 1, with $k = \lceil |F| / 2 \rceil$ and the bound (2.7).
- Algorithm 1, with $k = 1$, the bound (2.7), and the "strong" lexical $k = 1$ branching rule described above. Note that $k = 1$ corresponds to a ternary search tree.

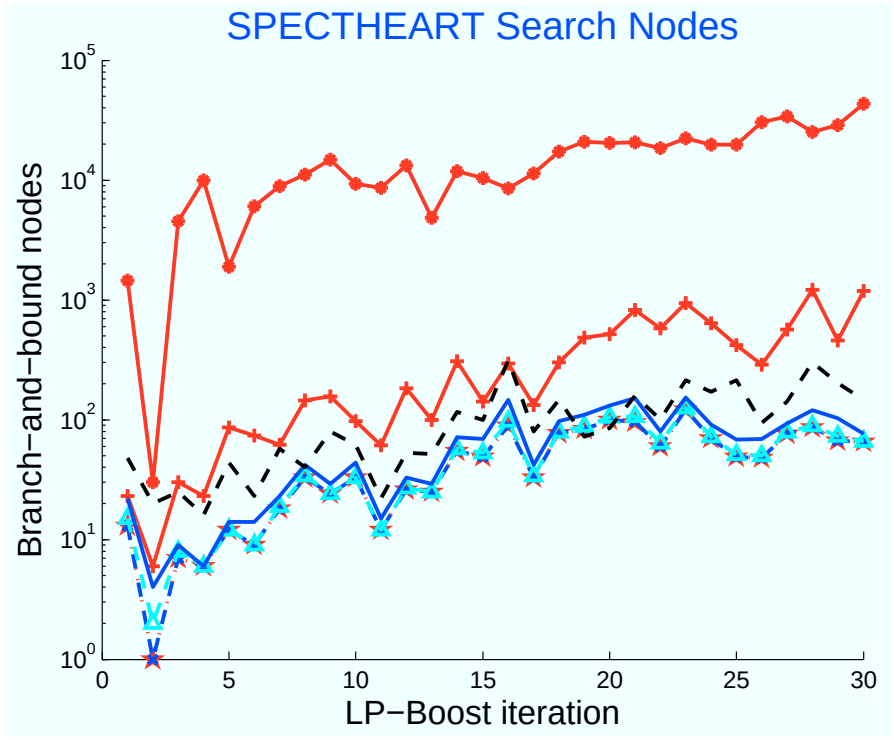
We used a best-first queueing discipline with the queue size limited to at most 500,000 subproblems. We also limited the CPU time of each branch-and-bound search to 60 minutes. LP-Boost starts with the weights $w(i)$ equal for all i, but subsequently adjusts the observation weights based on the dual variables of its LP formulation's separation constraints. We ran each case for 30 iterations, or until a subproblem encountered the queue

size or branch-and-bound time limit. As is well-known in practice, boosting algorithms tend to focus their later iterations on observations that are more difficult to classify. In our case, the later iterations produce longer monomials whose $|\text{Cover}(J, C)|$ is smaller. The later iterations also tend to have larger search trees and longer running times.

Table 2.1 shows the average CPU time and number of search nodes over iterations 1–15 and 16–30, for $k = 1$, $\lceil |F|/2 \rceil$ and $k = |F|$ with the simple bound (2.5) and the bound (2.7). The Figures 2.4–2.8 show a plot of the actual CPU time and number of search nodes over all iterations, and also display the performance for $k = \lceil |F|/4 \rceil$ and $k = \lceil 3|F|/4 \rceil$. Two conclusions appear to follow from the results in Table 2.1 and Figures 2.4–2.8. First, compared with [40], both our tighter bound and our new reverse greedy branching scheme significantly decrease the number of search nodes required; they allow larger problems to be solved, and improve running time in all but the easiest cases involving SPECTHRT. Second, choosing an intermediate number of branching features $k = \lceil |F|/2 \rceil$ performs better in terms of search nodes than $k = 1$. In Figures 2.4–2.8, it is also evident that $k \geq \lceil |F|/2 \rceil$ yields smaller search trees and faster running times than $k = \lceil |F|/4 \rceil$. There does not seem to be any material difference in the performance of algorithm variants with $k = |F|$, $k = 3\lceil |F|/4 \rceil$ and $k = \lceil |F|/2 \rceil$, when ordering the subproblems using the reverse greedy procedure. Finally, although taking $k = 1$ is not the best strategy in terms of search tree size, its specialized, faster branching procedure performs well enough that $k = 1$ is clearly best in terms of runtime. Thus, our new bound coupled with the $k = 1$ ternary branching scheme significantly outperform the method of [40] for larger datasets, and in later boosting iterations, when the weights become more "difficult".

Dataset	LP-Boost Iters	b_{gs} bound $k = \lvert F \rvert$ CPU Sec	b_{gs} bound $k = \lvert F \rvert$ BB Nodes	$k = \lvert F \rvert$ CPU Sec	$k = \lvert F \rvert$ BB Nodes	$k = \lceil \lvert F \rvert / 2 \rceil$ CPU Sec	$k = \lceil \lvert F \rvert / 2 \rceil$ BB Nodes	lexical strong branch $k = 1$ CPU Sec	lexical strong branch $k = 1$ BB Nodes
SPECTHRT	1-15	0.6	7791.5	0.2	21.5	0.2	22.4	0.1	50.5
$N = 22$	16-30	1.8	22751.1	0.4	74.6	0.4	78.5	0.3	162.9
CLVHEART	1-15	29.7	89551.2	16.9	626.2	17.3	654.7	9.5	1638.3
$N = 35$	16-30	99.8	317537.9	40.7	1872.3	41.2	1941.6	23.6	4831.6
HEPATITIS	1-15	17.3	83365.6	7.2	442.5	7.3	464.7	3.2	917.1
$N = 37$	16-30	Q LIMIT		13.5	970.9	13.8	1021.5	7.3	2650.3
PIMA	1-15	Q LIMIT		89.2	1290.5	90.0	1323.2	66.0	4606.3
$N = 33$	16-30	Q LIMIT		269.9	5499.3	269.6	5582.7	224.7	20907.1
CMC	1-15	Q LIMIT		1161.0	969.3	1158.6	972.7	496.9	3647.3
$N = 58$	16-30	Q LIMIT		LIMIT		LIMIT		1738.6	19437.3
HUHEART	1-15	Q LIMIT		LIMIT		LIMIT		264.0	14169.3
$N = 72$	16-30	Q LIMIT		LIMIT		LIMIT		763.6	47657.2

Table 2.1: Runtime and node averages over the specified LP-Boost [26] iterations, applying our algorithm to binarized UCI datasets [3]. "Q LIMIT" indicates an iteration encountered the 500,000-node queue limit, and "LIMIT" indicates an iteration encountered the one-hour time limit.



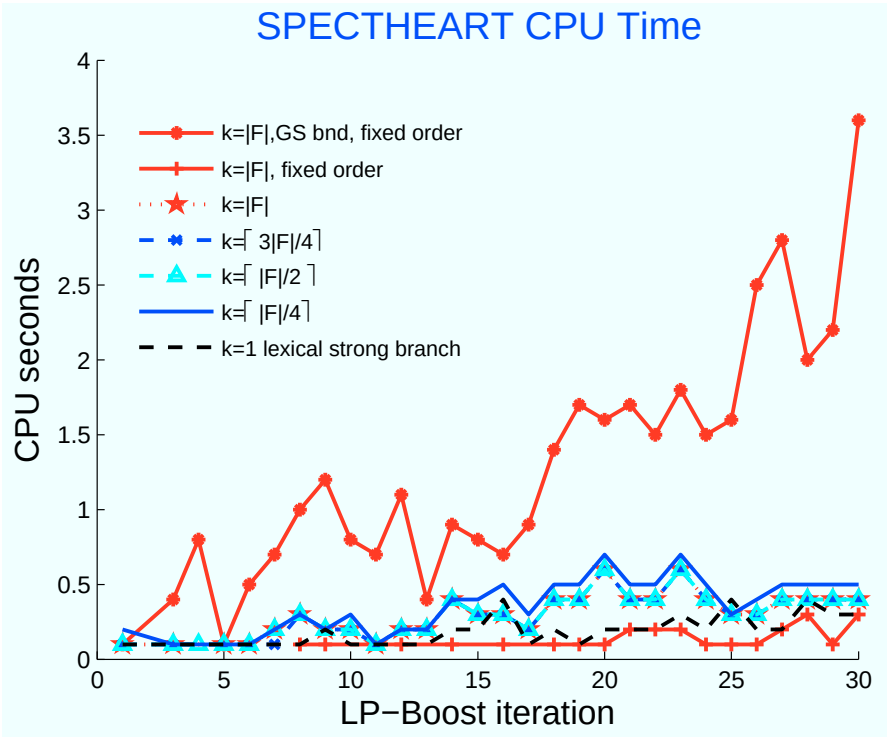
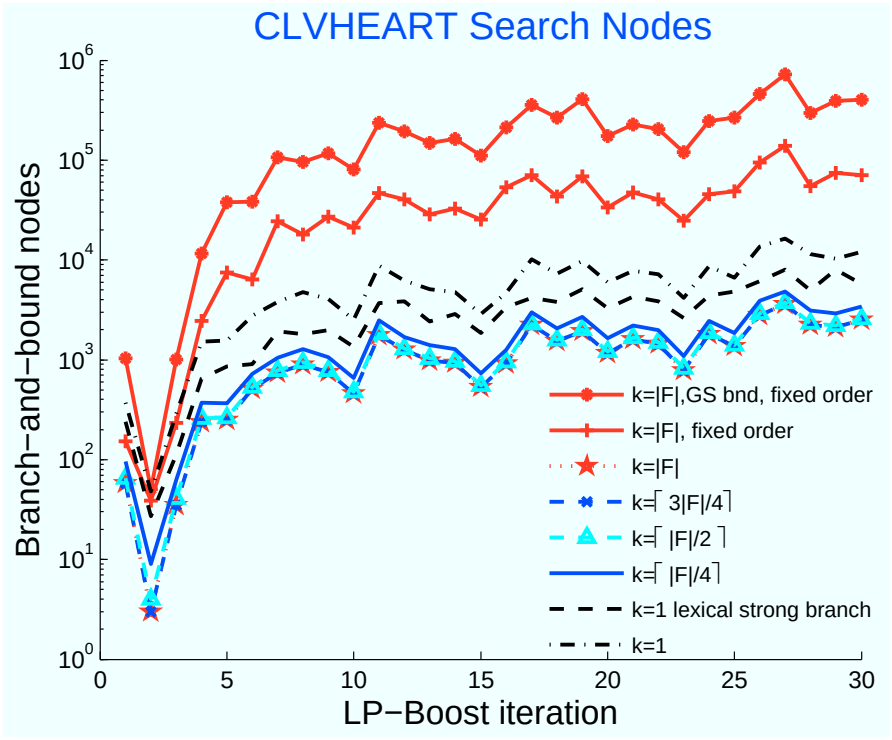


Figure 2.3: MMA computational performance on the UCI SPECTHEART dataset.



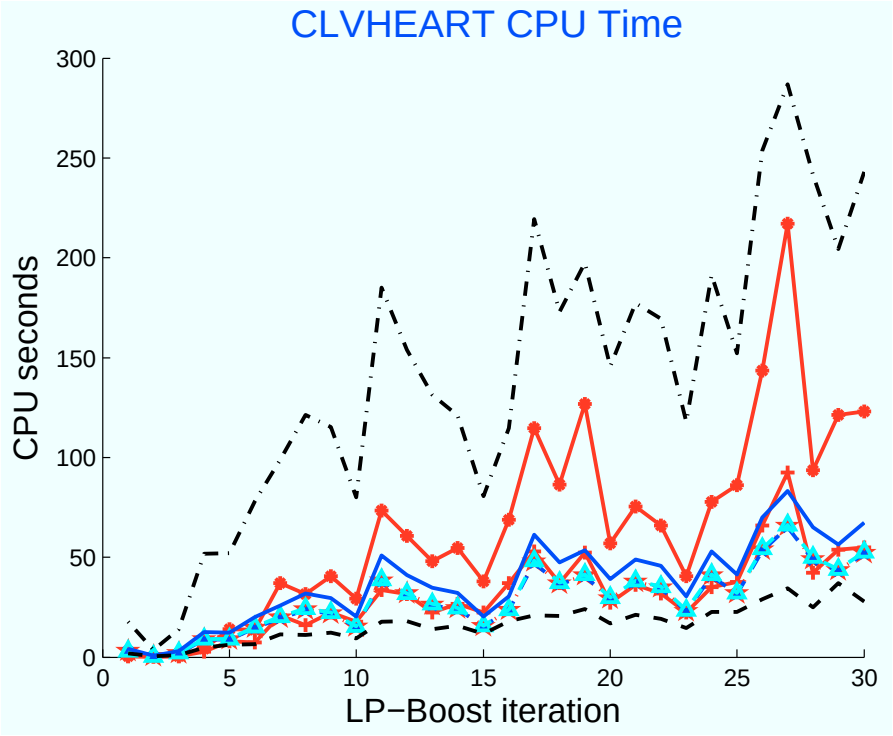


Figure 2.4: MMA computational performance on the UCI CLVHEART dataset.

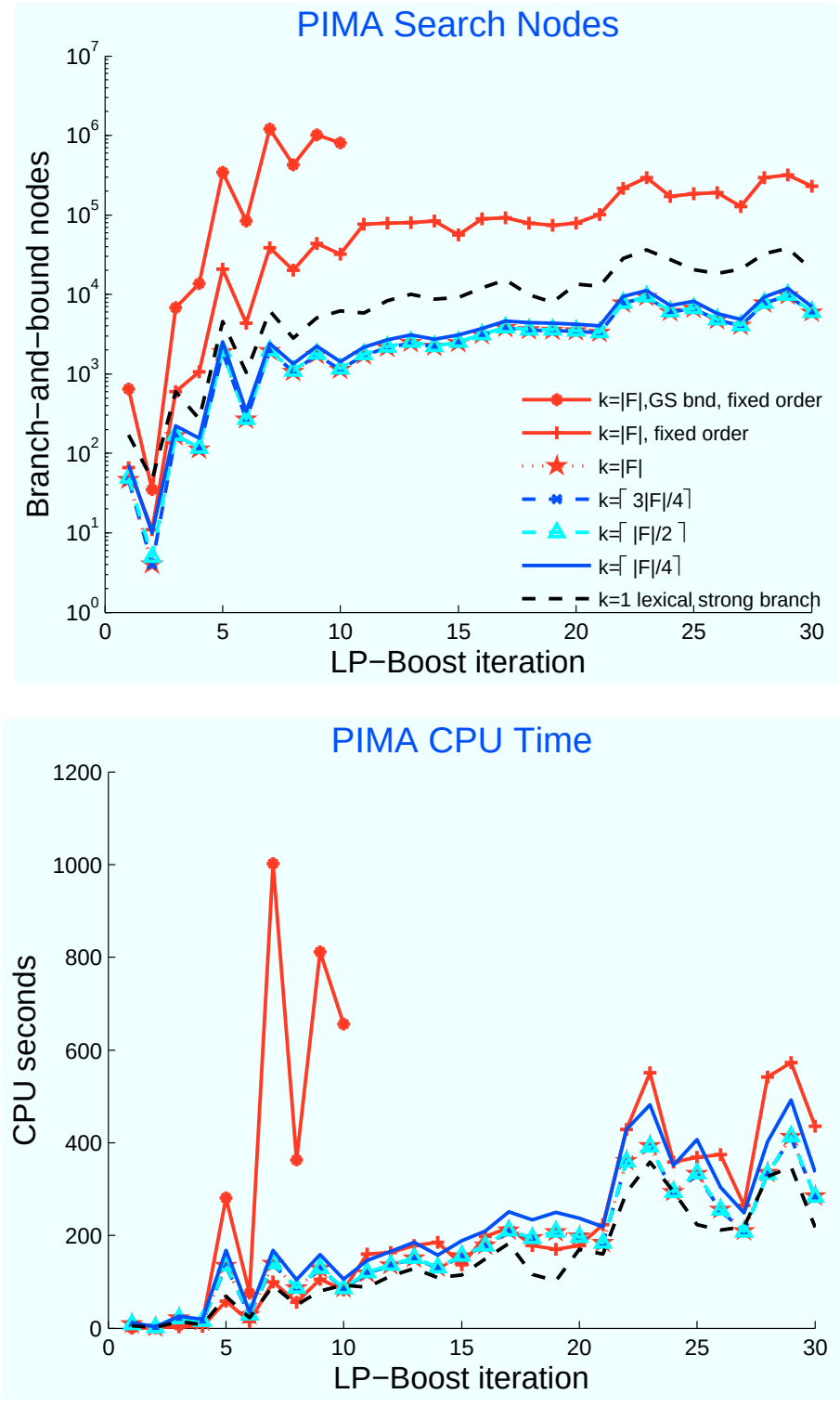
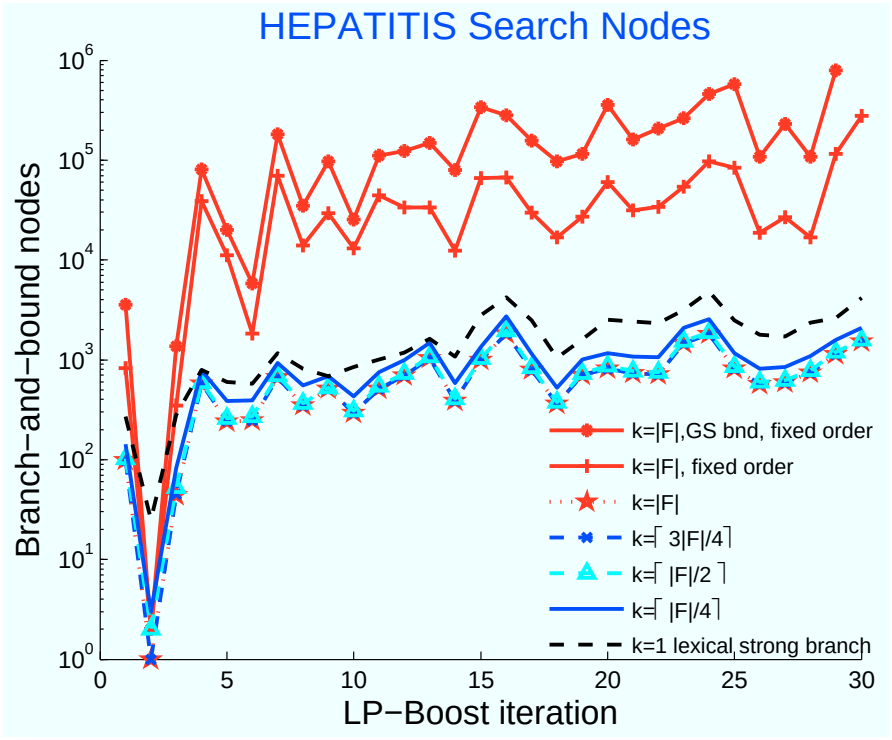


Figure 2.5: MMA computational performance on the UCI PIMA dataset.



