UNIVERSITY OF CALIFORNIA
SANTA CRUZ

ENTROPY REGULARIZATION AND SOFT MARGIN MAXIMIZATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Karen A. Glocer

December 2009

The Dissertation of Karen A. Glocer
is approved:

Professor Manfred Warmuth, Chair

Professor Darrell Long

Dr. James Theiler

Professor S.V.N. Vishwanathan

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Entropy Regularization and Soft Margin Maximization

by

Karen A. Glocer

In the machine learning community, margin maximization is a popular proxy for good generalization. When the data are not separable by a linear combination of hypotheses, maximizing the soft margin is considered to be more robust than maximizing the margin. LPBoost is a boosting algorithm that directly maximizes the soft margin by solving a linear programming problem. However, we can exhibit cases where the number of iterations is linear in the number of examples instead of logarithmic. Moreover, we show both theoretically and experimentally that this algorithm can exhibit extremely poor generalization error. This suggests that maximizing the soft margin by itself may be a poor proxy for good generalization performance. We introduce three algorithms that address the instability of LPBoost and have iteration bounds that are logarithmic in the number of examples. The first algorithm, SoftBoost, is based on relative entropy projection. The other two algorithms, Entropy Regularized LPBoost and Binary Entropy Regularized LPBoost, add relative entropy and binary relative entropy regularization to the soft margin respectively. Moreover, in the first large-scale experimental evaluation of optimization-based boosting algorithms, we demonstrate that even a small amount of entropy regularization suffices to stabilize LPBoost.

To my parents.

Acknowledgments

I would like to thank my advisor, Manfred Warmuth. I would like to thank my committee member S.V.N. Vishwanathan for teaching me about optimization, for serving on my committee, and for his contributions to [90]. I would like to thank James Theiler for many years of mentoring, financial support, for technical help, and for serving on my committee. I would like to thank Darrell Long for serving on my committee and for his extensive support. I would like to thank Gunnar Rätsch for the use of his cluster and for his contributions to [87]. I would like to thank the ISIS team and the ISSDM at Los Alamos National Laboratory for their financial support. I would like to thank Dima Kuzmin for reading through my proofs, giving me helpful suggestions, and driving all the way to Santa Cruz to keep me company writing. Also, Damian Eads and Ed Rosten have helped me in countless ways, from wonderful discussions, programming advice, an introduction to python, and flat out wonderful company. I would like to thank Maya Hristakeva and Wouter Koolen for their valuable comments. Also, I would like to thank Nancy David, Clint Scovel, Don Hush, James Howse, and Ingo Steinwart for many hours of interesting discussion and a great deal of inspiration. I would also like to thank my parents and my wonderful friends, whose unquestioning support and faith sustained me. Finally, significant portions of the text of this thesis is drawn from papers published in NIPS 2007 [90], ALT 2008 [87], and an ICML 2009 tutorial [89].

Chapter 1

Introduction

In binary classification problems, we are given a training set of $\pm$ labeled examples drawn i.i.d. from some underlying distribution. We use these labeled examples to find a function that maps each example to its label as accurately as possible. For a given example/label pair, the accuracy of this mapping is determined by a *loss function*. The *empirical risk* is the expected loss over all of the examples in the training set. The most common loss function used in binary classification is the *zero-one* loss. In the zero-one loss, classifying an example correctly incurs no loss and classifying an example incorrectly incurs a loss of 1. We define *empirical error* as the empirical risk with respect to the zero-one loss. Similarly, we define the *generalization error* as the risk with respect to the zero-one loss on data from the underlying distribution.

The ultimate goal of any classification algorithm is find a function that minimizes generalization error, but generalization error can only be minimized directly if the underlying distribution that generates the data is known. A common approach is

to minimize empirical error on a data set drawn i.i.d. from the underlying distribution in the hope that this will achieve a low generalization error.

We define a hypothesis as a function that predicts the label for each example. In the simplest case, a hypothesis makes predictions in the range $\{\pm 1\}$. A slightly richer class of hypotheses makes real-valued predictions in the range $[-1, 1]$, and this is the sort of hypothesis used in this thesis. Given a set of hypotheses $\mathcal{H}$, we say that data is linearly separable with respect to $\mathcal{H}$ if there is a linear combination of hypotheses $h \in \mathcal{H}$ that can correctly classify every example. When we say that the data is linearly separable, it is implicitly assumed that this is with respect to $\mathcal{H}$.

Boosting is a machine learning framework which combines a set of weak hypotheses, each of which performs slightly better than random guessing, into a strong hypothesis that is considerably better than random. To combine these hypotheses effectively, boosting algorithms weigh the examples in the training set such that the examples that are the most difficult to classify are given the most weight. For the algorithms proposed in this thesis, the weights are probability distributions. At each iteration, this distribution is given to an oracle, which must return a hypothesis that has a certain weak guarantee with respect to the current distribution on the examples. Intuitively, the hypothesis returned by the oracle should have a high weighted accuracy. The algorithm then redistributes the weight on the examples so that more weight is put on the examples that are harder to classify. In the next iteration, the updated distribution is given to the oracle, which provides a new hypothesis with the same weak guarantee for the new distribution. The boosting algorithm incorporates each new hypothesis

into its master hypothesis, which is a linear combination of the weak hypotheses it has received from the oracle. This continues until the boosting algorithm terminates, at which time it returns its master hypothesis which is a linear combination of all of the weak hypotheses it has received from the oracle.

In boosting there are two sets of weights: the weights on the examples and the weights on the hypotheses. Boosting algorithms are defined by the way they determine these weights. These two sets of weights are related by the *margin* and the *edge*, both of which are defined in Chapter 2. The margin of an example is the weighted accuracy of all of the hypotheses for a single example. Similarly, the edge of a hypothesis is the weighted accuracy of all of the examples for a single hypothesis.

In many boosting algorithms, the updated weight on the examples is a function of the margin of each example. For instance, AdaBoost [30], the most well known boosting algorithm, assigns each example weight proportional to the negative exponential of its margin. Even after the final linear combination classifies all training examples correctly, the generalization performance of AdaBoost has been observed to improve with additional iterations [61]. This is attributed to the observation that the minimum margin of the examples for the final linear combination continues to increase even after the training error stabilizes [71]. The problem with AdaBoost is that it only approximately maximizes the margin [68]. Furthermore, its performance deteriorates when the examples are not linearly separable [22].

There is a boosting algorithm that uses linear programming to maximize the margin explicitly. This algorithm is known as LPBoost [36, 64, 21]. In the noisy case,

the examples are not linearly separable and the hard margin becomes negative. To alleviate this problem, slack variables are introduced and LPBoost maximizes the soft margin. The resulting soft margin, defined precisely in Chapter 2, is considered to be more robust than the margin when the data is not linearly separable.

Although LPBoost is often used in practice, the analysis of this algorithm has proved to be elusive. In particular, there are no known iteration bounds, and we show in this thesis why this is the case. This result, originally shown in [87], is a lower bound on the number of iterations required by LPBoost that is linear in the number of examples. This is significant because a good iteration bound in the boosting context is one that is logarithmic in the number of examples. In this thesis we also show that any linearly separable dataset can be reduced to a dataset on which LPBoost misclassifies *all* examples by adding a bad example and a bad hypothesis. Although both of the results we have discussed are derived for the hard margin case, they can be extended to the soft margin case as well.

These two results suggest that directly maximizing the margin does not result in a robust boosting algorithm. In this thesis we present three boosting algorithms that provably maximize the soft margin but do not exhibit the same brittleness as LPBoost. These algorithms are SoftBoost, Entropy Regularized LPBoost, and Binary Entropy Regularized LPBoost. These algorithms share two important characteristics: they are motivated by either the relative entropy or the binary relative entropy, and given a sample of size N, these algorithms require at most $O(\frac{1}{\epsilon^2}\ln\frac{N}{\nu})$ iterations to produce a hypothesis within ϵ of the optimal soft margin. Here $\nu \in [1, N]$ is the *capping* parameter

that limits how much weight a given example can have. Capping is a common approach to making boosting algorithms robust to noise by preventing too much weight from accumulating on a few noisy examples.

The first algorithm, SoftBoost [87], minimizes the relative entropy to the initial distribution subject to linear constraints on the edges of all the hypotheses obtained so far. The edge, defined in Chapter 2, is the weighted accuracy of a hypothesis with respect to a distribution. The upper bound on the edges is gradually decreased, and this leads to the main problem with SoftBoost: the generalization error decreases slowly in early iterations. Although SoftBoost provably maximizes the minimum soft margin, the mechanism for doing so is not quite satisfactory, resulting in a problematic slow start in our experiments.

The second algorithm, Entropy Regularized LPBoost (ERLPBoost) [90], adds $\frac{1}{\eta}$ times the relative entropy to the initial distribution to the maximum soft edge objective of LPBoost and minimizes the resulting sum. A number of similar algorithms (such as ν-Arc [64]) were discussed in [62], but no iteration bounds were proved for them even though they were shown to have good experimental performance. Most recently, a similar boosting algorithm was considered by [69, 70] based on the Lagrangian dual of the optimization problem that motivates the ERLPBoost algorithm. However the $O(\frac{1}{\epsilon^{>3}} \ln N)$ iteration bound proved for the algorithm in [69, 70] is weaker than the ones considered in this thesis.

The third algorithm is Binary Entropy Regularized LPBoost (Binary ERLPBoost). It adds $\frac{1}{\eta}$ times the *binary* relative entropy to the initial distribution to the

optimization problem that defines LPBoost and minimizes the resulting sum. There is no *a-priori* reason to believe that one type of regularizer will result in lower generalization error than the other. Rather, the appeal of the binary entropy is that it implicitly enforces the soft margin constraints, which reduces the complexity of the optimization problem that must be solved at each iteration to compute the updated distribution on the examples. Interestingly, although the regularizer changes, the iteration bound remains the same.

Finally, this thesis presents the first experimental analysis of optimization-based boosting algorithms on large scale data. In our experiments, the oracle will return either decision stump hypotheses, raw data hypotheses, or SVM hypotheses. These are discussed in detail in Chapter 5. We will see that the results are heavily dependent on the class of hypotheses returned by the oracle, so when we discuss our results, we will specify the algorithm, the data set, and the hypothesis class. Our experimental evaluation addresses five principal questions.

In the first set of experiments, we demonstrate that LPBoost is unstable in practice. We then asked whether the instability is corrected by either replacing the hard margin with the soft margin or by adding entropy regularization. The experimental results show that the soft margin does not stabilize LPBoost. However, not only does entropy regularization result in a stable algorithm, even a small amount of regularization suffices.

In the second set of experiments, we ask whether the theoretical value of the regularization parameter for ERLPBoost and Binary ERLPBoost that we define in

Chapter 3 is optimal, and if not, what constitutes sufficient regularization? We found that the theoretical value of the regularization parameter is generally a good choice, but that in many cases, more regularization does not hurt performance.

In the third set of experiments, we compare the binary entropy regularizer and the relative entropy regularizer. In Chapter 3 we show that for the relative entropy regularizer, capping in the $\mathbf{d}$ domain results in slack variables in the $\mathbf{w}$ domain, which increases the complexity of the optimization problem. In contrast, for the binary entropy, capping in the $\mathbf{d}$ domain does not increase the complexity of the optimization problem in the $\mathbf{w}$ domain. Because the optimization problems are all implemented in the $\mathbf{w}$ domain, this suggests that the binary entropy regularizer should be faster than the relative entropy regularizer, but there is no reason to suspect that one will have better generalization performance than the other. We found that the binary entropy is not always faster than the relative entropy but their generalization performance is similar.

In the fourth set of experiments, we compare the corrective and totally corrective algorithms. The *corrective* family of boosting algorithms only update the weights on the examples based on the *last* hypothesis, while the *totally corrective* family of algorithms update the weights based on *all* past hypotheses. This is covered in Chapter 4 ERLPBoost and Binary ERLPBoost, and SoftBoost belong to the family of totally corrective algorithms. Shalev-Schwartz and Singer [77] proposed a corrective version of ERLPBoost. We refer to this algorithm as Corrective ERLPBoost, and it is discussed at length in Chapter 4. In this thesis, we also introduce Corrective Binary ERLPBoost,

a corrective algorithm that uses the binary relative entropy. algorithms. In these experiments, the results are heavily dependent on the hypothesis class. However, the SVM hypotheses outperformed the other hypothesis classes on every data set. For SVM hypotheses, the totally corrective algorithms outperformed the corrective algorithms in both generalization error and time.

In the last set of experiments we compare the best overall generalization error achieved by all of the previously mentioned algorithms for each hypothesis class on each data set. In these experiments we use a train-test-validation methodology described in Chapter 5. While all of the previous experiments tried to elucidate the properties of the algorithms we are studying, this is the evaluation that is of the most interest to practitioners who will eventually use the algorithms. Like the previous set of experiments, these results are heavily dependent on the hypothesis class, and SVM hypotheses outperformed the other hypothesis classes. For SVM hypotheses, the totally corrective algorithms outperformed the corrective algorithms in both generalization error and time.

The thesis is outlined as follows. In Chapter 2, we more rigorously define the LPBoost algorithm and show a lower bound on the iteration bound of LPBoost. In addition, we show that any linearly separable dataset can be reduced to one on which LPBoost misclassifies all examples. In Chapter 3 we present three entropy regularization algorithms that address the instability of LPBoost. These algorithms are SoftBoost, Entropy Regularized LPBoost, and Binary Entropy Regularized LPBoost. For each algorithm we discuss its termination condition, its Lagrangian Dual, the relationship between primal and dual variables, and its iteration bound. In Chapter 4 we discuss the

implementation details of the algorithms we use in our experiments. These experiments can be found in Chapter 5. Chapter 6 puts all of this in the context of the available scientific literature. Finally, the conclusions can be found in Chapter 7.

The results in this thesis have appeared in three places. The SoftBoost algorithm and the lower bound on the iteration bound on LPBoost were first presented in NIPS 2007 [87]. The ERLPBoost algorithm first appeared in ALT 2008 [90]. In addition, a subset of the experimental results in this thesis were also presented in a tutorial at ICML 2009 [89].

Chapter 2

LPBoost

In the boosting setting, we are given a set of N labeled training examples (x_n, y_n), $n = 1 \ldots N$, where the instances x_n are in some domain $\mathcal{X}$ and the labels $y_n \in \pm 1$. Boosting algorithms maintain a distribution $\mathbf{d}$ on the N examples, so $\mathbf{d}$ lies in the N dimensional probability simplex $\mathcal{S}^N$. Intuitively, the examples that are hard to classify are given more weight. The initial distribution $\mathbf{d}^0$ is uniform. In each iteration $t = 1, 2, \ldots$ the algorithm gives the current distribution $\mathbf{d}^{t-1}$ to an oracle (a.k.a. the weak learning algorithm), which returns a new hypothesis $h^t : \mathcal{X} \to [-1, 1]$ from some base hypothesis class $\mathcal{H}$. The hypothesis returned by the oracle comes with a certain guarantee of performance. This guarantee will be discussed in Chapter 2.1.