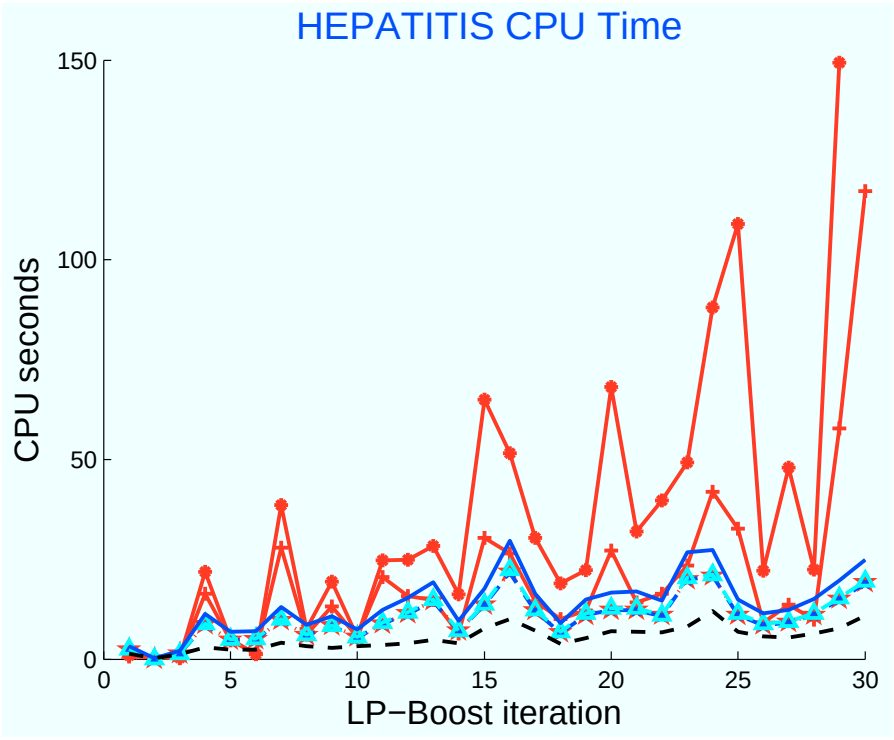


Figure 2.6: MMA computational performance on the UCI HEPATITIS dataset.

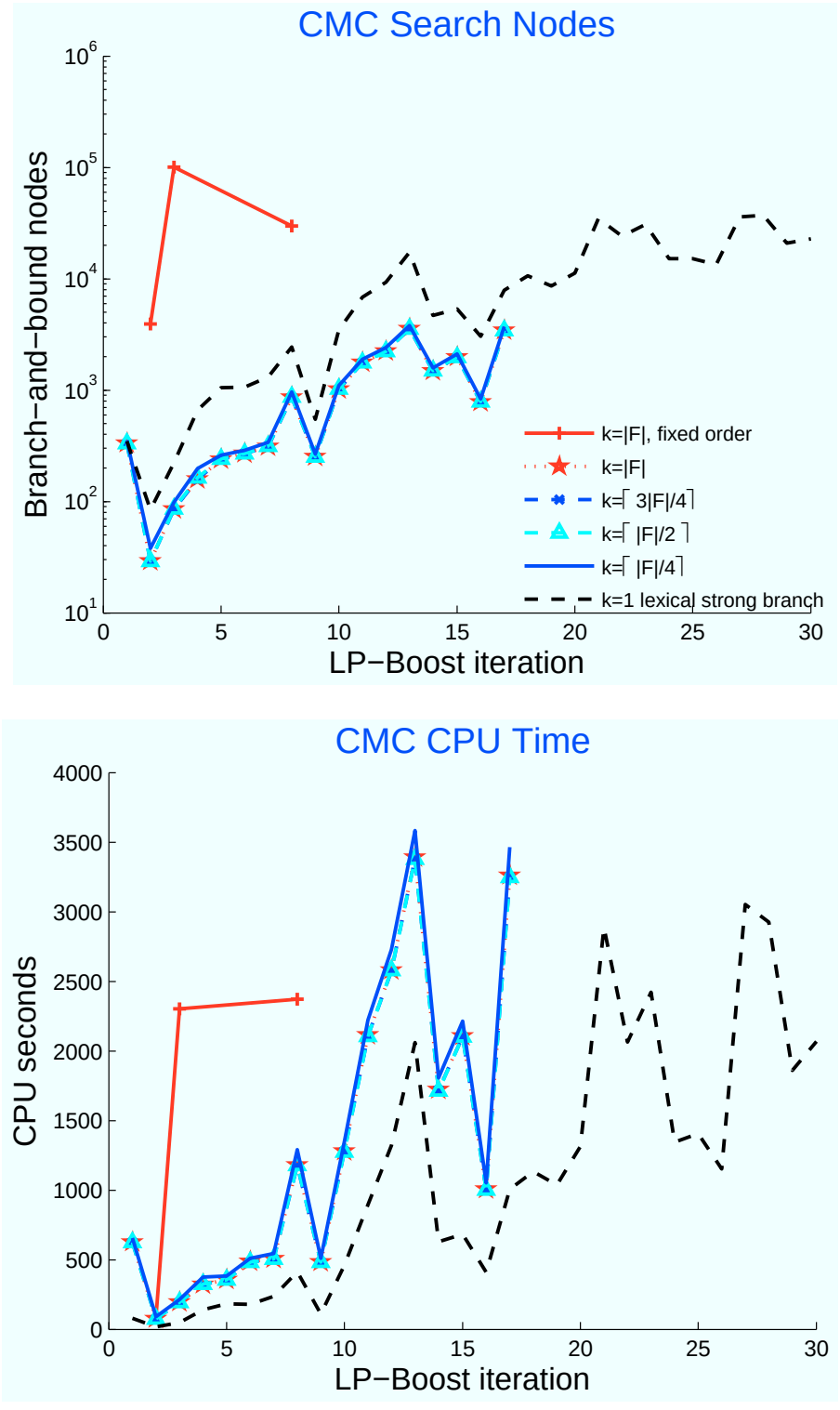


Figure 2.7: MMA computational performance on the UCI CMC dataset.

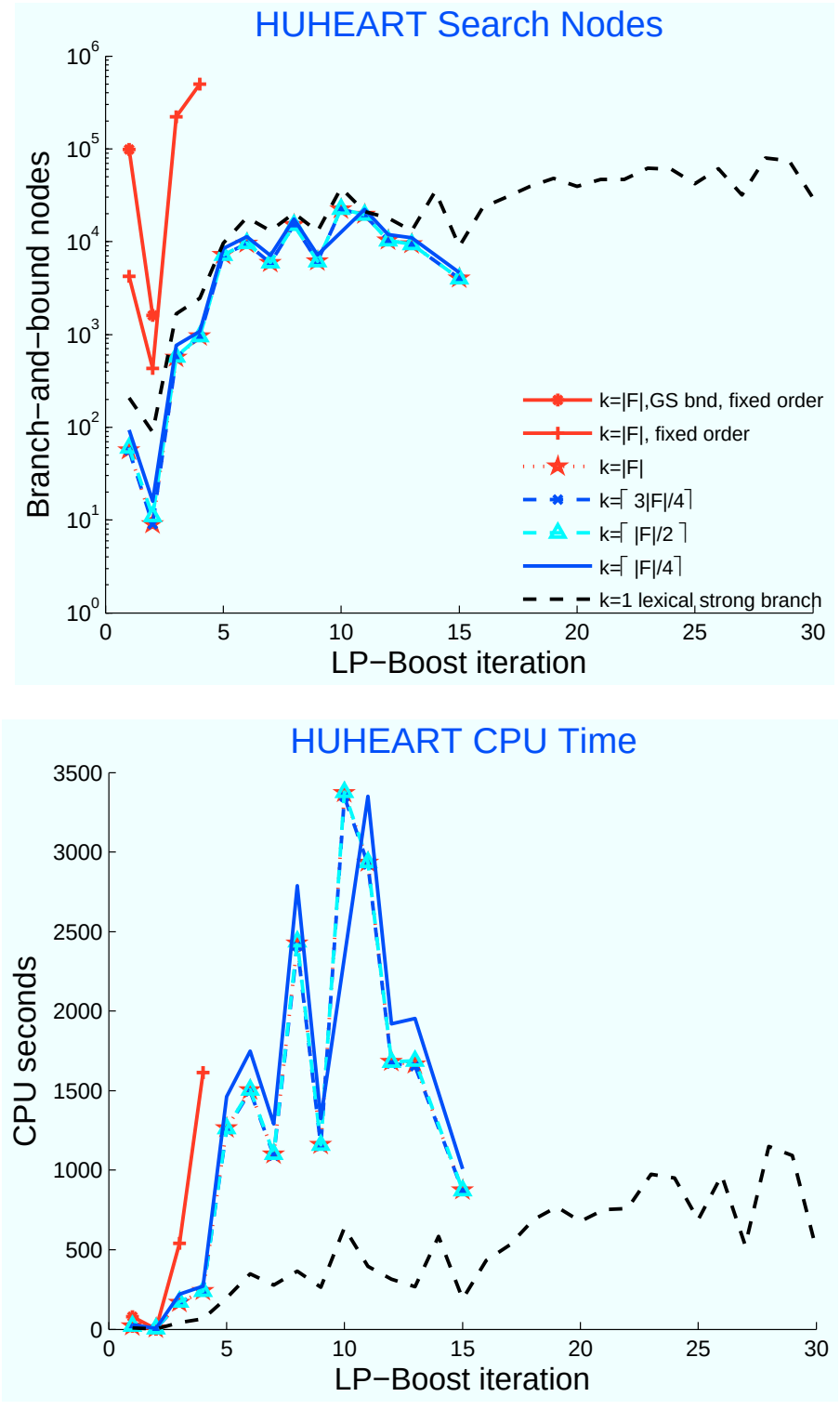


Figure 2.8: MMA computational performance on the UCI HUHEART dataset.

Chapter 3

Tightened L_0 relaxation for classification

In optimization-based classification model selection, for example when using linear programming formulations such as (1.13), the standard approach is to penalize the L_1 norm of a linear functional in order to select sparse models. Instead, we now propose a novel integer linear program for sparse classifier selection, generalizing the minimum disagreement hyperplane problem whose complexity has been investigated in computational learning theory. Specifically, our mixed integer problem is that of finding a separating hyperplane with minimum empirical error subject to an L_0-norm penalty. We show that the common "soft margin" linear programming formulations (1.13) and (1.12) for robust classification are equivalent to the continuous relaxation of our model. Since the initial continuous relaxation is weak, we suggest a tighter relaxation, using novel cutting planes, to better approximate the integer solution. We describe a boosting algorithm, based on linear programming with dynamic generation of cuts and columns, that solves our relaxation. We demonstrate the classification performance of our proposed algorithm with experimental results, and justify our selection of parameters using a minimum description length, compression interpretation of learning.

We will introduce a new combinatorial optimization problem which we call *sparse minimum disagreement hyperplane* (SMDH), and discuss its relation to soft margin maximization (1.13). Before we proceed to do so, we motivate our new problem by relating it to current statistical learning theory summarized in Section 1.3.

3.1 Statistical learning theory justifications for sparsity and minimizing code length

The main objective of statistical learning theory attempts to quantify prediction risk — the probability of error on the test (unseen) data for a given model. Initially Freund and Schapire [36] bounded the prediction risk of weighted voting classifiers in terms of the L_0-norm of λ by bounding the VC dimension of linear combinations of base classifiers in terms of the VC dimension of the base classifiers used, and $||\lambda||_0$, as shown by Theorem 1.3.1. Following later results by Bartlett [6], Schapire *et al.* [63] and Vapnik (see [66] section 5.5.6) providing risk bounds in terms of the margin of separation, algorithms for finding weighted voting classifiers have been mostly motivated by such bounds expressed in terms of the margin or equivalently the L_1 or L_2 norm of λ.

Luxburg, Bousquet and Schölkopf have investigated the connection between statistical learning theory and compression for SVMs [71]. Although SVMs are designed to maximize the margin of separation in the space of features subject to a soft margin penalty, Luxburg *et al.* find that their compression-based bounds often perform better that margin-based bounds. Their analysis relates the idea of compression to the notion of separation margin by showing that the larger the L_2 margin of separation, the lower the precision needed to encode the voting weights λ, and the more the classifier description may be compressed. While Luxburg *et al.* help to illuminate the relation between L_2 margin of separation and compression, their work raises the question whether there can be more direct approaches to "compressing" weighted voting classifiers. For example, are there efficient methods that more directly optimize sparsity and attempt to minimize the length of the classifier description?

Here, we adopt a strategy inspired by the MDL conception of learning and specifically two-part code length minimization. We would like to minimize the total length of both the code describing the classifier g, and the second part encoding which observations are misclassified by g. In the following we formulate such a code, whose length we seek to minimize, thereby we also minimize a risk bound such as (1.11), and thus we expect our selection of a classifier g to "generalize" well, *i.e.*, perform well with respect to unseen data.

Let $\dot{L}[g]$ denote the model code length of weighted voting classifier g. We consider an upper bound on the code length that is a function of subsets of $\mathcal{U}$. Let $\mathcal{G}$ be the set of all weighted voting classifier models. Our objective then is to select a g that minimizes the sum of the model code length function (or an upper bound) $\dot{L}$ of g, and the length of the misclassified data labels given g, that is,

$$\min_{g\in\mathcal{G}} \bar{L}[y,g] = \min_{g\in\mathcal{G}}\{\dot{L}[g] + L[y|g]\},$$

where $L[y|g]$ is the length of a code that encodes the the labels of observations that are misclassified by g, requiring at most $\sum_{i=1}^{M} \mathbf{I}(y_i g(A_i))\lceil \log M\rceil$ bits.

In our approach of implementing structural risk minimization, we suppose that the base classifiers are partitioned into a (finite) K subsets $\mathcal{U}_k$, $k = 1,\ldots,\mathcal{U}_K$, with each $\mathcal{U}_k$ representing the indices of classifiers that have equal complexity or "risk". The corresponding weighted voting classifiers can then be decomposed into subsets $S_j = \text{conv}(\{h_u \mid u \in \bigcup_{k=0}^{j}\mathcal{U}_j\})$, for $j = 1,\ldots,K$, where $\text{conv}(U)$ denotes the convex hull of set U. Each of the sets $\mathcal{U}_k$ corresponds to one of K tables in a *code book* [69], present at both the sender and receiver, and an element of the k^{th} table can be identified using $\lceil \log|\mathcal{U}_k|\rceil$ bits.

To describe the classifier g, we must specify $||\lambda||_0$ such table indices, and also specify which table each element of $\{u \mid \lambda_u > 0\}$ corresponds to, the latter requiring at most $||\lambda||_0 \lceil \log K\rceil$ bits. Having thus identified the features, one must finally encode the weights of the separating hyperplane weights $\lambda \in [0,1]^M$, which we can show requires at most $(||\lambda||_0 + 1)\lceil \log M\rceil$ bits to encode a set of affinely independent *support vectors*. The support vectors (as in SVM) are data points that lie on one of the hyperplanes given by $\sum_{u=1}^{U}\lambda_u h_u(x) = \pm 1$.

Theorem 3.1.1. *For a given set of points represented by the rows of $A \in \mathbb{R}^{M\times N}$, there is a two-part encoding of the hyperplane $g(x) = \lambda_u h_u(x) = 0$ that adds at most $(||\lambda||_0+1)\lceil \log M\rceil$ bits to the code length of $\{u \in \mathcal{U} \mid \lambda_u > 0\}$.*

Proof. Consider the $||\lambda||_0$-dimensional space defined by the features in $\{u \in \mathcal{U} \mid \lambda_u > 0\}$. The number of affinely independent vectors in this space can be at most $||\lambda||_0 + 1$. Thus at most $||\lambda||_0 + 1$ support vectors are needed to define one of the supporting hyperplanes $\sum_{u=1}^{U}\lambda_u h_u(x) = \pm 1$ as their affine combination. The parallel plane going through the

origin is $\sum_{u=1}^{U} \lambda_u h_u(x) = 0$. For any two-part code where the sender and receiver share A_i for $i = 1, \ldots, M$, a data point vector can be identified using $\lceil \log M \rceil$ bits. □

Using this information, a receiver can decode λ. The total length of a code that encodes g is:

$$\begin{aligned}
\dot{L}[g] &= ||\lambda||_0 \lceil \log K \rceil + \sum_{u \in \mathcal{U}: \lambda_u > 0} \lceil \log \left| \mathcal{U}_{k(u)} \right| \rceil + (||\lambda||_0 + 1) \lceil \log M \rceil \\
&= ||\lambda||_0 \left(\lceil \log K \rceil + (1 + 1/||\lambda||_0) \lceil \log M \rceil \right) + \sum_{u \in \mathcal{U}: \lambda_u > 0} \lceil \log \left| \mathcal{U}_{k(u)} \right| \rceil \qquad (3.1) \\
&\leq ||\lambda||_0 \left(\lceil \log K \rceil + 2 \lceil \log M \rceil \right) + \sum_{u \in \mathcal{U}: \lambda_u > 0} \lceil \log \left| \mathcal{U}_{k(u)} \right| \rceil \qquad (3.2) \\
&\leq ||\lambda||_0 \left(\lceil \log K \rceil + 2 \lceil \log M \rceil + \max_{u \in \mathcal{U}: \lambda_u > 0} \lceil \log \left| \mathcal{U}_{k(u)} \right| \rceil \right) \qquad (3.3)
\end{aligned}$$

where $k : \mathcal{U} \to \{1, \ldots, K\}$ and $||\lambda||_0 \geq 1$. Thus, by (3.3), minimizing $||\lambda||_0$ is equivalent to minimizing an upper bound on the code length $\dot{L}[g]$. Also, it follows from (3.2) that there is a tighter bound on the code length which can be written as a function that is linear in the support of λ.

3.2 The sparse minimum disagreement hyperplane problem - a hard problem

3.2.1 Problem formulation

In classification problems, the computational challenge is often associated with the size of $\mathcal{U}$ (also known as the dimension of the *feature space*), as well of the size of the data M.

The problem of finding a hyperplane g that minimizes the sum of (1.2) over $i = 1, \ldots, M$ is known as the *minimum disagreement halfspace* problem (MDH) [44, 2]. When considering the 0/1 loss (1.2), it turns out that even when $\mathcal{U}$ is given in the input, the problem of finding a loss minimizing hyperplane is $\mathcal{NP}$-hard [44, 9]. This problem has been solved using heuristics based on nonlinear programming by Mangasarian [53] and Bennett and Bredensteiner [10]. In their work, both Mangasarian and Bennett and Bredensteiner refer to a hardness result for the same problem that appeared in the PhD thesis of Heath [43]. The

problem of minimizing the classification error is also known to be a case of *maximum feasible subsystem*: the problem of finding a feasible subsystem containing as many inequalities as possible from a given infeasible system $Ax \leq b$ [56].

We would like to extend the minimum disagreement hyperplane problem to penalize the L_0 norm of λ. We call the more general problem including an L_0 penalty the *sparse minimum disagreement hyperplane* problem (SMDH). In the basic version of the problem, we penalize all non-zero components of λ uniformly through the same penalty parameter C. The problem can be stated as:

Sparse Minimum Disagreement Hyperplane (SMDH)

Input: A matrix $H \in \{-1, 0, 1\}^{M \times U}$ of base classifier labels, a vector $y \in \{-1, 1\}^M$ of sample labels, and a penalty $0 \leq C < M$

Problem: To find a separating hyperplane, given by $\lambda \in \mathbb{R}_+^U$, such that $\sum_{i=1}^{M} \mathbf{I}(y_i H_i \lambda < 1) + C \, ||\lambda||_0$ is minimized.

Note that we exclude the case that $C \geq M$ because it leads to the trivial solution $\lambda = 0$. We now formulate SMDH as a Mixed Integer Program (MIP), using binary variables μ_u to indicate whether feature u is used, and binary variables ξ_i to indicate whether observation i is misclassified:

$$\min_{\xi,\mu,\lambda} \left\{ \sum_{i=1}^{M} \xi_i + C \sum_{u=1}^{U} \mu_u \;\middle|\; (\xi, \mu, \lambda) \in Q_{H,y} \cap \left(\{0,1\}^M \times \{0,1\}^U \times \mathbb{R}_+^U \right) \right\}, \tag{3.4}$$

where $Q_{H,y}$ is a "soft margin" classification polytope defined as:

$$Q_{H,y} = \left\{ (\xi, \mu, \lambda) \in [0,1]^M \times [0,1]^U \times \mathbb{R}_+^U \;\middle|\; \begin{array}{l} \operatorname{diag}(y) H \lambda + (MK + 1)\xi \geq \mathbb{1} \\ \lambda \leq K\mu \end{array} \right\},$$

where K is a suitably large constant and $\operatorname{diag}(y)$ is the diagonal matrix with entries $y_1, \ldots, y_M$. We show that (3.4) is correct for all large enough K. Note that SMDH and formulation (3.4) always have a feasible solution. Therefore, since the objective value of any feasible solution to either SMDH or formulation (3.4) must be a nonnegative integer, each must always have an optimal solution. Thus, to prove the equivalence of (3.4) and SMDH, it is sufficient to show in the following theorem that the optimal solution values of SMDH

and (3.4) are equal.