The most common measure of the performance of a hypothesis h^t is its classification error. When the range of hypothesis h^t is in $\{-1, 1\}$, classification error can be expressed as

$$\frac{1}{N} \sum_{n=1}^{N} I(h^t(x_n) \neq y_n),$$

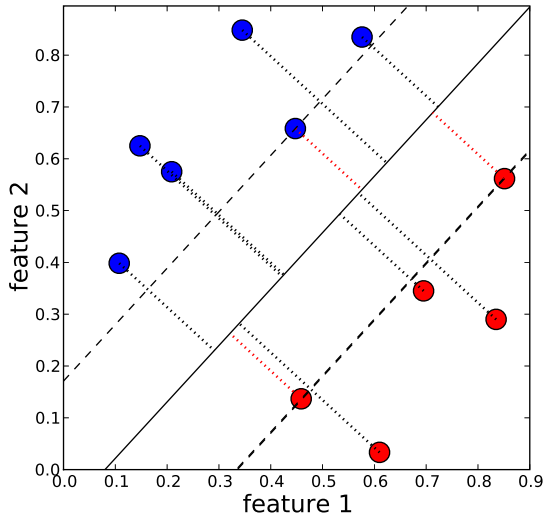


Figure 2.1: Illustration of the margin concept. The *geometric margin* of an example, shown by the dotted lines, is the Euclidean distance between the example and the separating hyperplane. The smallest margins are shown in red.

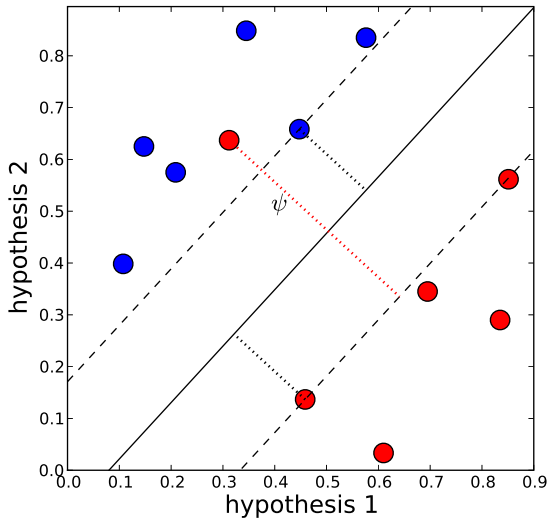


Figure 2.2: Illustration of the soft margin concept. One point does not satisfy the margin requirement, which is shown by the dashed lines. The slack variable ψ is the distance between the misclassified example and dashed line.

Symbol	*Description*
N	Number of examples.
t	Iteration number.
T	Final number of iterations.
$\mathcal{S}^M$	Probability simplex of dimension M.
$\mathbf{x}_n$	n^{th} example.
$\mathcal{X}$	Domain of examples.
$y_n \in \{-1, 1\}$	n^{th} label.
$h^t(x_n) \in [-1, 1]$	Prediction of hypothesis t on example x_n.
$\mathcal{H}$	Hypothesis class.
$u_n^t = y_n h^t(\mathbf{x}_n)$	Convenient notation for combining labels and hypotheses.
$\mathbf{d} \in \mathcal{S}^N$	Distribution on examples.
$\mathbf{w} \in \mathcal{S}^t$	Distribution on hypotheses at iteration t.
$\boldsymbol{\psi} \in [0, \infty]^N$	Slack variable for soft margin problem.
ϵ	Precision parameter.
ν	Capping parameter.
g	Guarantee of the oracle.
P_{LP}^t	The solution to the LPBoost problem at iteration t.
P_{LP}	The solution to the LPBoost problem over *all* hypotheses.

Table 2.1: Notation for margins, edges, and LPBoost

error	edge
0	+1
1	-1
0.5	0

Table 2.2: The relationship between the error and the edge of hypothesis t.

where I is the indicator function. The weighted error of a hypothesis h^t with respect to distribution $\mathbf{d}$ is

$$\epsilon_{h^t} = \sum_{n=1}^{N} d_n I(h^t(x_n) \neq y_n).$$

Note that the weighted error can be defined w.r.t any distribution $\mathbf{d}$, not just the current distribution $\mathbf{d}^{t-1}$.

A more convenient quantity is the edge, which measures the weighted accuracy of a single hypothesis.

Definition 2.1 *The edge of hypothesis h^t w.r.t. distribution $\mathbf{d}$ is defined as*

$$\sum_{n=1}^{N} d_n y_n h^t(x_n).$$

When the range of h^t is ± 1 instead of the interval [-1,1], then the edge is just an affine transformation of the weighted error of the hypothesis h^t because $I(h^t(x_n) \neq y_n) = -2y_n h^t(x_n) - 1$. Note that the edge is also defined when $h^t \in [-1, 1]$. A hypothesis that predicts perfectly has error 0 and edge 1 while a hypothesis that always predicts incorrectly has error 1 edge -1. A random hypothesis has error approximately 0.5 and edge approximately 0. The higher the edge, the more useful the hypothesis is for classifying the training examples. These results are summarized in Table 2.2.

It is convenient to define an N-dimensional vector $\mathbf{u}^t$ that combines the base hypothesis h^t with the labels y_n of the N examples: $u_n^t := y_n h^t(x_n)$. With this notation, the edge of the hypothesis h^t w.r.t. $\mathbf{d}$ becomes simply the dot product $\mathbf{u}^t \cdot \mathbf{d}$. After a hypothesis h^t is received, the algorithm must update its distribution $\mathbf{d}^{t-1}$ on the examples using $\mathbf{u}^t$. At this time, the algorithm also updates its distribution on the hypotheses $\mathbf{w}^t$. The master hypothesis returned by the boosting algorithm is always a convex combination of base hypotheses $f_{\mathbf{w}}(x_n) = \sum_{q=1}^{T} w_q^T h^q(x_n)$, where w_q^T is the coefficient of the hypothesis h^q added at final iteration T.

Just as the edge measures of the weighted accuracy of a hypothesis, the margin measures the weighted accuracy of an example.

Definition 2.2 *We define the* margin *of example* (x_n, y_n) *over hypotheses* $h^1 \dots h^t$ *w.r.t. distribution* $\mathbf{w} \in \mathcal{S}^t$ *as*

$$y_n(\sum_{q=1}^{t} w_q h^q(x_n)) = \sum_{q=1}^{t} w_q u_n^q.$$

Note that for hypotheses $h^1 \dots h^t$, the margin can be defined w.r.t. any $\mathbf{w} \in \mathcal{S}^t$, not just $\mathbf{w}^t$. The margin of a set of examples is taken to be the minimum margin of the set.

When the sign of the master hypothesis $\sum_{q=1}^{T} w_q^T h^q(x_n)$ agrees with the label y_n for all $n = 1 \dots N$, then the examples are separated by the hyperplane defined by the $\mathbf{w}^T$ vector, and margin of the set of examples is positive. When this is the case, we say that the examples are linearly separable. Note that edges are linear in the distribution over the examples and margins are linear in the distribution over the current set of hypotheses.

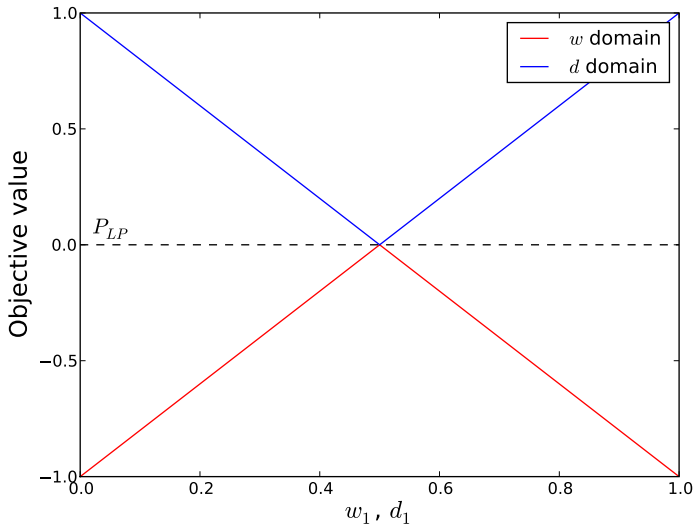


Figure 2.3: The optimization problem solved by LPBoost in both the $\mathbf{d}$ and the $\mathbf{w}$ domain for a simple problem of 2 examples and 2 hypotheses. In the $\mathbf{d}$ domain (blue), the objective function is the l.h.s. of (2.1) and in the $\mathbf{w}$ domain (red), the objective function is the r.h.s. of (2.1).

Edges and margins are related in a fundamental way. By linear programming duality, the minimum-maximum edge equals the maximum-minimum margin:

$$\min_{\mathbf{d}\in\mathcal{S}^N} \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} \;=\; \max_{\mathbf{w}\in\mathcal{S}^t} \min_{n=1,2,\ldots,N} \sum_{q=1}^{t} u_n^q \, w_q. \tag{2.1}$$

When $||\mathbf{w}||_2 = 1$, then $\sum_{q=1}^{T} w_q u_n^q$ is the Euclidean distance between example x_n and the hyperplane defined by $\mathbf{w}$, also known as the geometric margin [17]. However, in the above optimization problem, the constraint $\mathbf{w} \in \mathcal{S}^t$ implies that $||w||_1 = 1$. Thus, boosting is based on the 1-norm margin and support vector machines [6] maximize the geometric margin. Figure 2.1 shows the geometric margins of a set of examples with respect to their separating hyperplane. The minimum margin of the set is shown in red. In this case, three points are actually tied for the smallest margin.

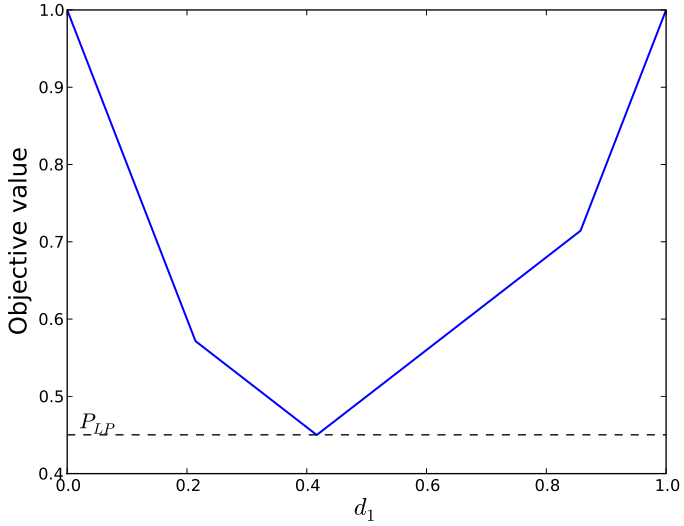


Figure 2.4: The optimization problem solved by LPBoost in the **d** domain for a more complex problem of 2 examples and 4 hypotheses. Because $d_2 = 1 - d_1$, it suffices to show d_1. The objective function is the l.h.s. of (2.1).

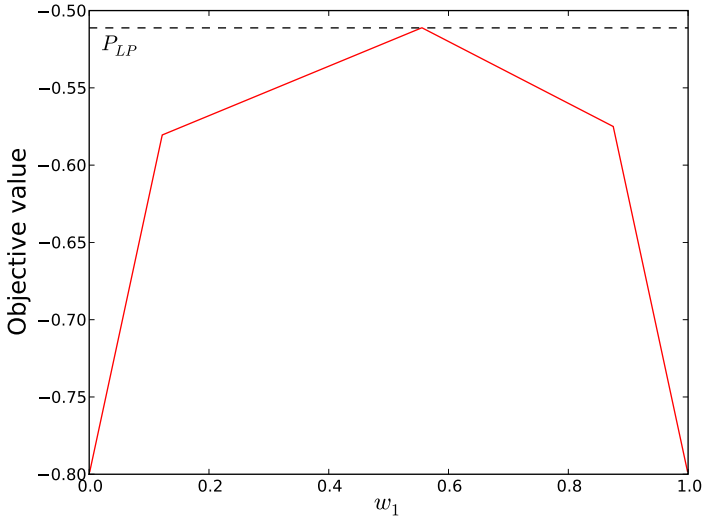


Figure 2.5: The optimization problem solved by LPBoost in the **w** domain for a more complex problem of 4 examples and 2 hypotheses. Because $w_2 = 1 - w_1$, it suffices to show w_1. The objective function is the r.h.s. of (2.1).

Much of the machine learning community works in the margin domain, which we often call the **w** domain or the hypothesis domain because we are optimizing the weights on the hypotheses. In contrast, most of the analysis in this thesis will take place in the edge domain, which we will often call the **d** domain or the example domain because we are optimizing the weights on the examples. As a consequence of (2.1), the optimal values of the two domains are equivalent.

To help visualize these optimization problems, Figure 2.3 plots the value of the objective function in both the **d** and **w** domains for a simple problem of two examples and two hypotheses. Note that this is the only problem size that can be represented in two dimensions in both domains. The red line in Figure 2.3 corresponds to the value on the r.h.s. of (2.1) and the blue line corresponds to the value on the l.h.s. of (2.1). Observe that the minimum value in the **d** domain has the same objective value and the maximum value in the **w** domain. This is precisely what (2.1) leads us to expect. Because there are only two examples, **d** is a vector of length 2. Also, $d_2 = 1 - d_1$, which allows us to represent this as a two-dimensional problem. The same is true in the **w** domain.

Unfortunately, this is not a very interesting problem, and additional insight can be gained from slightly larger problems. Figure 2.4 plots the value of the objective function on the l.h.s. of (2.1) for a problem of 2 examples and 4 hypotheses. Similarly, Figure 2.5 shows the value of the objective function on the r.h.s. of (2.1) for a *different* problem of 4 examples and 2 hypotheses. The problems in these figures are different because it is no longer possible to represent these problem sizes in two dimensions for

both domains.

In the case when examples are not separable by a linear combination of the base hypotheses, then the margins are naturally replaced by the *soft margins*. The term "soft" here refers to a relaxation of the margin constraint. We now allow examples to lie below the margin but penalize them linearly via slack variables ψ_n. Consequently, the margin we optimize no longer needs to be the minimum margin of the set. To clearly distinguish between the margin and the soft margin, we call the margin of (2.1) the hard margin. In Figure 2.2, there is one example that is not classified correctly by at least the minimum soft margin, which is shown by the dashed line. In fact, this example is not even on the right side of the hyperplane. If the soft margin is denoted by ρ, then the slack variable $\psi_n = \min(0, \rho - \mathbf{w} \cdot \mathbf{u}_n)$. Thus the slack variable $\psi_n > 0$ only when $\mathbf{w} \cdot \mathbf{u}_n < \rho$.