Theorem 3.2.1. *If $K \geq M^{M/2}$, then for any optimal solution $(\xi^*, \mu^*, \lambda^*)$ of (3.4), it must be that $\sum_{i=1}^{M} \xi_i^* + C \sum_{u \in \mathcal{U}} \mu_u^* = \min_{\lambda \in \mathbb{R}_+^N} \sum_{i=1}^{M} \mathbf{I}(y_i H_i \lambda \leq 0) + C \, ||\lambda||_0 = \sum_{i=1}^{M} \mathbf{I}(y_i H_i \lambda^* \leq 0) + C \, ||\lambda^*||_0$.*

To prove this result, we will require the following two Lemmas.

Lemma 3.2.2. *If there exists a hyperplane $g(x) = \sum_{u \in \Gamma} \lambda_u x_u = 0$ that separates $S^+ \subseteq \Omega^+$ and $S^- \subseteq \Omega^-$, for some $\Gamma \subseteq \mathcal{U}$, then there exists λ^* such that $\lambda_u^* \leq M^{M/2}$ for all $u \in \Gamma$, and $g(x) = \sum_{u \in \Gamma} \lambda_u^* x_u = 0$ also separates S^+ and S^-.*

Proof. Let $\hat{y}$ denote the subvector of y with elements $S^+ \cup S^-$ and $\hat{H}$ the submatrix of H with rows $S^+ \cup S^-$ and columns Γ. The points S^+ and S^- are linearly separable if the linear system $\text{diag}(\hat{y})\hat{H}\lambda \geq \mathbb{1}$ has a feasible solution. The system of inequalities $\text{diag}(\hat{y})\hat{H}\lambda \geq \mathbb{1}$ has a solution if it has a basic feasible solution λ^*. Let B denote a basis corresponding to a submatrix of $\text{diag}(\hat{y})\hat{H}$. The basic feasible solution λ^* must satisfy the linear system

$$B\lambda^* = \mathbb{1}.$$

Now, by Cramer's rule and the non-negativity constraints, we have

$$\lambda_u^* = \frac{\det\left(B^{(u)}\right)}{\det(B)} \geq 0$$

where $B^{(u)}$ is the matrix B with column u replaced by $\mathbb{1}$. Since the rank of B is bounded by $|S^+ \cup S^-| \leq M$, we have using Hadamard's bound [17] that $\left|\det\left(B^{(u)}\right)\right| \leq M^{M/2}$. Since B is a basis, $\det(B) \neq 0$. Thus, by the definition of a determinant and since $B_{ij} \in \{-1, 0, 1\}$,

$$|\det(B)| = \left| \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} B_{i\sigma(i)} \right| \geq 1,$$

and the claim follows. □

Lemma 3.2.3. *If there exists a hyperplane $\sum_{t \in \Gamma} \lambda_t h_t(x) = 0$ that separates $S^+ \subseteq \Omega^+$ and $S^- \subseteq \Omega^-$, then there exists λ^* such that $\sum_{t \in \Gamma} \lambda_t^* h_t(x) = 0$ separates S^+ and S^- and*

$||\lambda^*||_0 \leq \text{rank}(\tilde{H}) \leq |S^+| + |S^-|$, *where* $\tilde{H}$ *is the submatrix of* H *with columns* Γ *and rows* $S^+ \cup S^-$.

Proof. Let

$$Y = \begin{bmatrix} y_{i_1}\tilde{H}_{i_1} \\ \vdots \\ y_{i_k}\tilde{H}_{i_k} \end{bmatrix},$$

where $\{i_1, \ldots, i_k\} = S^+ \cup S^-$. By feasibility of λ, the polyhedron

$$\left\{ \lambda \in \mathbb{R}_+^T \;\middle|\; \sum_{t\in\Gamma} y_i H_{it}\lambda_t \geq \mathbb{1} \text{ for } i \in S^+ \cup S^- \right\} \tag{3.5}$$

has a vertex corresponding to a basic feasible solution λ^*.

The number of nonzero components of each basic feasible solution is at most the rank of the constraint matrix, $\text{rank}(Y)$, and $\text{rank}(Y) = \text{rank}(\tilde{H})$ since the rank of $\tilde{H}$ will not change if one simply negates some rows. Thus,

$$||\lambda^*||_0 = |\{t \in \Gamma \mid \lambda_t^* > 0\}| \leq \text{rank}(Y) = \text{rank}(\tilde{H}) \leq |S^+| + |S^-|. \qquad \square$$

Proof of Theorem 3.2.1. By the constraints $\lambda_u^* \leq K\mu_u^*$ of formulation (3.4),

$$||\lambda^*||_0 \leq \sum_{u\in\mathcal{U}} \mu_u^*.$$

By the constraint $\text{diag}(y)H\lambda^* + (MK+1)\xi \geq \mathbb{1}$, it follows that $\sum_{i=1}^M \mathbf{I}(y_i H_i \lambda^* \leq 0) \leq \sum_{i=1}^M \xi_i^*$. Thus, by optimality of λ for SMDH, it follows that

$$\sum_{i=1}^M \mathbf{I}(y_i H_i \lambda \leq 0) + C\,||\lambda||_0 \leq \sum_{i=1}^M \mathbf{I}(y_i H_i \lambda^* \leq 0) + C\,||\lambda^*||_0 \leq \sum_{i=1}^M \xi_i^* + C\sum_{u\in\mathcal{U}} \mu_u^*. \tag{3.6}$$

We now prove the reverse inequality between the first and last quantities. The feasibility of λ for SMDH implies the linear separability of the subsets $S^+ = \{i \in \Omega^+ \mid \mathbf{I}(y_i H_i \lambda > 0)\}$ and $S^- = \{i \in \Omega^- \mid \mathbf{I}(y_i H_i \lambda > 0)\}$.

Now by Lemma 3.2.3, there exists λ' such that $||\lambda'||_0 \leq |S^+| + |S^-| \leq M$ and λ' separates

S^+ and S^-. By the optimality of λ for SMDH, we also have $||\lambda||_0 \leq M$.

By Lemma 3.2.2 with $\Gamma = \{u \in \mathcal{U} \mid \lambda'_u > 0\}$, there exists λ'', with $\lambda''_u \leq M^{M/2} \leq K$ for all $u \in \mathcal{U}$, with the same support as λ', separating S^+ and S^-. Thus, $|y_i H_i \lambda''| \leq M^{M/2+1}$ for all $i = 1, \ldots, M$. Now let

$$\xi''_i = \begin{cases} 1 & \text{if } y_i H_i \lambda'' < 1 \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mu''_u = \begin{cases} 1 & \text{if } \lambda''_u > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Then, for all $i \in \{1, \ldots, M\}$,

$$y_i H_i \lambda'' + \left(M^{M/2+1} + 1\right) \xi''_i \geq 1.$$

Thus, $(\xi'', \mu'', \lambda'')$ is feasible for (3.4), and $\sum_{i=1}^M \xi''_i = \sum_{i=1}^M \mathbf{I}(y_i H_i \lambda < 1)$, so, by the optimality of $(\xi^*, \mu^*, \lambda^*)$,

$$\sum_{i=1}^M \xi^*_i + C \sum_{u \in \mathcal{U}} \mu^*_u \leq \sum_{i=1}^M \xi''_i + C \sum_{u \in \mathcal{U}} \mu''_u \leq \sum_{i=1}^M \mathbf{I}(y_i H_i \lambda < 1) + C \, ||\lambda||_0 .$$

Thus, all the relations in (3.6) hold with equality, and thus λ^* is also an optimal solution to SMDH. □

We next show that the continuous relaxation of (3.4), which can be stated as

$$\min \left\{ \textstyle\sum_{i=1}^M \xi_i + C \sum_{u=1}^U \mu_u \mid (\xi, \mu, \lambda) \in Q_{H,y} \right\}, \tag{3.7}$$

is equivalent to the soft margin formulation (1.13) with appropriate choices of the penalties C and D. Formulation (1.13) is known to be equivalent to the soft margin maximization formulation (1.12) by Ratsch *et al.* [59] and Bennett *et al.* [11], so that it follows that (1.12) is also equivalent to the relaxation of the SMDH problem relaxation (3.7). The theorem will

also enable us to claim in Section 3.4 that our continuous relaxation formulation provides a tightened relaxation of the discrete SMDH formulation (3.4), by introducing novel cutting planes for the "soft margin" formulation.

Theorem 3.2.4. *(ξ, λ) is an optimal solution of (1.13) if and only if $(\xi/(MK+1), \lambda/K, \lambda)$ is an optimal solution of (3.7) with penalty $C = 1/(D(M + 1/K))$.*

Proof. First we show that every feasible solution (ξ, λ) of (1.13) corresponds to a feasible solution $(\xi/(MK+1), \lambda/K, \lambda)$ of (3.7) with objective value $\frac{1}{D(MK+1)} \left(\sum_{i=1} D\xi_i + \sum_{u=1}^{U} \lambda_u \right)$.

Assume (ξ, λ) is a feasible solution of (1.13). Now,

$$\operatorname{diag}(y)H\lambda + \xi = \operatorname{diag}(y)H\lambda + (MK+1)I\xi/(MK+1) \geq \mathbb{1}, \tag{3.8}$$

and $\mu = \lambda/K$ imply that $(\xi/(MK+1), \lambda/K, \lambda)$ is feasible for (3.7).

Letting $C = 1/(D(M+1/K))$, the objective value of the solution $(\xi/(MK+1), \lambda/K, \lambda)$ of (3.7) is

$$\sum_{i=1}^{M} \xi_i/(MK+1) + \frac{1}{D(M+1/K)} \sum_{u=1}^{U} \lambda_u/K = \frac{1}{D(MK+1)} \left(D \sum_{i=1}^{M} \xi_i + \sum_{u=1}^{U} \lambda_u \right). \tag{3.9}$$

Since the map $(\xi, \lambda, D) \mapsto (\xi/(MK+1), \lambda, 1/(D(M+1/K)))$ is a bijection (where $\mu = \lambda/K$ is fixed), and following the equality in (3.8) and (3.9), the converse also follows.

In (3.7), since the nonnegative variables μ_u have positive objective coefficients, each appears only in the constraint $\mu_u \geq \lambda_u/K$, and the objective is being minimized, all optimal solutions of (3.7) must satisfy $\mu = \lambda/K$. The claim then follows since the objective value of all feasible solutions under the bijective correspondence given by $(\xi, \lambda, D) \mapsto (\xi/(MK+1), \lambda, 1/(D(M+1/K)))$, where $\mu = \lambda/K$, is scaled by the constant $1/(D(MK+1))$. □

The objective function of (3.4) minimizes misclassification plus a complexity penalty proportional to the number of features used. Thus, the SMDH formulation (3.4) corresponds to minimizing an upper bound on the total code length when the features being combined have equal complexity penalties. However, we may wish to assign different penalties c_u to different features $u \in \mathcal{U}$ (but most likely use the same c_u for all u in the same $\mathcal{U}_k$). Moreover,

the SMDH formulation (3.4) may have many alternate solutions: even with respect to a fixed ξ and μ, there may be multiple feasible hyperplanes $\sum_{u\in\mathcal{U}:\mu_u=1}\lambda_u h_u(x)=0$; we may wish to discern between these hyperplanes on the basis of the margin of separation $0<\rho\leq 1$. We generalize our formulation to include penalties that vary with u and to accept the margin ρ as a parameter; to make this parameterization sensible, we must normalize the weights λ, which requires an additional constraint. Finally, for reasons that should become clear in course, we give the formulation with respect to some subset of features $\Gamma\subseteq\mathcal{U}$:

$$\min\left\{\sum_{i=1}^{M}\xi_i+\sum_{u\in\Gamma}c_u\mu_u \;\middle|\; (\xi,\mu,\lambda)\in Q_{H,y,\rho}(\Gamma)\cap\left(\{0,1\}^M\times\{0,1\}^{|\Gamma|}\times\mathbb{R}_+^{|\Gamma|}\right)\right\}, \quad (3.10)$$

where $Q_{H,y,\rho}(\cdot)$ is a soft margin classification polytope with a required margin ρ:

$$Q_{H,y,\rho}(\Gamma)=$$
$$\left\{(\xi,\mu,\lambda)\in[0,1]^M\times[0,1]^{|\Gamma|}\times\mathbb{R}_+^{|\Gamma|} \;\middle|\; \begin{array}{ll}\sum_{u\in\Gamma}y_iH_{iu}\lambda_u+(1+\rho)\xi_i\geq\rho & ,\ i=1,\ldots,M\\ \sum_{u\in\Gamma}\lambda_u=1 & \lambda\leq\mu\end{array}\right\}.$$

3.2.2 Computational complexity

The SMDH problem generalizes the MDH problem, so that it is at least as hard to solve computationally. Specifically, the MDH problem is solved by (3.4) with $C=0$. In the following we will refer to a solution (ξ,μ,λ) of (3.4), with $C=0$, as an MDH solution. Höffgen, Simon and Van Horn [44] show that the MDH problem is not possible to approximate within a factor better than $(1-\epsilon)\log M$ for any $\epsilon>0$ unless $\mathcal{NP}\subseteq\mathcal{DTIME}(M^{\log\log M})$, where $\mathcal{DTIME}(n)$ is the class of problems that can be solved in deterministic time n. Arora, Babai, Stern and Sweedyk [2] improved the inapproximability factor to $2^{\log^{0.5-\epsilon}M}$, making the weaker assumption that $\mathcal{NP}\not\subseteq\mathcal{DTIME}(M^{\mathrm{poly}(\log M)})$. By the restriction of SMDH with $C=0$, the inapproximability result of Arora *et al.* directly applies. We state in the following theorem a more general result for SMDH with constant penalties, based on the inapproximability of MDH [2]. Note that if $C\geq M$, then SMDH has the trivial solution $\lambda=0$ and $\xi_i=1$ for all $i\in\{1,\ldots,M\}$. We will require the following Lemmas to derive our inapproximability result:

Lemma 3.2.5. *Given an MDH instance with input $H \in \{-1,0,1\}^{M\times U}$, $y \in \{-1,1\}^M$, and some constant C, then for all $k \geq \lceil C \rceil M + 1$, there exists an $O(k\,\mathrm{poly}(M,U))$ reduction to an SMDH instance $H' \in \{-1,0,1\}^{Mk\times U}$ and $y' \in \{-1,1\}^{Mk}$, such that MDH has an optimal solution $(\hat{\xi},\hat{\mu},\hat{\lambda})$ if and only if SMDH has an optimal solution (ξ^*,μ^*,λ^*), where $\sum_{i=1}^{Mk}\xi_i^* = k\sum_{i=1}^{M}\hat{\xi}_i$ and $\sum_{u=1}^{U}\mu_u^* \leq M$.*

Proof. Given the input (H,y) of MDH, and a constant C, construct an instance of SMDH (H',y'), by creating k duplicates of H_i in the matrix H', and k duplicates of y_i in the vector y', for each row H_i of H. Without loss of generality, assume that the rows of H' and y' are indexed such that $H_i = H'_i$ and $y_i = y'_i$ for $i = 1,\ldots,M$. Let (ξ^*,μ^*,λ^*) be an optimal SMDH solution for the input (H',y') with penalty C. Let $(\hat{\xi},\hat{\mu},\hat{\lambda})$ be an optimal solution of MDH, corresponding to formulation (3.4) for the input (H,y) with penalty $C = 0$, and value $z_{\mathrm{MDH}} = \sum_{i=1}^{M}\hat{\xi}_i + 0\sum_{u=1}^{U}\hat{\mu}_u = \sum_{i=1}^{M}\hat{\xi}_i$.

Now, since feasible solutions of MDH and SMDH must always exist, we only need to prove $z_{\mathrm{MDH}} = \sum_{i=1}^{M}\hat{\xi}_i = \sum_{i=1}^{Mk}\xi_i^*/k$. Assume to the contrary

$$z_{\mathrm{MDH}} \neq \sum_{i=1}^{Mk}\xi_i^*/k.$$

First, we note that we must have $z_{\mathrm{MDH}} \leq \sum_{i=1}^{Mk}\xi_i^*/k$, for otherwise $\sum_{i=1}^{M}\xi_i^* < z_{\mathrm{MDH}}$, which contradicts the optimality of z_{MDH}. On the other hand, if $z_{\mathrm{MDH}} < \sum_{i=1}^{Mk}\xi_i^*/k$, then

$$z_{\mathrm{MDH}} \leq \sum_{i=1}^{Mk}\xi_i^*/k - 1 \tag{3.11}$$

By Lemma 3.2.3, since the sets $\{i \in \Omega^+ \mid \hat{\xi}_i = 0\}$ and $\{i \in \Omega^- \mid \hat{\xi}_i = 0\}$ are linearly separable, there exists λ corresponding to a hyperplane that separates $\{i \in \Omega^+ \mid \hat{\xi}_i = 0\}$ and $\{i \in \Omega^- \mid \hat{\xi}_i = 0\}$, with $||\lambda||_0 \leq M$. Let

$$\mu_u = \begin{cases} 1 & \text{if } \lambda_u \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Then the optimal SMDH solution value z_{SMDH} must satisfy

$$\begin{aligned}
z_{\text{SMDH}} &= \sum_{i=1}^{Mk} \xi_i^* + C \sum_{u=1}^{U} \mu_u^* \\
&\leq k \sum_{i=1}^{M} \hat{\xi}_i + C \sum_{u=1}^{U} \mu_u && \text{[by optimality of SMDH]} \\
&\leq k z_{\text{MDH}} + CM \\
&< k \left(z_{\text{MDH}} + 1 \right) && \text{[by } k \geq \lceil C \rceil M + 1\text{]} \\
&\leq \sum_{i=1}^{Mk} \xi_i^* && \text{[by (3.11)]} \\
&\leq z_{\text{SMDH}}
\end{aligned}$$

From this contradiction, we conclude that $z_{\text{MDH}} = \sum_{i=1}^{Mk} \xi_i^* / k = \sum_{i=1}^{M} \xi_i^*$. □

Lemma 3.2.6. *An $\alpha(M)$ approximation factor for SMDH with constant penalty parameter C, for some $\alpha : \mathbb{N}_+ \to \mathbb{R}_+$ implies an $\alpha(M(\lceil C \rceil M + 1))\beta$ approximation of MDH for some $\beta \in O(1)$.*

Proof. We will reduce and MDH instance (H, y) to SMDH with a constant C using the reduction of Lemma 3.2.5, making $k = \lceil C \rceil M + 1$ duplicates of each row of H and y, in H' and y' respectively.

Suppose there is an $\alpha(kM) = \alpha(M(\lceil C \rceil M + 1))$-factor approximation algorithm for SMDH. Let (ξ, μ, λ) and $(\xi^*, \mu^*, \lambda^*)$ denote an SMDH solution of the approximation algorithm and optimal solution, respectively. Let $(\hat{\xi}, \hat{\mu}, \hat{\lambda})$ denote an optimal MDH solution, and z_{MDH} denote its value. Without loss of generality we may assume $\sum_{i=1}^{M} \hat{\xi}_i \geq \kappa$, for a constant $\kappa \in \mathbb{N}_+$. Otherwise, we can solve MDH exactly by excluding each subset up to size κ and finding the corresponding separating hyperplanes by linear programming in polynomial time. Now,

$$\begin{aligned}
z_{MDH} = \sum_{i=1}^{M} \hat{\xi}_i &= \sum_{i=1}^{M(\lceil C \rceil M + 1)} \xi_i^* / (\lceil C \rceil M + 1) && \text{[by Lemma 3.2.5]} \\
&\leq \left(\sum_{i=1}^{M(\lceil C \rceil M + 1)} \xi_i^* + C \sum_{u=1}^{U} \mu_u^* \right) / (\lceil C \rceil M + 1)
\end{aligned}$$

$$
\begin{aligned}
&\leq \left(\sum_{i=1}^{M(\lceil C \rceil M+1)} \xi_i + C \sum_{u=1}^{U} \mu_u \right) / (\lceil C \rceil M + 1) \\
&\leq \alpha(M(\lceil C \rceil M + 1)) \left(\sum_{i=1}^{M(\lceil C \rceil M+1)} \xi_i^* + C \sum_{u=1}^{U} \mu_u^* \right) / (\lceil C \rceil M + 1) \quad \text{[by assumption]} \\
&\leq \alpha(M(\lceil C \rceil M + 1)) \left((\lceil C \rceil M + 1) \sum_{i=1}^{M} \hat{\xi}_i + CM \right) / (\lceil C \rceil M + 1) \quad \text{[by Lemma 3.2.5]} \\
&\leq \alpha(M(\lceil C \rceil M + 1))(\sum_{i=1}^{M} \hat{\xi}_i + 1) \\
&\leq \alpha(M(\lceil C \rceil M + 1))(1 + 1/\kappa) \sum_{i=1}^{M} \hat{\xi}_i,
\end{aligned}
$$

which yields an $\alpha(M(\lceil C \rceil M + 1))(1 + 1/\kappa) = \alpha(M(\lceil C \rceil M + 1))\beta$-factor approximation for MDH, with $\beta \in O(1)$. □

Theorem 3.2.7. *The SMDH problem, with a constant C, cannot be approximated to within any constant factor, assuming $\mathcal{P} \neq \mathcal{NP}$.*

Proof. By Lemma 3.2.6 a constant factor approximation for SMDH yields a constant factor approximation for MDH and a contradiction [44, 2] □

Theorem 3.2.8. *For any constant penalty C and $\epsilon > 0$, the SMDH problem cannot be approximated within a factor of $2^{\log^{0.5-\epsilon} M}$ unless $\mathcal{NP} \subseteq \mathcal{DTIME}(M^{\text{poly}(\log M)})$.*

Proof. By Lemma 3.2.6, a $2^{\log^{0.5-\epsilon} M}$-factor approximation for SMDH, for some $\epsilon > 0$, yields a $\beta 2^{\log^{0.5-\epsilon}(M(\lceil C \rceil M+1))}$-factor approximation for MDH, for some $\beta \in O(1)$. Now, because $C \leq M$,

$$\beta 2^{\log^{0.5-\epsilon}(M(\lceil C \rceil M+1))} \leq \beta 2^{\log^{0.5-\epsilon} M^4} \beta 2^{4^{0.5-\epsilon} \log^{0.5-\epsilon} M} \leq 2^{\log^{0.5-\epsilon'} M}$$

for some $0 < \epsilon' \leq \epsilon$, $M_0 \in \mathbb{N}_+$ and all $M \geq M_0$. This is a contradiction with the inapproximability of MDH [2]. □

The continuous relaxations of models (3.4) and (3.10) can be quite weak. First we elaborate the on the weakness of the relaxation, and then we suggest novel cutting planes for the purpose of strengthening the relaxation.

3.3 Relaxing the hard problem and strengthening the relaxation

The weakness of the continuous relaxations of models (3.4) and (3.10) is due to a large *integrality gap*. The integrality gap of a MIP relaxation is defined as $\sup_{H,y} z(H,y)/z_R(H,y)$, where $z(H,y)$ and $z_R(H,y)$ are the optimal solution values of the SMDH MIP and its continuous relaxation, respectively (see Vazirani [70]).

In order to show a lower bound for the integrality gap we will consider a particular construction of a simple SMDH instance with $C = 1$ and $\text{diag}(y)H = I$, where I is the identity matrix, meaning each classifier covers only a single observation. Since each observation $i \in \{1, \ldots, M\}$ must be either classified correctly by the single classifier u with $y_i H_{iu} = 1$ and $\mu_u = 1$, or otherwise $\xi_i = 1$, this instance has an optimal integer solution of value M, where M of the μ_u and ξ_i variables assume a value of one and all of the remaining variables are zero. The relaxation of MDH, however, has the feasible solution $\xi_i = 1/(MK+1)$ for $i = 1, \ldots, M$, and $\mu = 0$, with value $M/(MK+1)$. In Lemma 3.2.2, we proved a large upper bound for the required constant K, *i.e.*, formulation (3.4) is correct for all $K \geq M^{M/2}$. Smaller values of K that maintain the correctness of (3.4) may be possible. However, we can show a lower bound for any constant K that maintains the correctness of the formulation by constructing the following SMDH instance (different than the simple instance above used to demonstrate the gap): let

$$\text{diag}(y)H = \begin{pmatrix} 1 & 0 & \ldots & & 0 \\ -1 & 1 & 0 & \ldots & 0 \\ 0 & -1 & \ddots & & 0 \\ & & -1 & 1 & 0 \\ -1 & \ldots & & -1 & 1 \end{pmatrix}.$$

Now, in order to admit the feasible SMDH solution $\lambda_1 = 1$, $\lambda_2 = 3$, $\ldots$,$\lambda_{M-1} = M-1$, $\lambda_M = 0$, $\xi_1 = \ldots = \xi_{M-1} = 0$, $\xi_M = 1$ formulation (3.4) must have $K \geq M(M-1)/2$. Thus, the integrality gap of SMDH satisfies

$$\sup_{H,y} \frac{z(H,y)}{z_R(H,y)} \geq \frac{M}{M/(M(M(M-1)/2)+1)} \geq M^2(M-1)/2.$$

Using the same simple SMDH instance, with $C = 1$ and $\text{diag}(y)H = I$, we can also show a large lower bound factor of M for the MIP formulation (3.10). In the case of (3.10), due to the normalization constraint, the relaxation solution may have $\mu_u = \frac{1}{M}$ for all $u \in \mathcal{U}$, compared with the integer solution having $\mu_u = 1$ for all $u \in \mathcal{U}$. This instance, therefore, proves the integrality gap lower bound for (3.10), with

$$\frac{\sup_{H,y} z(H,y)}{z_R(H,y)} \geq \frac{\sum_{i=1}^{M} 1}{\sum_{i=1}^{M} 1/M} = M.$$

We now consider adding valid inequalities to (3.10) in order to strengthen its relaxation. We say that a base classifier h *distinguishes* between a pair (i, i') if it classifies them differently but classifies at least one of them correctly, *e.g.*, $h_u(A_i) = y_i \neq h_u(A_{i'})$. Let $S_{i,i'} = \{u \in \mathcal{U} \mid h_u(A_i) = y_i \neq h_u(A_{i'})\}$ denote the set of base classifiers that correctly classify observation i and distinguish it from i'. We consider the following inequality for each pair of observations $(i, i') \in (\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+)$:

$$\xi_i + \xi_{i'} + \sum_{u \in S_{i,i'} \cap \Gamma} \mu_u \geq 1. \tag{3.12}$$

Intuitively, such a cutting plane implies that either we misclassify at least one of the of the observations i or i', or we need to distinguish between the two using at least one of the distinguishing features in $S_{i,i'}$.

Theorem 3.3.1. *The inequalities* (3.12) *are valid, that is, they hold for all integer-feasible solutions of* (3.10).

Proof. Take any $(i, i') \in \Omega^+ \times \Omega^-$. If $\xi_i + \xi_{i'} \geq 1$ then (3.12) clearly holds. Otherwise, $i \in \Omega^+$ and $\xi_i = 0$ imply that $\sum_{t \in \Gamma} H_{it}\lambda_u \geq \rho$. Now, $\rho > 0$ implies that $h_u(A_i)\lambda_u > 0$ for some $t \in \Gamma$; $\lambda_u \geq 0$ and $h_u(A_i) = 1$ imply $\mu_u > 0$. The proof for $(i, i') \in \Omega^- \times \Omega^+$ is similar. □

In the following, we will denote by $\mathcal{A}$ some subset of pairs in $(\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+)$. Now, we let

$$\mathcal{R}(\mathcal{A}, \Gamma) = \left\{ (\xi, \mu, \lambda) \in [0,1]^M \times [0,1]^{|\Gamma|} \times \mathbb{R}_+^{|\Gamma|} \;\middle|\; \xi_i + \xi_{i'} + \textstyle\sum_{u \in S_{i,i'} \cap \Gamma} \mu_u \geq 1,\ \forall (i, i') \in \mathcal{A} \right\}$$

denote the polyhedron implied by the cutting planes (3.12) corresponding to the pairs of observations in $\mathcal{A}$.

As a direct consequence of Theorem 3.3.1,

$$Q_{H,y,\rho}(\Gamma) \cap \{0,1\}^M \times \{0,1\}^{|\Gamma|} \subseteq Q_{H,y,\rho}(\Gamma) \cap \mathcal{R}(\mathcal{A},\Gamma) \subseteq Q_{H,y,\rho}(\Gamma)$$

Similarly, it can be shown by letting $\rho = 1$ that the inequalities (3.12) are valid for (3.4), so that

$$Q_{H,y} \cap \{0,1\}^M \times \{0,1\}^U \subseteq Q_{H,y} \cap \mathcal{R}(\mathcal{A},\mathcal{U}) \subseteq Q_{H,y}.$$

Finally, we note that following work done on characterization of the set cover polytope [4, 22], or equivalently by applying a special case of Chvatal-Gomory cuts, the inequalities (3.12) can be further strengthened by introducing certain inequalities derived from triples of observation pairs. However, we have not pursued this line of research further.

3.4 L_0-Relaxed Boosting: a boosting formulation with relaxed L_0 complexity penalties

We now describe a boosting algorithm for the continuous relaxation of SMDH, strengthened by inequalities of the form (3.12). Our algorithmic approach is similar to Demiriz *et al.* [26]: We use column generation to iteratively generate the columns of $\mathcal{U}$ as needed. At each iteration, the set of features is restricted to the subset $\Gamma \subseteq \mathcal{U}$ of the columns that have been generated so far. At each iteration, the base learner algorithm generates a new feature from

$\mathcal{U} \setminus \Gamma$. Given a current set of features Γ and set of pairs $\mathcal{A}$, we arrive at the relaxation

$$\min_{\lambda,\mu,\xi} \quad \sum_{i=1}^{M} \xi_i + \sum_{u=1}^{U} c_u \mu_u \tag{3.13a}$$

$$\text{s.t.:} \quad \sum_{u\in\Gamma} y_i H_{iu} \lambda_u + (1+\rho)\xi_i \geq \rho \qquad i = 1, ..., M \tag{3.13b}$$

$$\sum_{u\in\Gamma} \lambda_u = 1 \tag{3.13c}$$

$$\xi_i + \xi_{i'} + \sum_{u\in S_{ii'}\cap\Gamma} \mu_u \geq 1 \qquad (i,i') \in \mathcal{A} \tag{3.13d}$$

$$\mu_u - \lambda_u \geq 0 \qquad u \in \Gamma \tag{3.13e}$$

$$\lambda_u, \mu_u, \xi_i \geq 0 \qquad u \in \Gamma,\ i = 1, ..., M \tag{3.13f}$$

3.4.1 Dual formulation, the base learning problem and termination

We next derive the dual formulation of (3.13) in order to formulate the pricing/base learning problem. The dual formulation is also useful in proving the algorithm's termination condition, which guarantees an optimal solution (or an approximately optimal solution). Let w_i, α, $v_{ii'}$, and q_u correspond respectively to the dual variables (or Lagrange multipliers) of i^{th} constraint (3.13b), (3.13c), constraint (3.13d) of pair (i,i'), and the u^{th} constraint (3.13e).

$$\max_{w,v,\alpha,q} \quad \sum_{i=1}^{M} \rho w_i + \sum_{(i,i')\in\mathcal{A}} v_{ii'} + \alpha \tag{3.14a}$$

$$\text{s.t.:}$$

$$\sum_{(k,l):i\in\{k,l\}} v_{kl} + (1+\rho) w_i \leq 1 \qquad i \in \{1, \ldots, M\} \tag{3.14b}$$

$$\sum_{\substack{(k,l)\in\mathcal{A}:\\ k=i\vee l=i}} v_{ii'} + q_u \leq c_u \qquad u \in \Gamma \tag{3.14c}$$

$$\alpha + \sum_{i=1}^{M} y_i \hat{H}_{it} w_i - q_u \leq 0 \qquad u \in \Gamma \tag{3.14d}$$

$$w \geq 0, q_u, v_{ii'} \geq 0 \qquad (i,i') \in \mathcal{A}, u \in \Gamma \tag{3.14e}$$

Substituting $q_u = c_u - \sum_{(i,i')\in\mathcal{A}:S_{ii'}\ni t} v_{ii'}$, which is the largest value of q_u that satisfies (3.14c), will preserve all feasible solutions, since choosing q_u as large as possible must satisfy the corresponding inequality (3.14d), for any given feasible (α, w), and q_u does not appear in the objective or any constraint other than (3.14c) or (3.14d). Thus, we can state the dual LP of (3.13) as

$$\max_{u,v,\alpha} \quad \sum_{i=1}^{M} \rho w_i + \sum_{(i,i')\in\mathcal{A}} v_{ii'} + \alpha \tag{3.15a}$$

$$\text{s.t.:} \quad \sum_{\substack{(k,l)\in\mathcal{A}:\\ k=i\vee l=i}} v_{kl} + (1+\rho)w_i \le 1 \qquad i \in \{1,\dots,M\} \tag{3.15b}$$

$$\alpha + \sum_{(i,i')\in\mathcal{A}:S_{i,i'}\ni u} v_{ii'} + \sum_{i=1}^{M} y_i H_{iu} w_i \le c_u \qquad u \in \Gamma \tag{3.15c}$$

$$w \ge 0, v_{ii'} \ge 0 \qquad (i,i') \in \mathcal{A}. \tag{3.15d}$$

At each iteration, boosting algorithms invoke a base learning algorithm to find the best hypothesis to add to the classifier. In Section 1.5 we described the base learning problem which arises as the pricing problem of LP-Boost and its relation to maximum agreement problems in the computational learning literature. The base learning problem that arises with our new formulation (3.13) involves finding $u \in \mathcal{U}$ that produces the most violated constraint of the form (3.15c), and is similar to (1.15), but involves an additional weight measure $v : (\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+) \to \mathbb{R}_+$ corresponding to the Lagrange multipliers of the constraints (3.13d); it takes the form

$$\bar{c} = \min_{u\in\mathcal{U}} \left\{ c_u - \sum_{i=1}^{M} y_i H_{iu} w_i - \sum_{(i,i')\in\mathcal{A}:u\in S_{ii'}} v_{ii'} - \alpha \right\}, \tag{3.16}$$

The minimizer of $\bar{c}$ can be determined irrespective of the constant α, which may be moved outside the min operation. In the general case, it is unsurprising that the problem (3.16) is $\mathcal{NP}$-hard. This fact is confirmed in Section 3.5 in the special case where the columns of H correspond to vertices of subcubes of the binary hypercube. Whenever $\bar{c} \ge 0$, the dual constraints (3.15c) must be satisfied for all $u \in \mathcal{U}$, and the dual LP (3.15) becomes feasible.

Thus, $\bar{c} \geq 0$ proves optimality of a selection of base classifiers (and columns) $\Gamma \subseteq \mathcal{U}$, and defines our termination condition. It is also possible to terminate early while guaranteeing an approximate solution of the relaxation [51], although in the experiments of Section 3.6 we did not find this necessary.

Let us denote the optimal solution value of the SMDH problem (3.10) by $z(\Gamma)$, and its relaxation (3.13) optimal value by $z_R(\Gamma)$ (for a particular fixed instance (H, y, ρ)). Having solved the relaxation (3.13) for a set of columns $\Gamma \subseteq \mathcal{U}$, it may be possible terminate the column generation procedure early while guaranteeing a lower bound on the solution value. The following lower bound is based on a general lower bound commonly used for early termination of column generation [51].

Suppose b is an upper bound on the number of features in the optimal solution, *i.e.*, $\sum_{u=1}^{U} \mu_u^* \leq b$ for the optimal solution $(\xi^*, \mu^*, \lambda^*)$ of (3.13), with value $z_R(\Gamma)$. By Lemma 3.2.3, a trivial choice is $b = M$, but it may be possible to find smaller values in terms of M and c, for example if $c_u = C \geq 1$ for all $u \in \mathcal{U}$, then $b = \lceil M/C \rceil$. If $\bar{c}$ is the optimal reduced cost computed by (3.16), then the optimal solution value can improve by at most $|\bar{c}b|$, so that the following bounds are guaranteed for the solution of $z_R(\mathcal{U})$:

$$z_R(\Gamma) + \bar{c}b \leq z_R(\mathcal{U}) \leq z_R(\Gamma). \tag{3.17}$$

Further, for the integer optimal solution of the master problem $z(\mathcal{U})$, we can derive a tighter lower bound when c_u are integer for all $u \in \mathcal{U}$, namely

$$\lceil z_R(\Gamma) + \bar{c}b \rceil \leq \lceil z_R(\mathcal{U}) \rceil \leq z(\mathcal{U}) \leq \hat{z}, \tag{3.18}$$

where $\hat{z}$ is any upper bound on $z(\mathcal{U})$. The upper bound $\hat{z}$ can be computed by simply rounding up any non-integer variables in the solution (ξ, μ, λ) of (3.13), or by any other rounding that maintains feasibility for (3.13). Note that we can rewrite (3.10), whenever the coefficients c_u are rational, as an equivalent optimization problem with integer coefficients c_u.

Finally, we note that in order to compute a risk upper bound such as (1.11), we can use

any feasible rounding of an intermediate solution (ξ, μ, λ), with value $\hat{u}$, as an upper bound for $z(\mathcal{U})$ and thus for the code length $\min_{g \in \mathcal{G}} \bar{L}[y, g]$.

Algorithm 2 L_0-RBoost

Input: $M \times N$ matrix A and labels $y \in \{-1, 1\}^M$
Output: (ξ, μ, λ)
Let $\Gamma \leftarrow \{1, 2\}$, where $h_1(A_i) = 1$ and $h_2(A_i) = -1$ for all $i \in \{1, \ldots, M\}$, $\mathcal{A} \leftarrow \emptyset$
repeat
  Solve (3.13) and obtain the solution (ξ, μ, λ) and Lagrange multipliers (w, v, α)
  Solve the base learning problem:

$$u^* = \text{argmin}_{u \in \mathcal{U}}\, c_u - \sum_{(i,i') \in \mathcal{A}: S_{ii'} \ni u} v_{ii'} - \sum_{i=1}^{M} y_i H_{iu} w_i - \alpha$$

$$\bar{c} = c_{u^*} - \sum_{(i,i') \in \mathcal{A}: S_{ii'} \ni u^*} v_{ii'} - \sum_{i=1}^{M} y_i H_{iu^*} w_i - \alpha$$

  $\Gamma \leftarrow \Gamma \cup \{u^*\}$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(i, i') \in (\Omega + \times \Omega^-) \cup (\Omega^- \times \Omega^+) \mid h_{u^*}(A_i) \neq h_{u^*}(A_{i'})\}$
  Let $V = \left\{(i, i') \;\middle|\; \xi_i + \xi_{i'} + \sum_{u \in S_{i,i'} \cap \Gamma} \mu_u < 1\right\}$
  if $\bar{c} \geq 0$ and $V \neq \emptyset$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \cup V$
  end if
until $\bar{c} \geq 0$ and $V = \emptyset$

3.4.2 The boosting algorithm

The L_0-RBoost algorithm is shown as Algorithm 2. We generate columns for the primal formulation (3.13), rather than (equivalently) generating cuts for a dual formulation as suggested by Demiriz *et al.* for LP-Boost. An attractive property of the dual formulation is the ease of finding an initial feasible solution by assigning the observations equal weights $w_i = 1/M$. In our case, we are interested in the integrality of the solution vectors ξ and μ, so we prefer to work in the primal space. We initialize Γ to contain two columns that are both "simple" and easy to compute, corresponding to the constant base classifiers: we take $\Gamma = \{1, 2\}$, where $h_1(A_i) = 1$ and $h_2(A_i) = -1$ for all $i \in \{1, ..., M\}$. Next, we iterate by solving the relaxation (3.13), and then solving a base learning problem (3.16) to find the best pair of variables λ_u and μ_u to be added. Note that it may be beneficial in practice to add more than a single column u in each iteration, though we did not experiment with such

strategies. We terminate when the dual becomes feasible, *i.e.*, when (3.15c) is satisfied for all $u \in \mathcal{U}$, implying an optimal solution.