After adding slack variables ψ_n, the resulting optimization problem (2.2) is again a linear program,

$$\max_{\mathbf{w} \in \mathcal{S}^t,\, \boldsymbol{\psi} \geq 0} \; \min_{n=1,2,\ldots,N} \left(\sum_{q=1}^{t} u_n^q \, w_q + \psi_n \right) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n, \tag{2.2}$$

where the trade-off parameter ν is in the range $[1, N]$. The $\boldsymbol{\psi}$ variables are non-negative because they correspond to Lagrange multipliers for inequality constraints. Adding slack variables in the hypothesis domain (2.2) gives rise to the capping constraints $\mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$ in the Lagrange dual example domain (see e.g. [64, 21] for an early discussion of capping):

$$\min_{\mathbf{d} \in \mathcal{S}^N,\, \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}} \; \max_{q=1,2,\ldots,t} \; \mathbf{u}^q \cdot \mathbf{d}. \tag{2.3}$$

When $\nu = 1$, the $\boldsymbol{\psi}$ variables cancel in (2.2), so the slack variables disappear

and this problem is equivalent to the r.h.s. of (2.1). In addition, the capping constraints in (2.3) are vacuous and this problem is equivalent to the l.h.s. of (2.1). We now reason that when $\nu < 1$, the capping constraints in (2.3) are also vacuous. The objective function of (2.2) can be written as $\sum_{q=1}^{t} u_n^q \, w_q + \left(1 - \frac{1}{\nu}\right) \sum_{n=1}^{N} \psi_n$. When $\nu < 1$, the coefficient $(1 - \frac{1}{\nu})$ is negative, so the objective function is always maximized when $\boldsymbol{\psi} = \mathbf{0}$. Finally, when $\nu > N$, there is no $\mathbf{d} \in \mathcal{S}^N$ that satisfies the constraint $\mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$, so the feasible set is empty.

By linear programming duality, the value at iteration $t = 1, 2, \ldots$ of (2.2) is equal to the value of its dual (2.3), and we will denote this value by P_{LP}^t. The equality is visualized for the hard margin in Figure 2.3. We do not know how to illustrate the soft margin in this way. This equality of (2.2) and (2.3) suggests that boosting algorithms that cap the weight on the examples do well on inseparable data for the same reasons as algorithms that maximize the soft margin.

2.1 Assumption on the Oracle

In each iteration t, the boosting algorithm sends a distribution $\mathbf{d}^{t-1}$ to the oracle and the oracle returns a hypothesis h^t from the base hypothesis set $\mathcal{H}$. The returned hypothesis satisfies a certain quality assumption. The strongest assumption is that the oracle returns a hypothesis with maximum edge, i.e. $h^t \in \operatorname*{argmax}_{h \in \mathcal{H}} \mathbf{u}^h \cdot \mathbf{d}^{t-1}$.

This strong oracle is also assumed for the recently introduced corrective algorithm of [77]. In contrast, in this thesis we follow [66, 91, 87] and only require a lower

bound g on the edge of the hypothesis returned by the oracle. Iteration bounds for this weaker oracle are much harder to obtain.

We assume that given any distribution $\mathbf{d}^{t-1} \in \mathcal{S}^N$ on the examples, the oracle returns a hypothesis h^t with edge at least g. We call g the *guarantee* of the oracle. In other words, if $\mathbf{u}^t$ is the vector that combines the base hypothesis h^t returned by the oracle with the labels of the examples, then the edge $\mathbf{u}^t \cdot \mathbf{d}^{t-1}$ is guaranteed to be at least g.

Now we must consider the range of the guarantee g for which the oracle assumption holds. Because $\mathbf{u}^t \in [-1, +1]^N$, it is easy to achieve $g \geq -1$. We claim that the maximum achievable guarantee is $g = P_{\text{LP}}$, where P_{LP} is defined as the value of (2.3) w.r.t. the entire hypothesis set $\mathcal{H}$ from which oracle can choose:

$$P_{\text{LP}} := \min_{\mathbf{d} \in \mathcal{S}^N} \sup_{\mathbf{d} \leq \frac{1}{\nu}\mathbf{1}} \sup_{h \in \mathcal{H}} \mathbf{u}^h \cdot \mathbf{d}.$$

It follows from the definition of P_{LP} that for any distribution $\mathbf{d}$ such that $\mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$, we have $\max_{h \in \mathcal{H}} \mathbf{u}^h \cdot \mathbf{d} \geq P_{\text{LP}}$. Therefore, for any distribution $\mathbf{d}$ on the examples, there always exists a hypothesis in $\mathcal{H}$ with edge at least P_{LP}. Also, for any optimal distribution that realizes the value P_{LP}, there is no hypothesis of edge strictly greater than P_{LP}.

For computational reasons, the guarantee g of an oracle may be less than P_{LP}, and therefore we formulate our algorithm and iteration bound for an oracle w.r.t. any guarantee $g \in [-1, P_{\text{LP}}]$. It should be emphasized that our algorithm does not need to know the guarantee g achieved by the oracle.

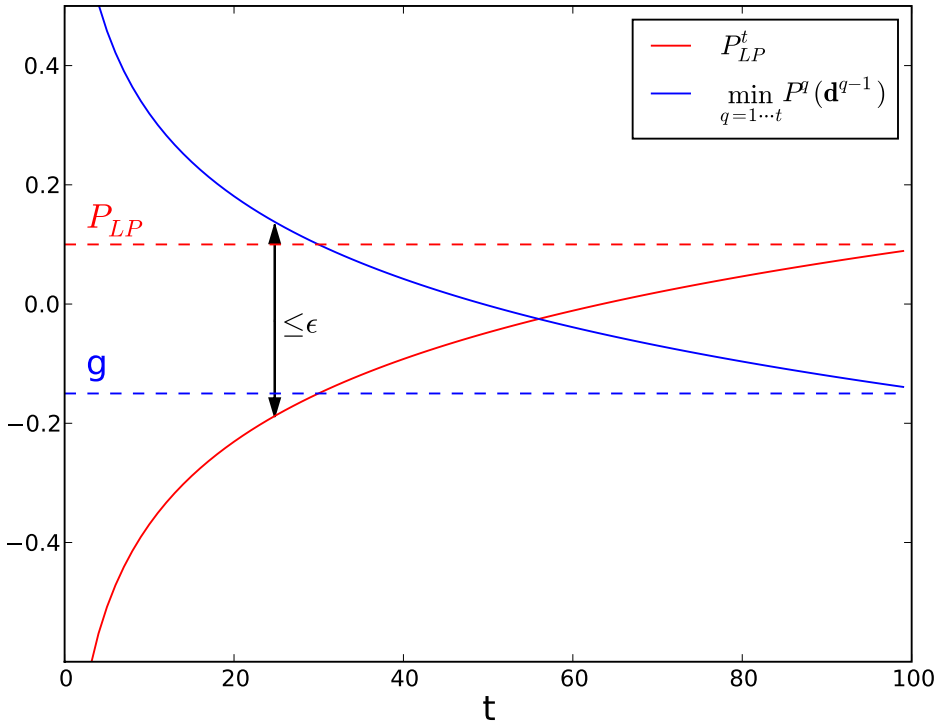


Figure 2.6: Depiction of the LPBoost stopping criterion: $\min_{q=1\ldots T} \mathbf{u}^q \cdot d^{q-1} - P^T_{LP} \leq \epsilon/2$ implies $g - P^T_{\mathrm{LP}} \leq \epsilon$.

2.2 LPBoost

There is a simple linear programming problem that defines a basic boosting algorithm: update $\mathbf{d}^t$, the weights on the examples at iteration t, to any distribution that minimizes the maximum edge of the t hypotheses seen so far. That is, $\mathbf{d}^t \in \operatorname{argmin}_{\mathbf{d}\in\mathcal{S}^N} \max_{q=1\ldots t} \mathbf{u}^q \cdot \mathbf{d}$. This argmin can be solved via linear programming [36, 21]. The resulting boosting algorithm is the hard margin version of LPBoost [36]. Note that the linear program solved by LPBoost typically has multiple solutions. Therefore, the distributions chosen by LPBoost in each iteration are typically not unique and depend

Algorithm 2.1 LPBoost

1. **Input:** $S = \langle (x_1, y_1), \ldots, (x_N, y_N) \rangle$, accuracy parameter $\epsilon > 0$, and smoothness parameter $\nu \in [1, N]$.

2. **Initialize:** $\mathbf{d}^0$ to the uniform distribution and $\widehat{\gamma}_0$ to 1.

3. **Do for** $t = 1, \ldots$

   (a) Send $\mathbf{d}^{t-1}$ to oracle and obtain hypothesis h^t.