If the number of observations is not too large, we may simply solve (3.13) using all possible cuts, that is, start with $\mathcal{A} = (\Omega^+ \times \Omega^-) \cup (\Omega^- \times \Omega^+)$. The number of possible cutting planes (3.12) is $2|\Omega^+||\Omega^-|$, which is polynomial in M, but may be prohibitively large for some datasets. In order to handle large input datasets, we dynamically generate the cutting planes; for each newly added column u, we add the cutting planes (3.12) that correspond to pairs of positive and negative observations that are distinguished by u, *i.e.*, pairs in:

$$\left\{(i, i') \in (\Omega + \times\Omega^-) \cup (\Omega^- \times \Omega^+) \;\mid\; h_u(A_i) \neq h_u(A_{i'})\right\} \setminus \mathcal{A} \tag{3.19}$$

These cutting planes are designed to push up the value of the newly generated variable μ_u as close as possible to 1; they may otherwise be as small as λ_u. We also note that in order to speed up the cut-adding step 8, we may add some subset of (3.19) instead of the entire set. In particular, we find it useful as a heuristic, to choose the subset of cuts based on a similarity measure of the pair of observations A_i and $A_{i'}$ associated with each cut. We will further elaborate on this idea in Section 3.6. Finally, before terminating, we make sure to add all remaining violated cuts in step 11 of Algorithm 2. This step prevents premature termination, and is especially effective in increasing the value of the variables ξ_i for any remaining misclassified observations. Initially, the number of misclassified observations is large, but it drops as the algorithm progresses; delaying step 11 thus allows fewer cuts to be generated.

3.4.3 Analysis and margin maximization with L_0 relaxation penalties

Algorithm 2 solves a tighter relaxation of the (generalized) SMDH problem than the straightforward relaxation that minimizes the L_1-norm of λ (*i.e.*, as used by LP-Boost). Alternatively, using any polynomial time algorithm to solve the LP in step 5, since the number of cuts (3.13d) is at most $|\Omega^+||\Omega^-|$, the running time of Algorithm 2 can be shown to be polynomial in the dimensions M and U, and the size of the encoding of the coefficients ρ

and c [50].

A disadvantage of our formulation may be that we may not know how to set the required margin ρ. An alternative may be to try to maximize the margin within the optimization problem. Let $z(\rho)$ denote the optimal solution value of (3.13) with parameter ρ. The following formulation maximizes ρ subject to a (relaxed) code length penalty proportional to $z(\rho)$:

$$\max_{\xi,\mu,\lambda,\rho} \left\{ \rho - \beta \left(\sum_{i=1}^{M} \xi_i + D \sum_{u=1}^{U} \mu_u \right) \mid (\xi, \mu, \lambda) \in Q_{H,y,\rho}(\Gamma) \cap \mathcal{R}(\mathcal{A}, \Gamma) \right\} \tag{3.20}$$

Making ρ a variable on both the left and right-hand side of the constraint $y_i H_i \lambda + (1+\rho)\xi_i \geq \rho$ results in a quadratic optimization problem. Alternatively, one may fix the coefficient $1+\rho$ on the left-hand side to be a large upper bounding constant such as $2 \geq 1+\rho$. However, this results in a poorer relaxation of the integer solution. We proceed to show that (3.20) can be solved in polynomial time in the input (when the input includes $\mathcal{U}$).

Lemma 3.4.1. *Suppose* (ξ, μ, λ) *is a feasible solution of* (3.13) *with* $\rho = \rho' > 0$. *Then there exists* $\lambda^* \geq 0$ *such that* (ξ, μ, λ^*) *is feasible for* (3.13), *with* $\rho = \rho'' \geq M^{-(M/2+1)}$.

Proof. Let $\Gamma = \{u \in \mathcal{U} \mid \lambda_u > 0\}$. The feasibility of (ξ, μ, λ) for (3.10) implies the linear separability of $S^+ = \{i \in \Omega^+ \mid \xi_i = 0\}$ and $S^- = \{i \in \Omega^- \mid \xi_i = 0\}$. By Lemmas 3.2.2 and 3.2.3, there exists a $\lambda' \geq 0$ that separates S^+ and S^-, satisfies $\lambda'_u \leq M^{M/2}$ for all $u \in \mathcal{U}$, and has $||\lambda'||_0 \leq M$. We then have

$$\sum_{u \in \Gamma} y_i H_{iu} \lambda'_u + \left(\sum_{u=1}^{U} \lambda'_u + 1 \right) \xi_i \geq 1,$$

for all $i \in \{1, \ldots, M\}$.

Dividing by $\sum_{u \in \mathcal{U}} \lambda'_u$ and substituting the variable $\lambda^*_u = \frac{\lambda'_u}{\sum_{u \in \mathcal{U}} \lambda'_u}$, we have $\sum_{u \in \mathcal{U}} \lambda^*_u = 1$ and

$$\sum_{u \in \Gamma} y_i H_{iu} \lambda^*_u + \frac{\sum_{u \in \mathcal{U}} \lambda'_u + 1}{\sum_{u \in \mathcal{U}} \lambda'_u} \xi_i \geq 1 / \sum_{u \in \mathcal{U}} \lambda'_u.$$

Now, letting $\rho'' = 1/\sum_{u\in\mathcal{U}} \lambda'_u$,

$$\sum_{u\in\Gamma} y_i H_{iu}\lambda^*_u + (1+\rho'')\xi_i \geq \rho''.$$

where $\rho'' = 1/\sum_{u=1}^{U} \lambda'_u \geq M^{-(M/2+1)}$, for all $i \in \{1,\ldots,M\}$. Thus, (ξ,μ,λ^*) is feasible for (3.13), with $\rho = \rho''$. □

Lemma 3.4.2. *The optimal solution value of* (3.13)*, $z(\rho)$, is a convex function of ρ.*

Proof. Consider the solutions (ξ',μ',λ') with $\rho = \rho_1$ and (ξ'',μ'',λ'') with $\rho = \rho_2$ to (3.13). Then, by adding α times the i^{th} constraint (3.13b) of Q_{H,y,ρ_1} and $(1-\alpha)$ times constraint (3.13b) of Q_{H,y,ρ_1} for $\alpha \in [0,1]$,

$$\begin{aligned}\alpha y_i\hat{H}_i\lambda' + \alpha(1+\rho_1)\xi'_i + (1-\alpha)y_i\hat{H}_i\lambda'' + (1-\alpha)(1+\rho_2)\xi''_i \\ = y_i\hat{H}_i(\alpha\lambda' + (1-\alpha)\lambda'') + (1+\alpha\rho_1+(1-\alpha)\rho_2)\,\xi_i \geq \alpha\rho_1 + (1-\alpha)\rho_2,\end{aligned}$$

and thus constraint (3.13b) is feasible for

$$(\bar{\xi},\bar{\mu},\bar{\lambda}) = (\alpha\xi' + (1-\alpha)\xi'', \alpha\mu' + (1-\alpha)\mu'', \alpha\lambda' + (1-\alpha)\lambda'')$$

in (3.13) with $\rho = \alpha\rho_1 + (1-\alpha)\rho_2$. Clearly, the constraints (3.13c), (3.13d) and (3.13e) are also feasible for $(\bar{\xi},\bar{\mu},\bar{\lambda})$. We then have:

$$\begin{aligned} z\left(\alpha\rho_1 + (1-\alpha)\rho_2\right) &\leq \sum_{i=1}^{M}\bar{\xi}_i + \sum_{u=1}^{U} c_u\bar{\mu}_u \\ &= \sum_{i=1}^{M}\left(\alpha\xi'_i + (1-\alpha)\xi''_i\right) + \sum_{u=1}^{U} c_u\left(\alpha\mu'_u + (1-\alpha)\mu''_u\right) \\ &= \alpha z(\rho_1) + (1-\alpha)z(\rho_2) \end{aligned}$$

□

Theorem 3.4.3. *Maximizing the L_1 margin subject to a relaxed L_0 penalty, as specified by* (3.20)*, can be approximated to within a factor of $1-\epsilon$, for any $\epsilon > 0$, in time polynomial in the input size M, U, the encoding size of the coefficients c and ρ, and $1/\epsilon$.*

Proof. By Lemma 3.4.2 the optimal solution of (3.13), $z(\rho)$, is a convex function of ρ. Thus, $\rho - \beta z(\rho)$ is a concave function of ρ, for any $\beta \geq 0$. By Lemma 3.4.1, it suffices to consider ρ within the interval $[1/M^{M/2+1}, 1]$. We consider approximately maximizing $\rho - \beta z(\rho)$ by binary search.

The number of evaluations needed in order to approximate the optimal margin ρ^* within $1-\epsilon$, for any given $\epsilon > 0$, is $\log\left(\frac{1-M^{-M/2-1}}{\epsilon}\right)$. Each such evaluation can be done by solving a linear program, which is solvable in time polynomial in M, T, and the encoding size of c and ρ. □

3.5 Application using Boolean monomial base classifiers

Now, assuming that our data A is binary, we consider a specific class of base classifiers corresponding to Boolean monomials. Note that any given data $A \in \mathbb{R}^{M \times N'}$ can be binarized using a number of binary attributes that is at most polynomially larger in M and N' than N' [15, 40]. To each monomial (J, C), we associate two features: a positive feature u^+ and a negative feature u^-; the corresponding base classifiers are $h_{u^+(J,C)}(x) = m_{J,C}(x)$ and $h_{u^-(J,C)} = -m_{J,C}(x)$. Let the positive reward associated with a monomial (J, C):

$$f^+(J,C) = w(\Omega^+ \cap \mathrm{Cover}(J,C)) - w(\Omega^- \cap \mathrm{Cover}(J,C)) + \sum_{(i,i') \in ((\Omega^+ \cap \mathrm{Cover}(J,C)) \times (\Omega^- \setminus \mathrm{Cover}(J,C)))} v_{ii'},$$

and the negative reward associated with (J, C):

$$f^-(J,C) = w(\Omega^- \cap \mathrm{Cover}(J,C)) - w(\Omega^+ \cap \mathrm{Cover}(J,C)) + \sum_{(i,i') \in ((\Omega^- \cap \mathrm{Cover}(J,C)) \times (\Omega^+ \setminus \mathrm{Cover}(J,C)))} v_{ii'},$$

where $w(S) = \sum_{i \in S} w_i$.The base learning problem is then to find (J, C) so that

$$f(J,C) = \max\left\{f^+(J,C), f^-(J,C)\right\} - c(|J| + |C|) \tag{3.21}$$

is maximized, where $c(k)$ denotes the complexity penalty of a monomial of order k. Here, we focus on base classifiers that correspond to Boolean monomials. In Chapter 4 we elaborate on the formulation of agreement problems with abstaining classifiers, and extend the algorithms for solving the new base learning (pricing) problem (3.16) in the case of Boolean monomials. The problem of finding a monomial of arbitrary order that maximizes $f(J, C)$ is a generalization of the maximum monomial agreement problem [48, 27], and thus it trivially follows by the restriction $v_{ii'} = 0$ for all (i, i') that the problem is $\mathcal{NP}$-hard. However, in many learning applications, it is reasonable to bound the order of the monomials by a small constant. Clearly, if the order of monomials is bounded by a constant K, then the monomial that maximally agrees with the data can be found in polynomial time by simple enumeration.

The MDL/compression approach discussed in Section 3.1 suggests one way of determining $c(k)$: the code book contains K tables, and the table for monomials of order k contains $|\mathcal{U}_k| = 2^k\binom{N}{k}$ entries. Based on the code length upper bound (3.2), normalizing by the approximate $\log M$ cost of a misclassified observation, we may define the cost of a Boolean monomial of order $k = |J| + |C|$ as:

$$c(k) = \frac{\log\left(2^k\binom{N}{k}\right) + \log K}{\log M} + \kappa = \frac{k + \log\binom{N}{k} + \log K}{\log M} + \kappa, \tag{3.22}$$

where κ is a constant that accounts for the bits that encode the weights λ. We can approximate the additional cost associated with the term $(||\lambda||_0 + 1)/\,||\lambda||_0$ in (3.2), which is associated with the cost of encoding λ, by choosing $\kappa \in [1 + 1/M, 2]$.

3.6 Experimental work and discussion

We compare the classification performance of L_0-RBoost (Algorithm 2) with LP-Boost with a Boolean monomial base classifier. The formulation used in LP-Boost is the ν-LP formulation (1.12), where $D = \frac{1}{\nu M}$ [26, 59]. The parameter ν also corresponds to an upper bound on the fraction of margin errors, *i.e.*, $|\{i \mid \xi_i > 0\}|/M$, as shown in Theorem 1.4.2. In our experiments, we consider base learning problems comprising either of all Boolean monomials of order $k = 1$, or all monomials up to order $k = 5$. Although the order of monomials

that we consider is bounded by constant, so that the maximum monomial agreement can be solved in polynomial time, it may be too computationally intensive to enumerate all monomials up to $k = 5$. Therefore, in order to find the monomial that best agrees with the data, we adapted a branch-and-bound algorithm for maximum monomial agreement [28, 40]. We further elaborate on the extension of the branch-and-bound algorithm for the new base learning (or pricing) problem in Section 4.2.

In our implementation of Algorithm 2, we have found it more effective to add only a subset of the cuts in step 8 of the algorithm. Experimenting with a binary matrix A we have found that a small Hamming distance between the vectors A_i and $A_{i'}$ to be a good indication of the potential of the corresponding cut to tighten the relaxation. Specifically, for fast processing of the cuts, after adding a base classifier u^*, we scan in quadratic time the pairs $(i, i') \in \Omega^+ \times \Omega^-$ for which $h_{u^*}(A_i) \neq h_{u^*}(A_{i'})$ and add only those cuts corresponding to pairs whose Hamming distance is less than a fixed factor of the minimum Hamming distance found so far.

That we are better able to minimize $||\lambda||_0$ than the solution of (1.12), for a given margin of separation, is suggested by Theorem 3.3.1. The theorem implies that adding the valid inequalities (3.12) may strengthen the "soft margin" formulations (1.13) and (1.12) as continuous relaxations of (3.4). The LP formulation (1.12) is known to find sparse classifiers, but is also sensitive to the choice of the tunable penalty parameter D. By assigning a small enough penalty D (large ν) in (1.12), it is possible to find a "degenerate" classifier with large ρ by assigning many observations to be outliers.

In our experiments with L_0-RBoost, we fixed the parameters c_u of formulation (3.13) to equal the code length penalty of Boolean monomial base classifiers (3.22). The approximating constant κ in (3.22) was set to 1.5. We then investigated the dependence of L_0-RBoost and LP-Boost on the tunable parameters ρ and ν, with respect to classification performance and sparsity.

Figures 3.1–3.5 show the algorithm's performance on the test and training data, and the dependence of $||\lambda||_0$ on the tunable parameters of the two algorithms. Each point of the plots corresponds to 10 replications of 10-fold experiment. k-fold experiments involve

a partitioning of the dataset into 10 parts, each of which is used as a test set in an experiment, while the remaining 9/10 of the data is used to train a classifier. The classification performance results are then averaged over the total 100 experiments. We can see that for both algorithms, the models selected tend to overfit the training data for small value of the parameters. The overfitting is most apparent with LP-Boost and $K = 5$. The experiments show that, over the entire range of parameter values, L_0-RBoost generalizes well compared with LP-Boost. We also find that L_0-RBoost is robust with respect to a wide range of choices of the parameter ρ. Specifically, even with very small values of ρ, we obtain classification models that generalize well. In LP-Boost, the performance of the algorithm is highly sensitive to the choice of the parameter ν. When the input data is not linearly separable by the set of base classifiers $\mathcal{U}$, for example in the case when $\mathcal{U}$ is the set of monomial classifiers with $K = 1$, a constant base classifier with zero margin (classifying all to be in a single class) becomes optimal. Further, the value of ν that may be considered too small varies for the different datasets; in Figure 3.5, the performance of LP-Boost with $K = 1$ is poor or mediocre for $\nu \leq 0.5$, while in Figure 3.3 we can see that the classification performance of LP-Boost on the CLHEART dataset, for $K = 1$, peaks at $\nu \approx 0.4$.

Although we expect classification performance to improve with either cross-validation or optimization of the margin subject to a code-length penalty (which could be done by solving formulation (3.20)), we will limit our experiments to using a fixed value ρ. In Table 3.1, we show the results of our experiment with a fixed value of $\rho = 20/M$ for five binarized UCI datasets [3]. We compare the performance of L_0-RBoost with monomial base classifiers, up to order $K = 1$ and $K = 5$ to LP-Boost using the same base classifiers and to SLIPPER. We first ran these algorithms within the same 10-fold experiments of L_0-RBoost (*i.e.*, using the same training and test sets). The table also shows a practical comparison of our results with the previously published results of LP-Boost with C4.5 and stump decision trees [26], and the previously published performance of the SLIPPER algorithm [20]. For LP-Boost, Demiriz *et al.* fine-tuned the parameter ν for the different datasets, so that we did not expect to match all of their classification results here. The SLIPPER algorithm uses AdABoost with a heuristic (greedy) monomial base learner. In order to prevent overfitting, the SLIPPER algorithm uses cross-validation to simplify, or prune some of the monomials

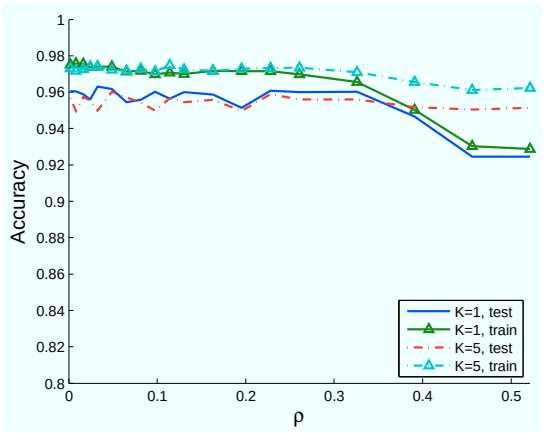
that are initially generated by the greedy algorithm. We ran the SLIPPER algorithm using the publicly available version with all parameters set to their default values. The apparent discrepancy in the results of the SLIPPER algorithm may be due to the binarization of the datasets in our experiments.

In Figures 3.6–3.10 the accuracy performance on the test set is plotted versus $||\lambda||_0$. The plots summarize the classification performance of the experiments depicted in Figures 3.1–3.5. As before, each point corresponds to an average of ten replications of a 10-fold experiment. L_0-RBoost seems to keep $||\lambda||_0$ within a smaller interval, which should be expected given that the cost coefficients c_u are fixed in all of the experiments. However, it also becomes apparent that the classifiers computed by L_0-RBoost, with only a few exceptions, are more accurate than those of LP-Boost for every value of $||\lambda||_0$.