   Set $u_n^t = h^t(x_n) y_n$.

   Assume $\mathbf{u}^t \cdot \mathbf{d}^{t-1} \geq g$, where g need not be known.

   (b) Update the distribution to any $[\mathbf{d}^t, P_{\mathrm{LP}}^t] \in \operatorname*{argmin}_{\mathbf{d},\gamma} \gamma$

   s.t. $\mathbf{u}^q \cdot \mathbf{d} \leq \gamma$, for $1 \leq q \leq t$, $d_n \leq 1/\nu$, for $1 \leq n \leq N$, and $\sum_n d_n = 1$.

   (c) **If** $\min_{q=1,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^{q-1} - P_{\mathrm{LP}}^t \leq \epsilon$ **then** set $T = t$ and break.

4. **Output:** $f_{\mathbf{w}}(x) = \sum_{q=1}^{T} w_q h^q(x)$, where the coefficients $\mathbf{w}$ maximize the soft margin over the hypothesis set $\{h^1, \ldots, h^T\}$ using the LP problem (2.2).

$$U = \begin{vmatrix} 0 & 0 & 3 \\ 1 & 1 & 2 \\ 0 & 0 & 3 \end{vmatrix}$$

Figure 2.7: Example of a pivot. In this matrix, the pivot is at position $(2, 3)$ and the pivot value, shown in red, is 2.

on the linear programming solver.

LPBoost begins by setting $\mathbf{d}^0$ to the uniform distribution. In iteration t, the LPBoost algorithm [21] sends its current distribution $\mathbf{d}^{t-1}$ to the oracle and receives a hypothesis h^t that satisfies $\mathbf{d}^{t-1} \cdot \mathbf{u}^t \geq g$. It then updates its distribution to $\mathbf{d}^t$ by solving the soft margin linear programming problem defined by (2.3) based on the t hypotheses received so far.

The goal of the boosting algorithms is to produce a convex combination of T hypotheses such that $P_{LP}^T \geq g - \epsilon$. Because we cannot assume that g is known, LPBoost monitors the quantity $\min_{q=1\ldots t} \mathbf{u}^q \cdot \mathbf{d}^{q-1} - P_{LP}^t$ instead, and terminates when this quantity is less than ϵ. By the assumption on the oracle, the sequence $\min_{q=1\ldots t} \mathbf{u}^q \cdot \mathbf{d}^{q-1}$ is non-increasing and it is lower-bounded by g. Similarly, the sequence P_{LP}^t is non-decreasing and is upper-bounded by P_{LP}. As a result, $\min_{q=1\ldots t} \mathbf{u}^q \cdot \mathbf{d}^{q-1} - P_{LP}^t \leq \epsilon$ ensures that $P_{LP}^t \geq g - \epsilon$. This stopping criterion is visualized by Figure 2.6.

2.3 Pivots and Linear Programming

In this section, we establish an elementary result about a special element in a matrix defining the $\min - \max$ problem in (2.1). This result will be applied in the analysis of Chapter 2.4 to show that the iteration bound of LPBoost must be at least

linear in the number of examples. This result is also used in Chapter 2.5 to show that any linearly separable dataset can be reduced to a dataset on which LPBoost misclassifies all examples.

Let $\mathbf{e}_i$ be a unit vector with a one in position i. Also, let U be the matrix whose columns are the $\mathbf{u}^q$ vectors and let $U_{i,j}$ refer to the element of U at row i and column j. Note that with this notation, $U_{i,j} = u_i^j$.

Definition 2.3 *Location (i, j) is a **pivot** of U if the element $U_{i,j}$ is greater than or equal to any element in its row and less than or equal to any element in its column.*

An example of a pivot can be seen in Figure 2.7. In this matrix, the pivot is at position $(2, 3)$ and the pivot value, shown in red, is 2.

Lemma 2.4 *If U contains a pivot at position (i, j), then the value of the optimization problem defined by (2.1) is the pivot value $U_{i,j}$ and $\mathbf{e}_i$, $\mathbf{e}_j$ are optimal solutions for $\mathbf{d}$ and $\mathbf{w}$ respectively [88].*

Proof Let P_{LP}^t be the value of the optimization problem defined by (2.1). Since the l.h.s. of (2.1) is a min over $\mathbf{d}$, we have that for any $\mathbf{d} \in \mathcal{S}^N$

$$P_{LP}^t \leq \max_{q=1\ldots t} \mathbf{u}^q \cdot \mathbf{d}. \tag{2.4}$$

For $\mathbf{d} = \mathbf{e}_i$, we get

$$P_{LP}^t \leq \max_{q=1\ldots t} u_i^q = u_i^j.$$

Since the r.h.s. of (2.1) is a max over $\mathbf{w}$, the following holds for any $\mathbf{w}$:

$$P_{LP}^t \geq \min_{n=1\ldots N} \sum_{q=1}^{t} u_n^t w_q. \tag{2.5}$$

$n \setminus t$	1	2	3	4	5
1	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
2	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
3	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
4	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
5	$\mathbf{-1+2\delta}$	$+1$	$-1+7\delta$	$-1+9\delta$	$+1-\delta$
6	$-1+3\delta$	$\mathbf{-1+4\delta}$	$+1$	$-1+9\delta$	$+1-\delta$
7	$-1+3\delta$	$-1+5\delta$	$\mathbf{-1+6\delta}$	$+1$	$+1-\delta$
8	$-1+3\delta$	$-1+5\delta$	$-1+7\delta$	$\mathbf{-1+8\delta}$	$+1-\delta$
P_{LP}^t	$\mathbf{-1+2\delta}$	$\mathbf{-1+4\delta}$	$\mathbf{-1+6\delta}$	$\mathbf{-1+8\delta}$	$\geq \delta/2$

Figure 2.8: An example where LPBoost provably requires $\Omega(N/2)$ iterations from [87]. These are the $\mathbf{u}^t$ vectors that are hard for LPBoost (for $\nu = 1$).

For $w = \mathbf{e}_j$, we get

$$P_{LP}^t \geq \min_{n=1\ldots N} u_n^j = u_i^j.$$

We conclude that $\mathbf{d} = \mathbf{e}_i$ makes (2.4) tight and $w = \mathbf{e}_j$ makes (2.5) tight. ■

2.4 A Lower Bound on the LPBoost Iteration Bound

To our knowledge, there is no known iteration bound for LPBoost even though it provably converges to within ϵ of the optimal solution of the optimization problem [21, 39]. Empirically, the number of iterations required for convergence depends on the linear programming optimizer, e.g. simplex or interior point solver [91]. In [87] Warmuth et al. established for the first time a lower bound showing that, independent of the optimizer, LPBoost can require $\Omega(N)$ iterations:

Theorem 2.5 *There exists a case where LPBoost requires $N/2$ iterations to achieve a hard margin that is within* 0.99 *of the optimum hard margin.*

Proof Assume we are in the hard margin case ($\nu = 1$). The counterexample has N examples and $\frac{N}{2}+1$ base hypotheses. After $\frac{N}{2}$ iterations, the optimal value P_{LP}^t for the chosen hypotheses will still be close to -1, whereas after the last hypothesis is added, this value is at least $\delta/2$. Here $\delta > 0$ is an arbitrary small number.

Figure 2.8 shows the case where $N = 8$ and $T = 5$, but it is trivial to generalize this example to any even N. There are 8 examples/rows and the five columns are the $\mathbf{u}$ vectors of the five available base hypotheses. Recall that the $\mathbf{u}$ vectors combine the labels and the examples. The examples are linearly separable because if we choose the weight on the hypotheses $\mathbf{w}$ to put half of the weight on the first hypothesis and half of the weight on the last hypothesis, then the margins of all examples are at least $\delta/2$.