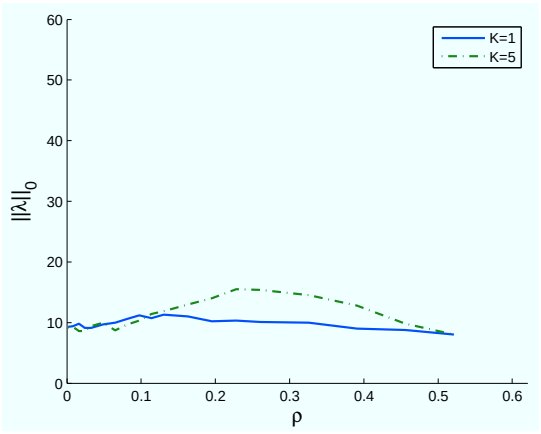
In the runs of L_0-RBoost, for which the results are shown in Table 3.1, we set the penalty constant $\kappa = 1.5$ in (3.22). The LP-Boost results are computed with a parameter $\nu = 0.50$ for $K = 5$, and $\nu = 0.56$ for $K = 1$. The parameters for LP-Boost were chosen so that the resulting classifiers were quite sparse, while achieving good classification performance on the Sonar and CLHEART datasets. The computational runs of Table 3.1 consist of 20 replications of a 10-fold experiment. We can see from the results of Table 3.1 that L_0-RBoost finds classifiers that are approximately as sparse as the classifiers found by LP-Boost, but usually achieve superior classification performance. Both algorithms seem to outperform SLIPPER. Some of the shortfalls of L_0-RBoost in fact occur with the less restricted class of monomials when $K = 5$, suggesting that overfitting may be occurring in some cases. The concern of overfitting may motivate us to evaluate other types of penalty functions in place of (3.22) and different values of the penalty constant in future work. Particularly, complexity measures that can be computed based on the input data, such as Rademacher complexity [7] could be the subject of further investigation.

Method	Datasets									
	BCW		VOTE		CLVHEART		HUHEART		SONAR	
	Acc	$\|\|\lambda\|\|_0$	Acc	$\|\|\lambda\|\|_0$	Acc	$\|\|\lambda\|\|_0$	Acc	$\|\|\lambda\|\|_0$	Acc	$\|\|\lambda\|\|_0$
L_0RBoost $K = 1$	0.963	9.1	0.950	6.0	0.846	10.3	0.812	10.4	0.712	7.2
L_0RBoost $K = 5$	0.950	9.4	0.960	3.0	0.833	38.9	0.807	27.4	0.725	21.3
LPBoost $K = 1$	0.925	3.8	0.957	2	0.774	7.6	0.803	3.8	0.735	8.6
LPBoost $K = 5$	0.937	5.5	0.957	2.1	0.810	25.3	0.797	11.2	0.734	30.8
SLIPPER	0.959	19.5	0.952	3.9	0.802	14	0.802	14.7	0.674	18.8
SLIPPER [20]	0.958	NA	NA	NA	0.752	NA	0.806	NA	0.745	NA
LPBoost stumps [26]	0.966	NA	NA	NA	0.795	70.8	NA	NA	0.870	85.7
LPBoost C4.5 [26]	0.959	NA	0.959	NA	0.791	NA	NA	NA	0.817	NA

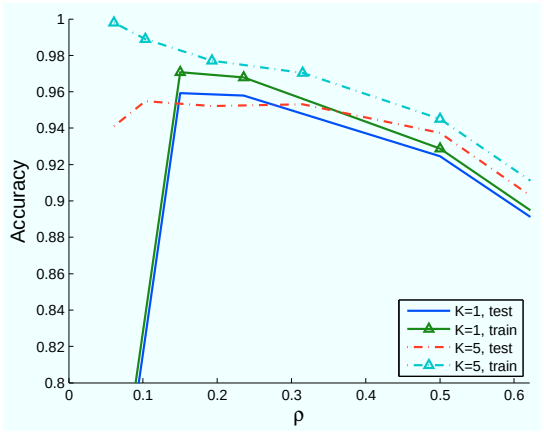
Table 3.1: Average accuracy and $||\lambda||_0$ for 20 replications of 10 folds each. The bottom three rows are as reported for SLIPPER [20] and LP-Boost [26]. A value of NA indicates that the data is unavailable from the corresponding publications.



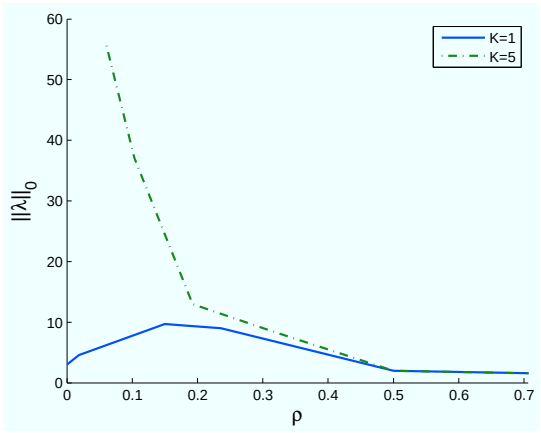
(a) L_0-RBoost accuracy vs. margin



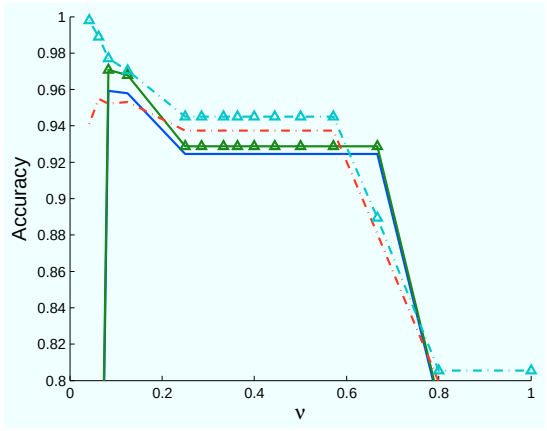
(b) L_0-RBoost sparsity vs. margin



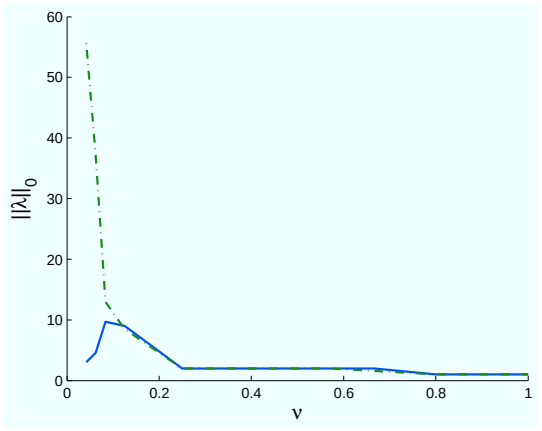
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

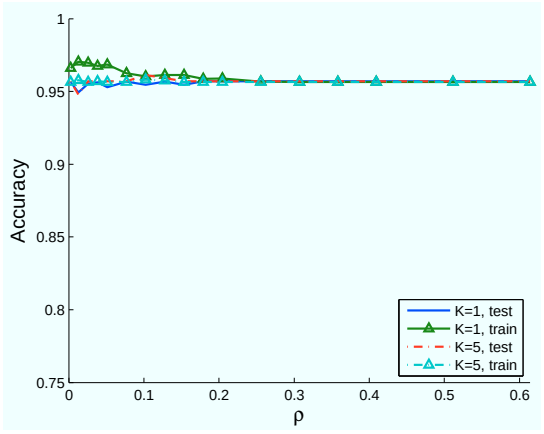


(e) LP-Boost accuracy vs. ν

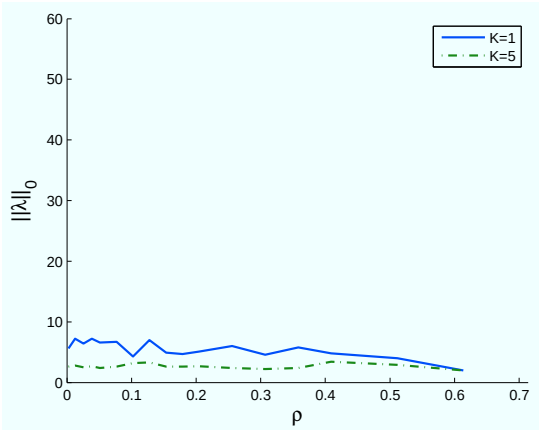


(f) LP-Boost sparsity vs. ν

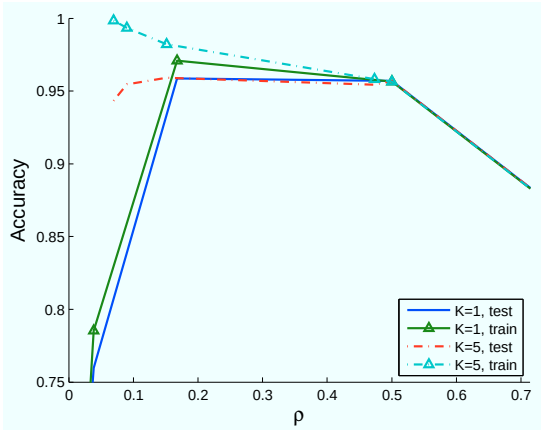
Figure 3.1: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the BCW dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



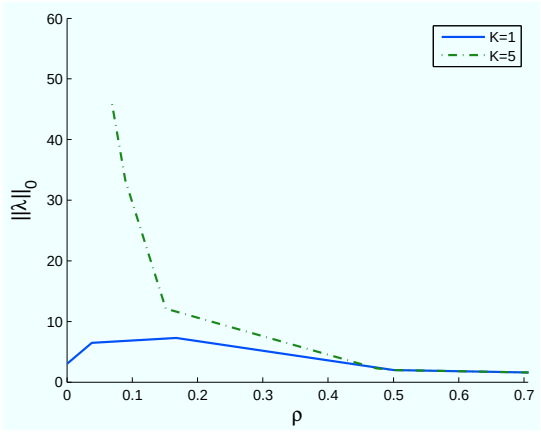
(a) L_0-RBoost accuracy vs. margin



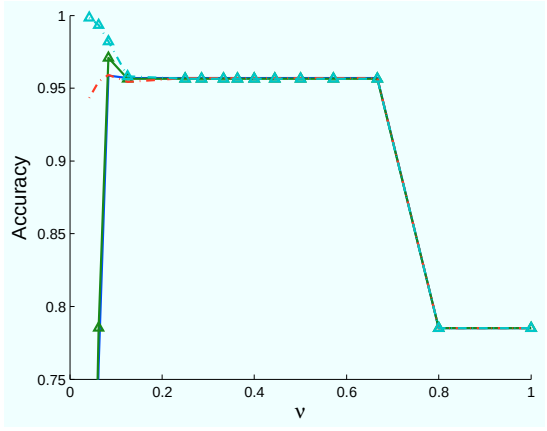
(b) L_0-RBoost sparsity vs. margin



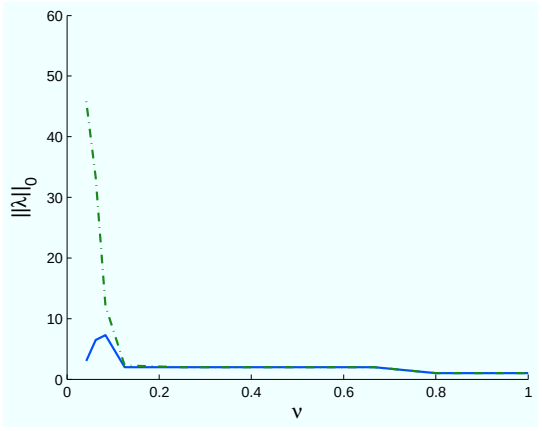
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

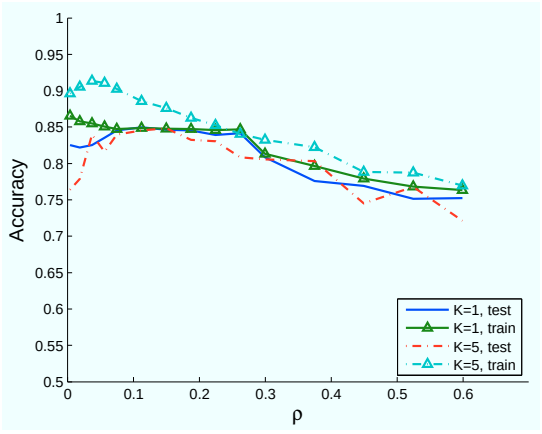


(e) LP-Boost accuracy vs. ν

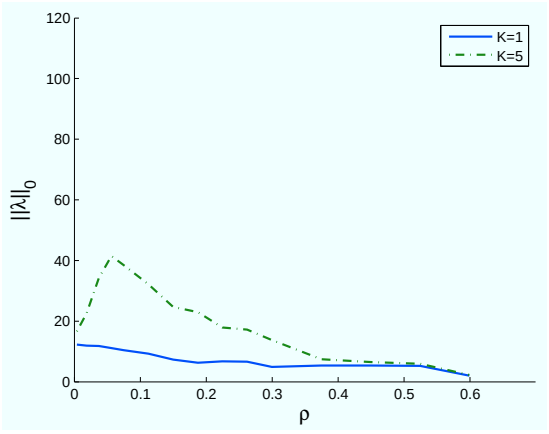


(f) LP-Boost sparsity vs. ν

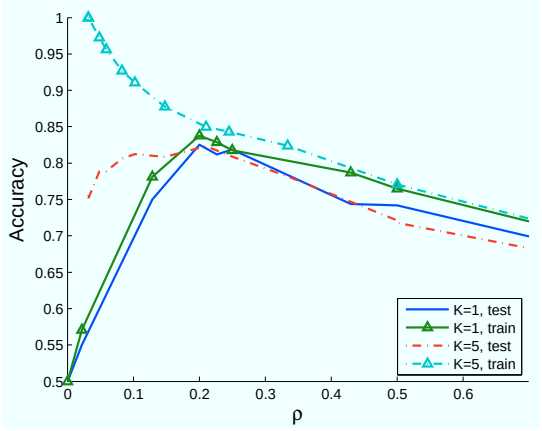
Figure 3.2: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the VOTE dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



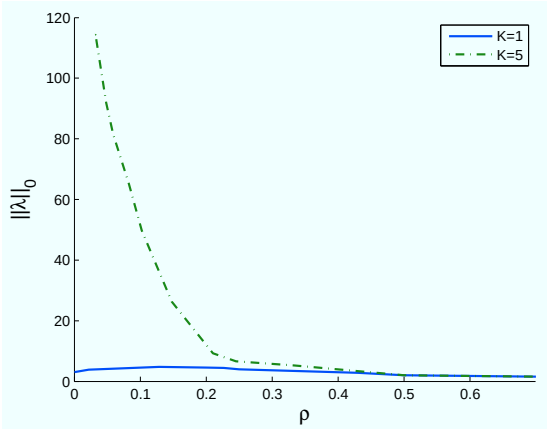
(a) L_0-RBoost accuracy vs. margin



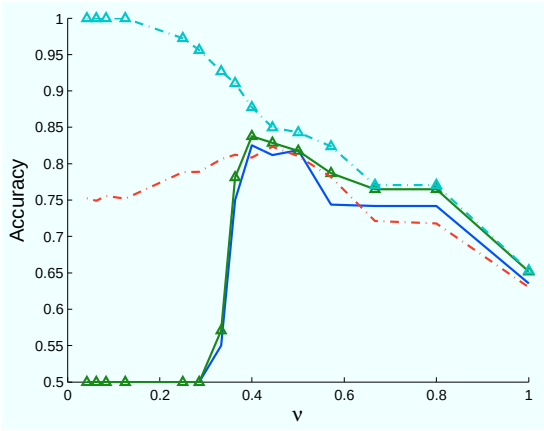
(b) L_0-RBoost sparsity vs. margin



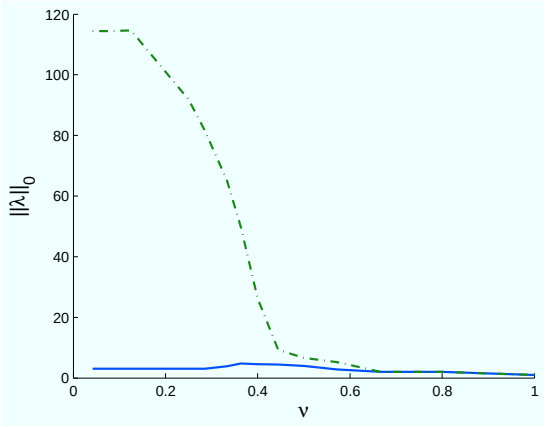
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

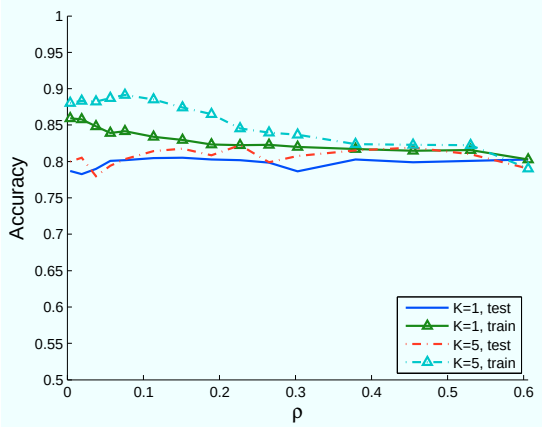


(e) LP-Boost accuracy vs. ν

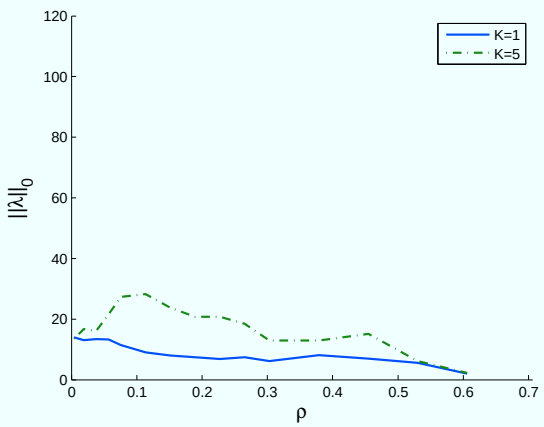


(f) LP-Boost sparsity vs. ν

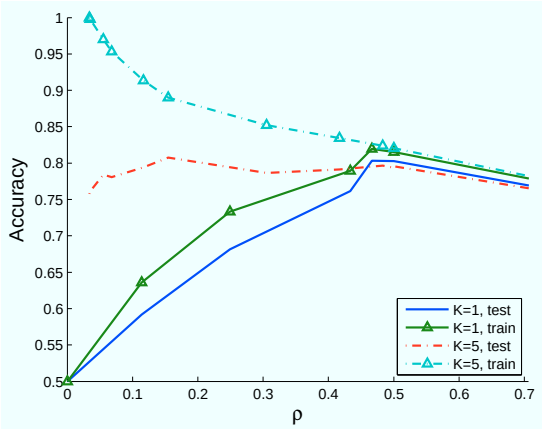
Figure 3.3: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the CLVHEART dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