We assume that in each iteration the oracle will return the remaining hypothesis with maximum edge. This will result in LPBoost choosing the hypotheses in order, and there will never be any ties. The initial distribution $\mathbf{d}^0$ is uniform. At the end of iteration t ($1 \leq t \leq N/2$), the bolded entry of column t in Figure 2.8 is a pivot of the matrix U whose columns are $\mathbf{u}_1 \ldots \mathbf{u}_t$. By Lemma 2.4, the distribution $\mathbf{d}^t$ will focus all its weight on example $N/2+t$, and the optimal distribution $\mathbf{w}^t$ will put all of its weight on the t^{th} hypothesis that was just received. The value of the optimization problem in (2.1) will be $-1+2\delta t$ at the end of iteration $t = 1, \ldots, N/2$. After $N/2$ iterations, the value P_{LP}^t of the underlying LP problem will still be close to -1, because δ can be made arbitrarily small. We reasoned already that the value for all $N/2+1$ hypotheses will be positive. If δ is sufficiently small, then after $N/2$ iterations LPBoost is still at least .99 away from the optimal solution. ■

Although the example set used in the above proof is linearly separable, we can modify it explicitly to argue that capping the distribution on examples will not help in the sense that "soft" LPBoost with $\nu > 1$ can still have linear iteration bounds. To negate the effect of capping, simply pad out the problem by duplicating all of the rows ν times. There will now be $\tilde{N} = N\nu$ examples, and after $\frac{N}{2} = \frac{\tilde{N}}{2\nu}$ iterations, the value of the game is still close to -1. This is not a claim that capping has no value. It remains an important technique for making an algorithm more robust to noise. However, it is not sufficient to improve the iteration bound of LPBoost from linear growth in N to logarithmic.

2.5 The Master Hypothesis Returned by LPBoost

Consider the case where LPBoost has terminated and we want to use the optimization problem defined by (2.1) to find a master hypothesis that is a convex combination of the T selected hypotheses: $f_{\mathbf{w}}(x_n) = \sum_{q=1}^{T} w_q h^q(x_n)$. We seek to determine whether this master hypothesis is a good one. Specifically, if there exists a convex combination of selected hypotheses that can correctly classify all but one example, then the master hypothesis should not have significantly worse performance. In this section we show that when the master hypothesis is computed via the optimization problem defined by (2.1), it is possible to reduce a linearly separable set of examples to one where LPBoost misclassifies all examples by adding one bad example and one bad hypothesis.

$n \setminus t$	1	2	3	4	5
1	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
2	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
3	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
4	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$
5	$-1+2\delta$	$+1$	$-1+7\delta$	$-1+9\delta$	$+1-\delta$
6	$-1+3\delta$	$-1+4\delta$	$+1$	$-1+9\delta$	$+1-\delta$
7	$-1+3\delta$	$-1+5\delta$	$-1+6\delta$	$+1$	$+1-\delta$
8	$-1+3\delta$	$-1+5\delta$	$-1+7\delta$	$-1+8\delta$	$+1-\delta$
9	$\boldsymbol{-3\delta}$	$\boldsymbol{-3\delta}$	$\boldsymbol{-3\delta}$	$\boldsymbol{-3\delta}$	$\boldsymbol{-3\delta}$
P^t_{LP}	-3δ	-3δ	-3δ	-3δ	-3δ

(a) Adding a bad example to the counterexample in Figure 2.8.

$n \setminus t$	1	2	3	4	5	6
1	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$	$\boldsymbol{-\delta}$
2	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$	$\boldsymbol{-\delta}$
3	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$	$\boldsymbol{-\delta}$
4	$+1$	$-1+5\delta$	$-1+7\delta$	$-1+9\delta$	$-1+\delta$	$\boldsymbol{-\delta}$
5	$-1+2\delta$	$+1$	$-1+7\delta$	$-1+9\delta$	$+1-\delta$	$\boldsymbol{-\delta}$
6	$-1+3\delta$	$-1+4\delta$	$+1$	$-1+9\delta$	$+1-\delta$	$\boldsymbol{-\delta}$
7	$-1+3\delta$	$-1+5\delta$	$-1+6\delta$	$+1$	$+1-\delta$	$\boldsymbol{-\delta}$
8	$-1+3\delta$	$-1+5\delta$	$-1+7\delta$	$-1+8\delta$	$+1-\delta$	$\boldsymbol{-\delta}$
9	-3δ	-3δ	-3δ	-3δ	-3δ	$\boldsymbol{-2\delta}$
P^t_{LP}	-3δ	-3δ	-3δ	-3δ	-3δ	-2δ

(b) Adding a bad hypothesis to the counterexample in Figure 2.8.

Figure 2.9: Even when LPBoost is given a good set of hypotheses, it can return a very bad final hypothesis. Note that these are the $\mathbf{u}^t$ vectors, so they incorporate both the examples and the labels.

First we describe this reduction using a concrete example. Suppose that there is a set of examples that can all be correctly classified by a convex combination of hypotheses. The counterexample from the previous section (Figure 2.8) satisfies this condition by putting equal weight on the first and last hypotheses. In Figure 2.9(a), we add a row of -3δ to the matrix in Figure 2.8. We then add hypothesis that misclassifies every example. This corresponds to the last column in Figure 2.9(b). Since position $(9, 6)$ in this matrix is a pivot, Lemma 2.4 tells us that the value of the optimization problem in (2.1) is -2δ and that $\mathbf{w}$ puts all of its weight on the last hypothesis. Therefore, LPBoost misclassifies all of the examples.

Now we show how the same reduction applies to any linearly separable dataset. Suppose we have a set of labeled examples (x_n, y_n) for $n = 1 \dots N-1$ and a set of hypotheses $h^1 \dots h^{T-1}$ such that the examples are linearly separable. Fix $0 < \delta \ll 1$. First we add a bad example x_N, such that $u_N^q = -3\delta$ for $q = 1 \dots t-1$. We then construct hypothesis T such that $u_n^T = -\delta$ for $n = 1 \dots N-1$ and $\mathbf{u}_N^T = -2\delta$. By construction, the value corresponding to example N and hypothesis T is a pivot, and by Lemma 2.4, the $\mathbf{w}$ found by LPBoost will put all of its weight on hypothesis T. Therefore, LPBoost will misclassify every example.

In summary, we can reduce any linearly separable set of examples to one where LPBoost misclassifies all examples simply by adding one bad example and one bad hypothesis. This reduction is independent of the size of the dataset and the number of hypotheses. Also, the reduction does not rely on the scale of the noisy example. As long as the values in the bad example are negative, they can be arbitrarily close to zero.

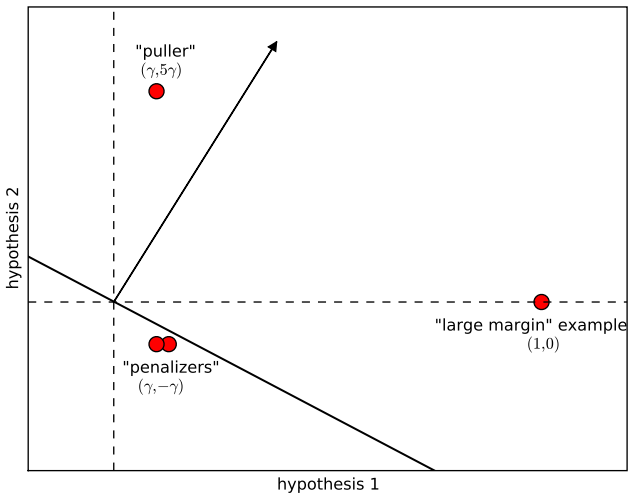


Figure 2.10: Long and Servedio's counterexample demonstrates no convex potential booster is robust to random classification noise.