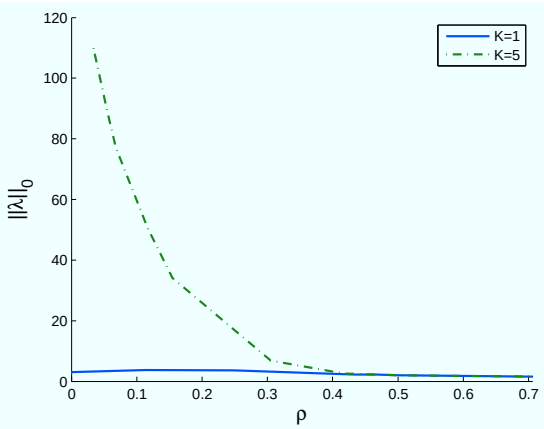
(a) L_0-RBoost accuracy vs. margin



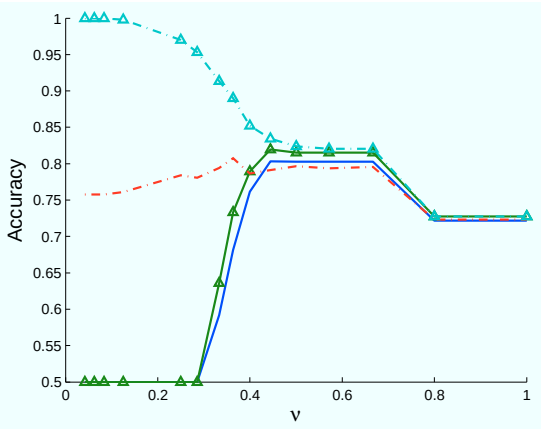
(b) L_0-RBoost sparsity vs. margin



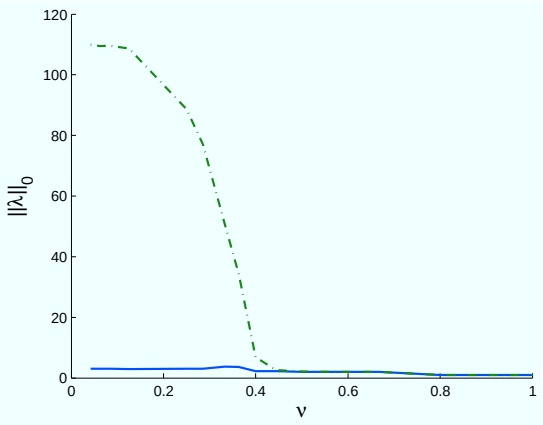
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

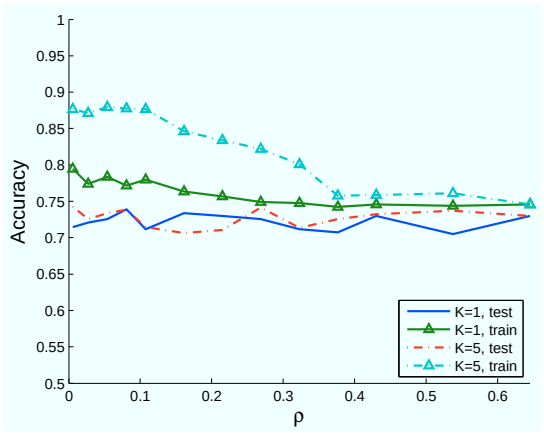


(e) LP-Boost accuracy vs. ν

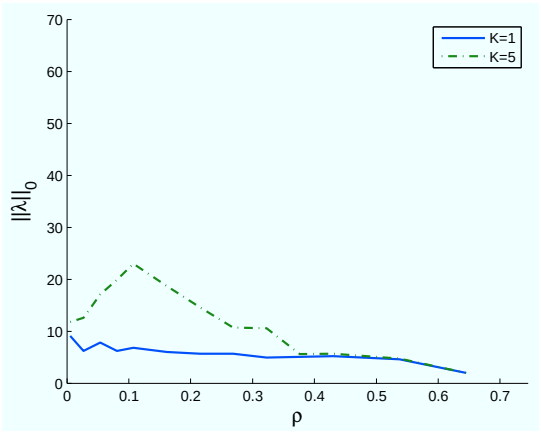


(f) LP-Boost sparsity vs. ν

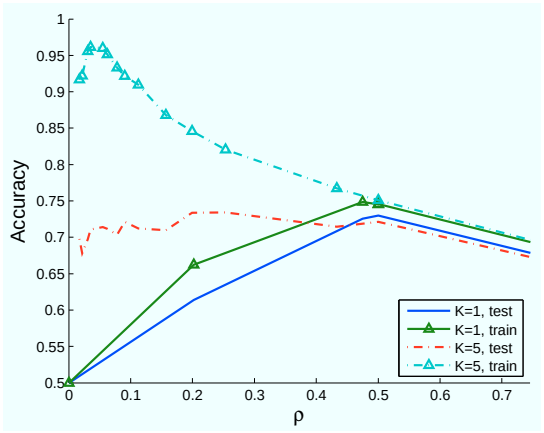
Figure 3.4: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers for the HUHEART dataset. Each point of the plot is computed by a averaging the accuracies of a 10-replication, 10-fold experiment.



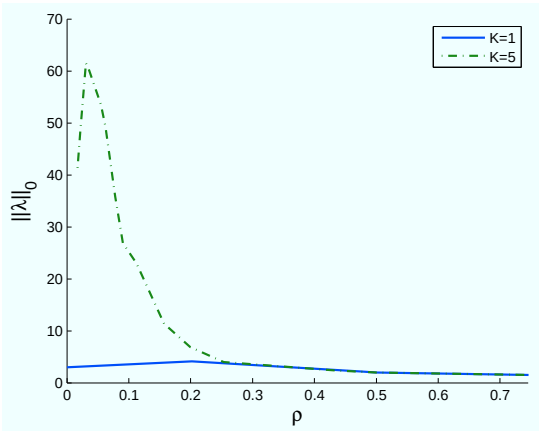
(a) L_0-RBoost accuracy vs. margin



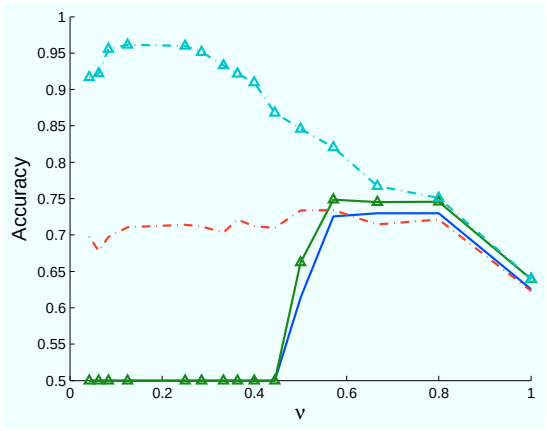
(b) L_0-RBoost sparsity vs. margin



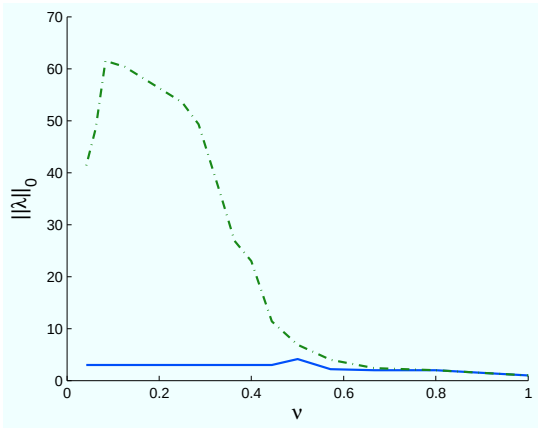
(c) LP-Boost accuracy vs. margin



(d) LP-Boost sparsity vs. margin

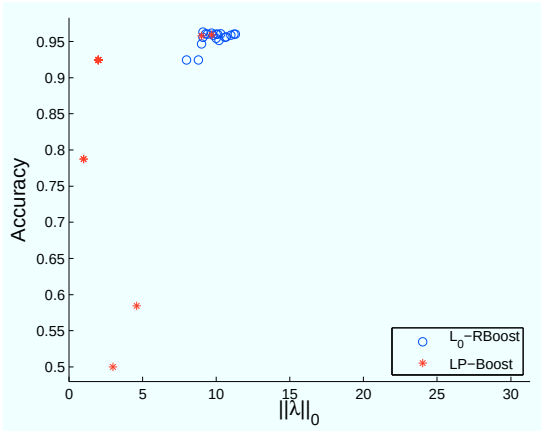


(e) LP-Boost accuracy vs. ν

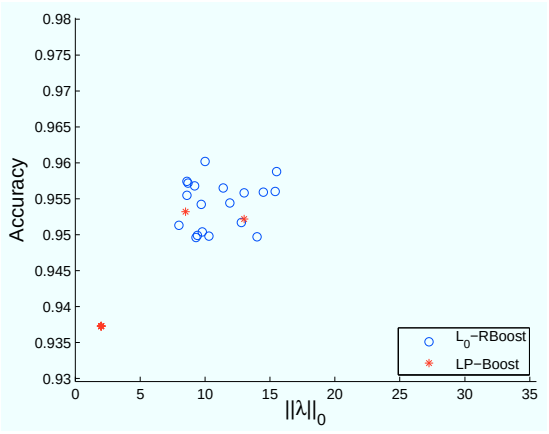


(f) LP-Boost sparsity vs ν

Figure 3.5: The classification performance and sparsity of LP-Boost versus L_0-RBoost with monomial base classifiers up to order $K = 1$ and $K = 5$ for the SONAR dataset. Each point of the plot is computed by averaging the accuracies of a 10-replication, 10-fold experiment.

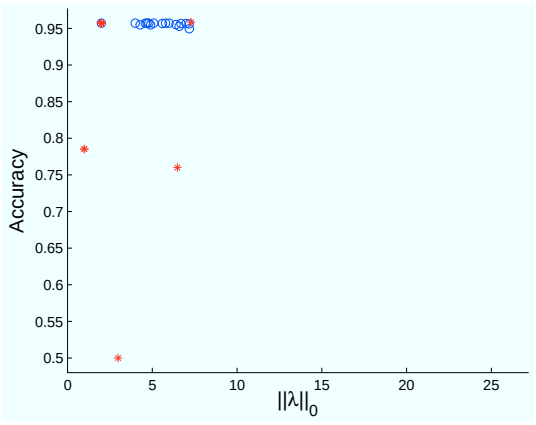


(a) $K = 1$

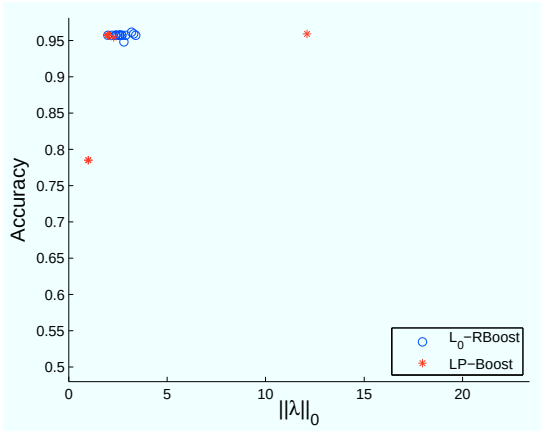


(b) $K = 5$

Figure 3.6: Test accuracy vs. $||\lambda||_0$ on the BCW dataset

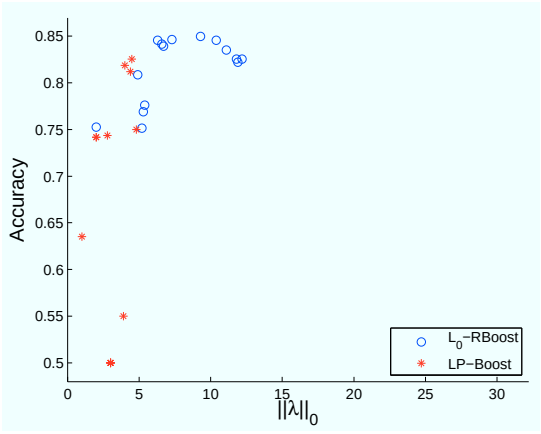


(a) $K = 1$

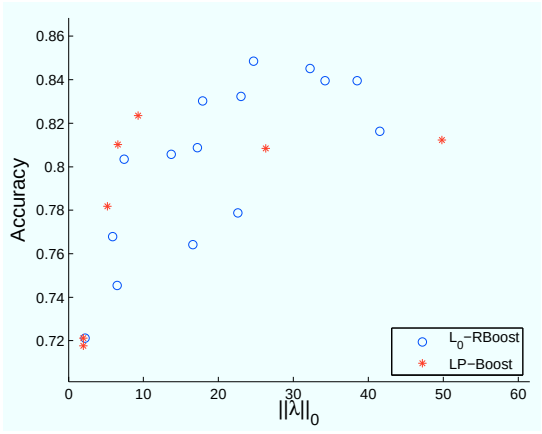


(b) $K = 5$

Figure 3.7: Test accuracy vs. $||\lambda||_0$ on the VOTE dataset

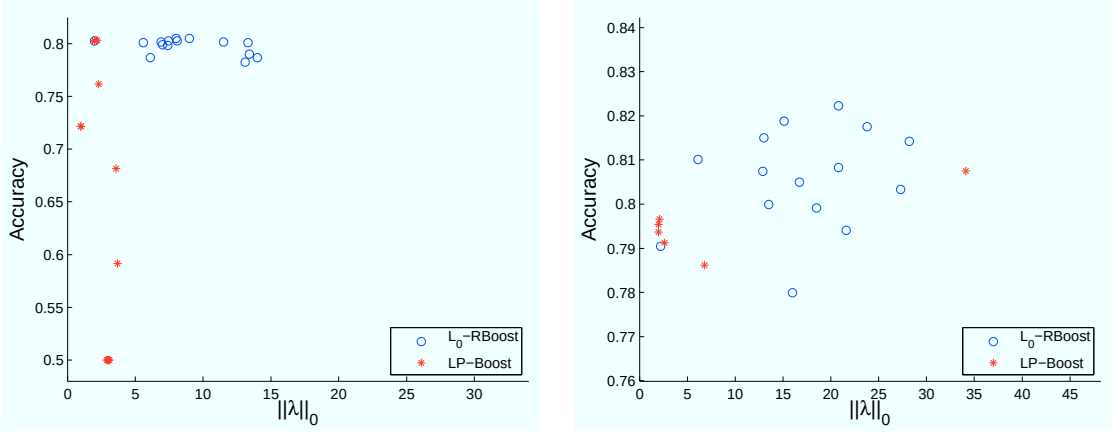


(a) $K = 1$



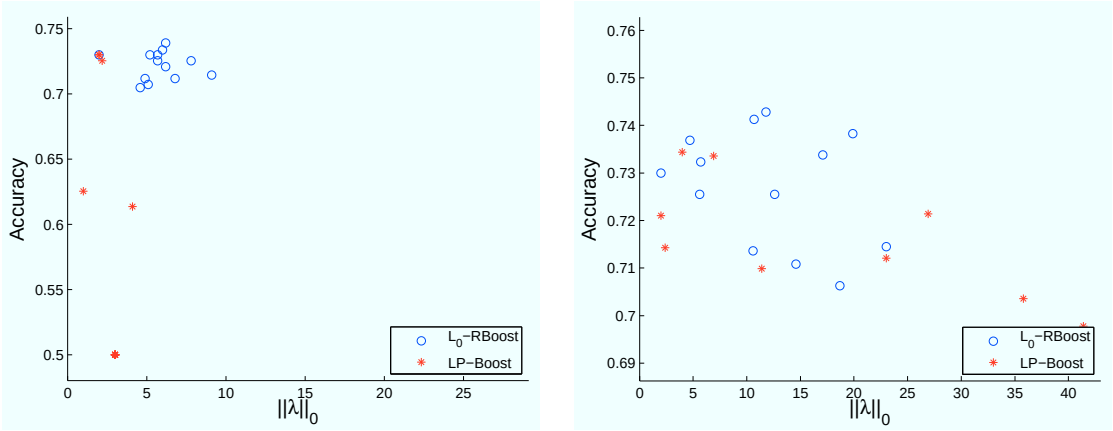
(b) $K = 5$

Figure 3.8: Test accuracy vs. $||\lambda||_0$ on the CLVHEART dataset



(a) $K = 1$ (b) $K = 5$

Figure 3.9: Test accuracy vs. $||\lambda||_0$ on the HUHEART dataset



(a) $K = 1$ (b) $K = 5$

Figure 3.10: Test accuracy vs. $||\lambda||_0$ on the SONAR dataset

Chapter 4

Generalized agreement problems

Agreement problems involve finding the hypothesis, or base classifier, that best agrees with the data, in a set of available base classifiers. The base learning, or pricing, problem of the linear programming formulation (1.13) motivates the well known maximum agreement problem. The tightening of the linear programming formulation (1.13), using cutting planes (3.12), results in the new base learning problem (3.16). This chapter briefly explores this new base learning problem.

4.1 Formulation of the problem with abstaining base classifiers

To further illustrate the new base learning problem (3.16) with abstaining base classifiers, we can define it as a graph theoretic problem in a directed bipartite graph. We maintain the assumption that our base classifiers abstain; for all $u \in \mathcal{U}$ that either $h_u(x) \in \{0, 1\}$ or $h_u(x) \in \{-1, 0\}$. Such abstaining classifiers are used to identically classify a (perhaps small) subset $I \subseteq \{1, \ldots, M\}$ as either positive with $h_u(A_i) = 1$ for all $i \in I$, or negative with $h_u(A_i) = -1$ for all $i \in I$, and abstain otherwise with $h_u(A_i) = 0$ for all $i \in \{1, \ldots, M\} \setminus I$. Our use of abstaining base classifiers is without loss of generality because ($\mathcal{U}$ being at most twice as large) for any non-abstaining $h_u : \mathbb{R}^N \to \{-1, 0, 1\}$ there exist $h_{u^+} : \mathbb{R}^N \to \{0, 1\}$ and $h_{u^-} : \mathbb{R}^N \to \{0, -1\}$ such that $h_u = h_{u^+} + h_{u^-}$. We further assume that for each $u^+ \in \mathcal{U}$ there is a corresponding negative base classifier $u^- \in \mathcal{U}$ such that $h_{u^-} = -h_{u^+}$.

We can now formulate the problem using some predefined set system $\mathcal{E} \subseteq 2^{\{1,\ldots,M\}}$, where each set $S \in \mathcal{E}$ corresponds to the support of h_u for some $u \in \mathcal{U}$, *i.e.*,

$$\mathcal{E} = \{I \subseteq \{1, \ldots, M\} \mid \exists u \in \mathcal{U} \text{ such that } i \in I \Leftrightarrow h_u(A_i) \neq 0\}.$$

We also define a directed bipartite graph $G = (\Omega^+, \Omega^-, \mathcal{A})$ with "positive" vertices Ω^+ and "negative" vertices Ω^-. Each arc (i, i') has a nonnegative weight $v(i, i') \geq 0$ corresponding to some reward for correctly classifying i, and distinguishing it from i'. In our application, the directed arcs $(i, i') \in \mathcal{A}$ correspond to constraints of the form (3.13d) of the LP (3.13), and typically the weights $v(i, i')$ will be determined by the dual variables corresponding to these constraints. Distinguishing between a pair (i, i') implies selecting a set $S \in \mathcal{E}$ such that S is an (i, i')-cut in G, *i.e.*, $i \in S$ but $i' \notin S$. The nonnegative weight $\bar{w}(i) \geq 0$, which is assigned to a vertex i, implies a reward $y_i h_{u(S)}(A_i)\bar{w}(i)$ (or penalty depending on the sign of $y_i h_{u(S)}(A_i)$) for each S such that $S \ni i$.

The net gain for assigning S to be positive is

$$f^+(S)_{\bar{w},v} = \bar{w}(\Omega^+ \cap S) - \bar{w}(\Omega^- \cap S) + v\left(\left(\Omega^+ \cap S\right) \times \left(\Omega^- \setminus S\right)\right),$$

and the net gain for assigning S to be negative:

$$f^-_{\bar{w},v}(S) = \bar{w}(\Omega^- \cap S) - \bar{w}(\Omega^+ \cap S) + v\left(\left(\Omega^- \cap S\right) \times \left(\Omega^+ \setminus S\right)\right).$$

We are free to assign S to be either a positive base classifier or a negative one; specifically, if

$$f^+_{w,v}(S) \geq f^-_{w,v}(S),$$

we assign S to be positive (by associating it with u^+), and otherwise we assign S to be negative (by associating it with u^-. Thus, the base learning problem is to find a set $S \in \mathcal{E}$ such that

$$f_{w,v}(S) = \max\left\{f^+_{w,v}(S), f^-_{w,v}(S)\right\} - c(S) \tag{4.1}$$

is maximized, where $c(S)$ is the cost of set $S \in \mathcal{E}$.

When $|\mathcal{E}|$ is polynomial in M, then it trivially follows that $\max_{S\in\mathcal{E}} f_{G,w,v}(S)$ can be solved in polynomial time in M. In the general case, however, it is unsurprising that the problem is $\mathcal{NP}$-hard. This is confirmed in Section 3.5 in the special case where the sets in $\mathcal{E}$ correspond to monomials, and finding S that maximizes (4.1) solves the maximum monomial agreement problem.

On the other hand, for some of the simple base classifiers considered in the literature, such as monomials of fixed degree, or decision trees of a single test (also known as "decision stumps") [35], maximization of (4.1) can be performed in polynomial time by simple enumeration. We now consider the case where $\mathcal{U}$ corresponds to abstaining monomials, and extend the branch-and-bound algorithm of Chapter 2 to solve this pricing problem. We used this algorithm in the experiments of Section 3.6.

4.2 Extending the Maximum Monomial Agreement algorithm

We extend Algorithm 1 to solve the pricing problem (3.16) on the space of Boolean monomials, which corresponds to finding a monomial (J, C) that maximizes the objective (3.21). We will use the same subproblem definition as Algorithm 1, of a partition of the variable indices into the 4-tuple (J, C, E, F).

First, we note that a simple solution to extending Algorithm 1 in order to apply it to the new pricing problem can be obtained by fixing the class of the monomial and defining the weight $w(\cdot)$ such that

$$w(i) = \begin{cases} \bar{w}(i) + \sum_{i' \in \Omega^-} v(i, i') & i \in \Omega^+ \\ \bar{w}(i) & i \in \Omega^-. \end{cases},$$

If restricting the class of S to be positive, the upper bound (2.7) can now be written as

$$b^+(J, C, E) = \sum_{\eta=1}^{q(E)} \left(w_\eta^+(J, C, E) - w_\eta^-(J, C, E) \right)_+. \tag{4.2}$$

When S is negative, we similarly define

$$b^-(J, C, E) = \sum_{\eta=1}^{q(E)} \left(w_\eta^+(J, C, E) - w_\eta^-(J, C, E) \right)_+. \tag{4.3}$$

If we consider only the case of positive S, we can now solve (3.21), using the upper bound

$$b^+(J, C, E) - c(|J| + |C|)$$

for subproblem (J, C, E, F), given that $c(k)$ is monotonically increasing in k; for any $(J', C') \in P(J, C, E) : |J'| + |C'| \geq |J| + |C|$ and therefore $c(|J'| + |C'|) \geq c(|J| + |C|)$. Thus, by simply adding the inequality $b^+(J, C, E) \geq w^+(J', C') - w^-(J', C')$, we obtain

$$b^+(J, C, E) - c(|J| + |C|) \geq w^+(J', C') - w^-(J', C') + c(\left|J'\right| + \left|C'\right|),$$

which proves property 2.4 and the validity of this bound. We solve a separate, similar problem for negative S, and choose the solution which obtains the maximum value.

However, in order to avoid having to solve the problem twice it is possible to explicitly extend the upper bound (2.7) for the new pricing problem. The resulting upper bound on the objective value of any subproblem $(J', C') \in P(J, C, E)$ is

$$\begin{aligned}\bar{b}(J, C, E) &= \max\left\{b^+(J, C, E), b^-(J, C, E)\right\} - c(|J| + |C|) \\ &= \max\left\{\begin{array}{l} \sum_{\eta=1}^{q(E)} \left(\sum_{i \in V_\eta^+} w_i - \sum_{i \in V_\eta^-} w_i + \sum_{\substack{i \in V_\eta^+, \\ i' \in \Omega^-}} v_{ii'}\right)_+ , \\ \sum_{\eta=1}^{q(E)} \left(\sum_{i \in V_\eta^-} w_i - \sum_{i \in V_\eta^+} w_i + \sum_{\substack{i \in V_\eta^-, \\ i' \in \Omega^+}} v_{ii'}\right)_+ \end{array}\right\} - c(|J| + |C|).\end{aligned} \tag{4.4}$$

We now consider tightening of this upper bound. As before, a choice of variables E to exclude from a monomial defines a partition of the set of observations into subsets of inseparable observations: $V_1^E, V_2^E, ..., V_q^E \subseteq \Omega^+ \cup \Omega^-$. For simplicity of notation, we omit the set E when it is clear in the context, so that $V_\eta = V_\eta^E$. For brevity, we also let $C_{J,C} = \text{Cover}(J, C)$ and $C_{J,C}^+ = \text{Cover}(J, C) \cap \Omega^+$. We also let $V_\eta^+ = \Omega^+ \cap V_\eta$. Furthermore, the sets $C_{J,C}^-$ and V_η^- are similarly defined. We can tighten the bound (4.4) by observing that for any $(J', C') \in P(J, C, E)$, and any $\eta \in \{1, \ldots, q(E)\}$, if $V_\eta \cap C_{J',C'} \neq \emptyset$ then $V_\eta \subseteq C_{J',C'}$. Thus, for $(i, i') \in V_\eta^+ \times V_\eta^-$, we may subtract $v(i, i')$ from each term corresponding to the positive case of (4.4), and similarly for each $(i, i') \in V_\eta^- \times V_\eta^+$ and each term corresponding to the negative case.

The tightened upper bound can then be stated as

$$b(J,C,E) = \left\{ \begin{array}{l} \sum_{\eta=1}^{q(E)} \left(\bar{w}(V_\eta^+) - \bar{w}(V_\eta^-) + v(V_\eta^+ \times \Omega^-) - v(V_\eta^+ \times V_\eta^-) \right)_+ \\ \sum_{\eta=1}^{q(E)} \left(\bar{w}(V_\eta^-) - \bar{w}(V_\eta^+) + v(V_\eta^- \times \Omega^+) - v(V_\eta^- \times \backslash V_\eta^+) \right)_+ \end{array} \right\} - c(|J| + |C|)$$

$$= \max \left\{ \sum_{\eta=1}^{q(E)} \left(f^+_{\bar{w},v}(V_\eta) \right)_+ , \sum_{\eta=1}^{q(E)} \left(f^-_{\bar{w},v}(V_\eta) \right)_+ \right\} - c(|J| + |C|). \tag{4.5}$$

In our computational experiments, we have used the branching scheme described in Chapter 2 branching on $k = |F|$ variables, with the new upper bound (4.5). The branching scheme with $k = |F|$ is both easy to implement and effective when applying a bound on the degree of a monomial, that is, $|J| + |C| \leq K$ for some $K \in \{1, \ldots, N\}$. In particular we found the algorithm very fast in the experiments of Section (3.6) when $K = 5$. In this case, with $K = 5$, although the problem can be solved in $O(N^5)$, and thus, polynomial time, the problem can be computationally intensive with simple enumeration. Moreover, in limited computational experiment without degree constraints, we have not observed any significant increase in the CPU time or branch-and-bound search nodes for the subproblem, compared with the basic subproblem solved by Algorithm 1. In fact, we often find that the increase in CPU time is insignificant compared with the increase in CPU time for the procedure which adds a set of cuts to the master problem.

Chapter 5

Conclusions and possible future work

In this dissertation, we first addressed the base learning problem for an important class of base classifiers, of Boolean monomials. We improved on a previous branch-and-bound algorithm [40] for finding a monomial that best agrees with a given set of binary data by making use of a tighter combinatorial upper bound for a subproblem's objective value. We also devised a dynamic branching scheme which makes use of information in the input dataset in order to choose variables to branch on. Although solving a hard combinatorial optimization problem, we showed that our branch-and-bound algorithms were able to solve small as well as medium size instances of the UCI repository [3].

We then proposed a straightforward extension of the well known discrete MDH problem. The discrete SMDH problem explicitly minimizes the sum of classification error and a penalty that is a linear function of the features selected to be used (or more precisely, linear in the indicator variables μ_u, which each indicate whether a feature u is used). We then offered a new interpretation of common "soft margin" LP formulations for finding weighted voting classifiers, as the continuous relaxations of the SMDH problem.

We introduced sparsity cutting planes for tightening the well known "soft margin" LP relaxation of SMDH. The number of all possible cutting planes of this type is $|\Omega^+|\,|\Omega^-|$. We demonstrated the effectiveness of our cutting planes for finding sparse separating hyperplanes in practice.

However, the sparsity cutting planes which we propose may not be sufficient to obtain an integer solution. Further strengthening of the cuts may be possible by using and extending techniques suggested for the set cover problem [4, 22] and recently for the related maximum feasible subset problem [56]. In current practical solution techniques, cutting planes alone are not sufficiently effective in finding integer solutions. Therefore, cutting methods are

usually combined with a branching scheme to obtain an integer solution. In our application with small instances it may be straightforward to directly branch on the variables ξ and μ. However, we are also interested in instances where dimension of μ can be very large. In this case, we may be interested in implementing a branch-and-price approach [5] for generating the base classifiers and the corresponding nonzero variables λ and μ. Branch-and-price involves sophisticated branching schemes for preventing the regeneration of a column by column generation after branching sets a variable to zero; in our case, such a branching scheme would have to be devised in order to indirectly branch on the μ variables.

We proved the correctness of the MIP formulation that we suggested for SMDH with $H \in \{-1, 0, 1\}^{M \times U}$. The correctness proof depends on an upper bound which we have derived for a component of the optimal solution hyperplane normal vector λ, and the assumption of $H \in \{-1, 0, 1\}^{M \times U}$. It is interesting to try to derive such an upper bound with real valued entries, or $H \in [-1, 1]^{M \times U}$.

We have derived inapproximability results for SMDH by reduction of the MDH problem while making use of an upper bound on the penalty term. While computational complexity has not been the focus of this dissertation, it was instrumental in motivating the solution of the SMDH relaxation. It is interesting to investigate whether our inapproximability results can be improved. Also, it would be interesting to investigate the computational complexity of SMDH with other values of the penalty parameter, such as $C \in O(\log M)$. It might also be interesting to research approximation algorithms for SMDH.

Finally, we considered predetermined complexity penalties for the case of Boolean monomials, which we derived from the length of a code that we proposed for encoding the base classifiers in this case. We could look into different type of codes for deriving the penalty parameters c, or just experiment and select the penalties from some interval via cross-validation. It would be interesting to experiment with other measures of information complexity that make use of the given data to compute the penalties such as Rademacher complexity [7]. The latter, when computed empirically, may apply more generally to different classes of base classifiers.

References

[1] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.

[2] Sanjeev Arora, László Babai, Jaques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and Systems Sciences*, 54:317–331, 1997.

[3] Arthur Asuncion and David J. Newman. UCI machine learning repository, 2007.

[4] Egon Balas and Shu Ming NG. On the set covering polytope: I. all the facets with coefficients in $\{0, 1, 2\}$. *Mathematical Programming*, 43(1-3):57–69, 1989.

[5] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[6] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):526–536, 1998.

[7] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[8] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[9] Shai Ben-David, Nadav Eiron, and Philip M. Long. On the difficulty of maximizing agreements. *Journal of Computer and Systems Sciences*, 66(3):496–514, 2003.

[10] Kristin P. Bennett and Erin J. Bredensteiner. A parametric optimization method for machine learning. *INFORMS Journal of Computing*, 9(3):311–318, 1997.

[11] Kristin P. Bennett and Erin J. Bredensteiner. Duality and geometry in SVM classifers. *Proceedings of the 17th International Conference on Machine Learning*, pages 57–64, 2000.

[12] Kristin P. Bennett and Olvi L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–34, 1992.

[13] Avrim Blum and John Langford. PAC-MDL bounds. In *COLT*, pages 344–357, 2003.

[14] Koen M. J. De Bontridder, B. J. Lageweg, Jan K. Lenstra, James B. Orlin, and Leen Stougie. Branch-and-bound algorithms for the test cover problem. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, pages 223–233, London, UK, 2002. Springer-Verlag.

[15] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, and Alexander Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[16] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306, 2000.

[17] Joel Brenner. The Hadamard maximum determinant problem. *The American Mathematical Monthly*, 79(6):626–630, 1972.

[18] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[19] N.H. Bshouty and L. Burroughs. Maximizing agreements and coagnostic learning. *Theoretical Computer Science*, 350(1):24–39, 2006.

[20] William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *In Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 335–342, 1999.

[21] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression and AdaBoost and Bregman distances. *Machine Learning*, 48:253–285, 2002.

[22] Gérard Cornuéjols and Antonio Sassano. On the 0, 1 facets of the set covering polytope. *Mathematical Programming: Series A and B*, 43(1):44–55, 1989.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[24] David R. Cox and E.J. Snell. *Analysis of binary data*. CRC Press, 1989.

[25] Yves Crama, Peter L. Hammer, and Toshihide Ibaraki. Cause-effect relationships and paritally defined Boolean functions;. *Annals of Operations Research*, 16:299–326, 1988.

[26] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[27] David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and Systems Sciences*, 52(3):453–470, 1996.

[28] Jonathan Eckstein and Noam Goldberg. An improved branch-and-bound method for maximum monomial agreement. In *OPT 2008 Optimization for Machine Learning, NIPS 2008 Workshop*, 2008.

[29] Jonathan Eckstein, Peter L. Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.

[30] Jonathan Eckstein, Cynthia A. Phillips, and William E. Hart. PEBBL 1.0 user guide. RUTCOR Research Report RRR 19-2006, RUTCOR, Rutgers University, 2006.

[31] Ran El-Yaniv, Dmitry Pechyony, and Elad Yom-Tov. Better multiclass classification via a margin-optimized single binary problem. *Pattern Recognition Letters*, 29:1954–1959, 2008.

[32] Theodoros Evgeniou, Tomas Poggio, Massimiliano Pontil, and Alessandro Verri. Regularization and statistical learning theory for data analysis. *Computantional Statistics and Data Analysis*, 38:421–432, 2002.

[33] Vitaly Feldman. Optimal hardness results for maximizing agreements with monomials. In *Proceedings of the Annual IEEE Conference on Computational Complexity*, pages 226–236, 2006.

[34] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[35] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *In Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[36] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139, 1997.

[37] Jerome H. Friedman. Fast sparse regression and classification. Technical report, Stanford University, 2008.

[38] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *Annals of Applied Statistics*, 2(3):916–954, 2008.

[39] Paul C. Gilmore and Ralph E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961.

[40] Noam Goldberg and Chung-chieh Shan. Boosting optimal logical patterns. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, 2007.

[41] Thore Graepel, Ralf Herbrich, Bernhard Schölkopf, Alex Smola, Peter Bartlett, Klaus-Robert Müller, Klaus Obermayer, and Robert Williamson. Classification on proximity data with lp-machines. *International Conference of Artificial Neural Networks*, pages 304–309, 1999.

[42] Peter D. Grünwald. *The minimum description length principle.* MIT Press, 2007.

[43] David Heath. *A geometric framework for machine learning.* PhD thesis, Johns Hopkins University, 1992.

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

[45] Michael Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4):807–837, 1993.

[46] Michael Kearns and Leslie G. Valiant. Learning Boolean formulae or finite automata is as hard as factoring. Technical Report TR-14-88, Harvard University Aiken Computation Laboratory, August 1988.

[47] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, January 1994.

[48] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[49] Subhash Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[50] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer-Verlag, 2002.

[51] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[52] Olvi L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13(3):444–452, 1965.

[53] Olvi L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[54] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning*, pages 118–183, 2003.

[55] Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227–234, 1995.

[56] Marc E. Pfetch. Branch-and-cut for the maximum feasible subsystem problem. *SIAM Journal on Optimization*, 19:21–38, 2008.

[57] Leonard Pitt and Leslie Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[58] Gunnar Rätsch, T. Onoda, and K . R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320, 2001.

[59] Gunnar Rätsch, Bernhard Schölkopf, Alex J. Smola, Sebastian Mika, Takashi Onoda, and Klaus-Robert Müller. Robust ensemble learning. In Alex J. Smola, Peter J. Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000.

[60] Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

[61] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(5):197–227, 1990.

[62] Robert E. Schapire. *The boosting approach to machine learning: an overview*. Springer Verlag, 2003.

[63] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:1651–1686, 1998.

[64] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rate predictions. *Machine Learning*, 37:297–336, 1999.

[65] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[66] Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. The MIT Press, 2002.

[67] Rocco A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

[68] John Shawe-taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44:1926–1940, 1998.

[69] Vladimir N. Vapnik. *Statistical learning theory*. John Wiley and Sons, 1998.

[70] Vijay V. Vazirani. *Approximation algorithms*. Springer Verlag, 2003.

[71] Ulrike von Luxburg, Olivier Bousquet, and Bernhard Schölkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5:293–323, 2004.

[72] Wotao Yin and Yin Zhang. Extracting salient features from less data via ℓ_1-minimization. *SIAG/OPT Views-and-News*, 19:11–19, 2008.

[73] Tong Zhang and Bin Yu. Boosting with early stopping: convergence and consistency. *Annals of Statistics*, 33(4):1538–1579, 2005.

Vita

Education

- PhD, Rutgers Center for Operations Research (RUTCOR), Rutgers University, New Brunswick, NJ, 2010
- Master of Science, Leon Recanati School of Business, Tel Aviv University, Tel Aviv, Israel, 2004
  - Decisions and Operations Research
- Bachelor of Science, University of Toronto, Toronto, Ontario, Canada, 1998
  - Computer Science major
  - Graduated with High Distinction
- Bachelor of Business Administration, Schulich School of Business, York University, Toronto, Ontario, Canada, 1996

Work experience

- Co-op/ part-time position, Telcordia Applied Research, Piscataway, NJ, December 2008 - August 2009
  - Researched algorithms for assignment of link weights for congestion avoidance in shortest path routing.
  - Researched optimization formulations for the joint routing and channel assignment problem in wireless networks.
- Intern, Sandia National Lab, Livermore, CA, Summer 2008
  - Mentors: Dr. Tamara G. Kolda and Dr. Ann Yoshimura
  - Researched global derivative free optimization methods and specifically extending the DIRECT algorithm for using external trial points.
- Graduate Research Assistant, Rutgers University, New Brunswick, NJ, 2007-2008
  - Mentors: Prof. Paul Kantor and Prof. Endre Boros
  - NSF funded project, "Deceptive Detection Strategies for Container Inspection".
  - Duties included basic research and implementation of optimization algorithms.
- Teaching Assistant, Rutgers University, New Brunswick, NJ, 2004-2007
  - Taught recitations and graded courses in Precalculus, Calculus II and Theory of Linear Optimization.

- Summer Research Project, Prof. Peter Hammer, RUTCOR (NSF and NIH funded project). August 2005.
  - Implemented a local search and simulated annealing algorithm for fine tuning Logical Analysis of Data (LAD) parameters through cross-validation.

Publications

- Noam Goldberg and Chung-chieh Shan, *Boosting Optimal Logical Patterns Using Noisy Data*, Proceedings of the SIAM International Conference on Data Mining, 2007.
- Noam Goldberg, Tamara G. Kolda and Ann Yoshimura, *Concurrent Optimization with DUET: DIRECT Using External Trial Points*, Sandia National Labs, Technical Report #SAND2008-5844.
- Noam Goldberg, Jonathan Word, Endre Boros and Paul Kantor, *Optimal Sequential Inspection Policies*, RUTCOR Research Report (RRR) #14-2008, also to appear in Annals of Operations Research.
- Jonathan Eckstein and Noam Goldberg, *An Improved Branch-and-Bound Method for Maximum Monomial Agreement*, RUTCOR Research Report (RRR) #14-2009, also presented at NIPS Workshop in Optimization for Machine Learning, Whistler BC, Canada, 2008.

Non academic work experience

- Operations Research Intern, Health Products Research, Strategic Planning Department, Whitehouse, NJ, Summer 2005.
- System Engineer, ECI Telecom, Petach Tikva, Israel, 2000-2004
- Software Engineer, ECI Telecom, Petach Tikva, Israel, 1998-2000