2.6 Robustness to Random Label Noise

Boosting algorithms can be viewed as the minimization of some function of the margins of the examples via coordinate descent [49, 34, 63, 8, 26]. This function of the margin, called a *potential function*, is denoted by ϕ and must satisfy the following properties:

1. ϕ is non-increasing, continuous, and differentiable.

2. $\phi'(0) < 0$ and $\lim_{x \to +\infty} \phi(x) = 0$.

The *convex potential booster*[1] as defined in this paper minimizes a global potential

$$\Phi(\mathbf{w}) := \sum_{n=1}^{N} \phi(y_n \sum_{q=1}^{t} w_q h^t(x_n)).$$

[1]It should be noted that not all convex potential functions result in boosting algorithms [25].

Long and Servedio [49] showed by counterexample that no convex potential booster can be provably robust to random classification noise. Random classification noise means that with some fixed probability, the label of an example is corrupted. The counterexample constructed by Long and Servedio is shown in Figure 2.10. It consists of four examples and two hypotheses. All of the examples have label 1. It is easy to see that hypothesis 1 correctly classifies all examples with margin at least γ. The examples are arranged such that if the label of the example with large margin is corrupted, its scale can overwhelm the other examples and will force the booster to put too much weight on hypothesis 2. The main result in [49] is that for any convex potential function and any noise rate in the range $(0, 1/2)$, there exists a $0 < \gamma < 1/6$ such that the optimal $\mathbf{w}$ misclassifies two of the four examples in Figure 2.10.

Neither LPBoost nor ERLPBoost (described in Chapter 3) satisfy the preconditions of Long and Servedio's theorem. First of all, LPBoost is not a convex potential booster. To see why, recall that LPBoost optimizes $\max_{\mathbf{w}} \min_{n=1\ldots N} y_n \sum_{q=1}^{t} w_q h^t(x_n)$, which is equivalent to minimizing

$$\min_{n=1\ldots N} -y_n \sum_{q=1}^{t} w_q h^t(x_n).$$

Because there is a min instead of a sum, LPBoost does not optimize a function of the form of $\Phi(\mathbf{w})$. In addition, the potential function for LPBoost would be $\phi(x) = -x$, which does not satisfy property 2 because $\lim_{x\to+\infty} \phi(x) = -\infty$. In the $\mathbf{w}$ domain, ERLPBoost optimizes

$$-\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \psi_n)) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n.$$

This does not fit the definition of a convex potential booster, but in the uncapped case ($\nu = 1$), the ψ_n variables vanish, and the function optimized by ERLPBoost becomes

$$-\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q).$$

Since the logarithm is monotonic, it can be dropped from the objective function, and the resulting function fits the definition of a convex potential booster. However, even in the uncapped case, Long and Servedio's theorem does not apply to ERLPBoost. The reason is that the form of the $\mathbf{w}$ vector found in their analysis requires $w_1 + w_2 > 1$, but our algorithms require $w_1 + w_2 = 1$.

Nevertheless, we show experimentally that both LPBoost and ERLPBoost exhibit the same problem as the convex potential boosters that satisfy the theoretical requirements of the Long and Servedio's analysis. Recall that the main result in [49] is that for any convex potential function there exists a γ such that the optimal $\mathbf{w}$ misclassifies two of the four examples in Figure 2.10. Thus, to show that an algorithm is vulnerable to this counter example, it suffices to find a γ with this property. When we ran LPBoost and ERLPBoost on Long and Servedio's counterexample with $\gamma = 0.1$, half of the examples were misclassified. Furthermore, no value of the capping parameter ν or regularization parameter η was enough make the algorithms more robust. In the end, the large margin of the bad example is enough to force both algorithms to put too much weight on hypothesis 2 (the bad hypothesis) and not enough on hypothesis 1 (the good hypothesis).

Long and Servedio's example is significant because it shows that convex potential boosters cannot be provably robust to random label noise. However, because it relies on extreme margin imbalances and a small sample size, it is not an argument that convex potential boosters perform badly in practice.

The reduction in the previous section differs from Long and Servedio's example in several important ways. First, it is not intended to argue against all convex potential boosters – it was only intended to argue against LPBoost. Second, the reduction relies on pivots to uniquely determine the optimal linear programming solution, not on extreme margin imbalances. In the reduction, the one bad example can have values that are arbitrarily close to zero. Third, the reduction only requires the original data set to be linearly separable, but it makes no assumption about the structure of the data set or its size.

Chapter 3

Entropy-Based Boosting Algorithms

In this chapter, we present three algorithms: SoftBoost, Entropy Regularized LPBoost (ERLPBoost), and Binary Entropy Regularized LPBoost (Binary ERLPBoost). Boosting algorithms maintain a distribution $\mathbf{d}$ on the examples, and this distribution is used to select the next hypothesis. These algorithms employ either the relative entropy or the binary relative entropy to distribute the weights on the examples more uniformly, thereby avoiding the problem caused by concentrating too much weight on a few examples, as LPBoost does (see discussion in Chapter 2). SoftBoost minimizes the entropy subject to an increasing number of constraints, while the entropies are used as regularizers in ERLPBoost and Binary ERLPBoost.

In Chapter 2.4, we showed that if LPBoost has iteration bounds, they are at least linear in the number of examples. In contrast, for the algorithms in this chapter we will prove iteration bounds that are logarithmic in the number of examples. More specifically, all three algorithms will come ϵ-close to the optimal linear programming

Symbol	*Description*
N	Number of examples.
t	Iteration number.
T	Final number of iterations.
$\mathcal{S}^M$	Probability simplex of dimension M.
$\mathbf{x}_n$	n^{th} example.
$\mathcal{X}$	Domain of examples.
$y_n \in \{-1, 1\}$	n^{th} label.
$h^t(x_n) \in [-1, 1]$	Prediction of hypothesis t on example x_n.
$\mathcal{H}$	Hypothesis class.
$u_n^t = y_n h^t(x_n)$	Convenient notation for combining labels and hypotheses.
$\mathbf{d} \in \mathcal{S}^N$	Distribution on examples.
γ	maximum edge w.r.t. distribution $\mathbf{d}$
$\mathbf{w} \in \mathcal{S}^t$	Distribution on hypotheses at iteration t.
$\boldsymbol{\psi} \in [0, \infty]^N$	Slack variable for soft margin problem.
β	Lagrange multiplier for $\sum_{n=1}^N d_n = 1$ constraint.
ϵ	Precision parameter.
ν	Capping parameter.
g	Guarantee of the oracle.
P_{LP}^t	Value of the LPBoost objective function at iteration t.
P_{LP}	Value of the LPBoost objective function over *all* hypotheses.
$P_S(\mathbf{d})$	Value of the SoftBoost objective function
$\widehat{\Theta}_S^t(\mathbf{w}, \boldsymbol{\psi})$	Lagrangian dual of P_s
$P(\mathbf{d})$	Value of the ERLPBoost objective function
$\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$	Lagrangian dual of P
$P_U(\mathbf{d})$	Value of the Unnormalized ERLPBoost objective function
$\widehat{\Theta}_U$	Lagrangian dual of P_U
$P_B(\mathbf{d})$	Value of the Binary ERLPBoost objective function
$\widehat{\Theta}_B^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$	Lagrangian dual of P_B
$\mathcal{A}_{\mathbf{w}}$	Active set for $\mathbf{u} \cdot \mathbf{d} \leq \gamma$ constraints
$\mathcal{A}_{\boldsymbol{\psi}}$	Active set for $d_n \leq 1/\nu$ constraints
δ^t	simple stopping criterion for ERLPBoost
δ_B^t	simple stopping criterion for Binary ERLPBoost
$\widetilde{\delta}^t$	practical stopping criterion for Binary ERLPBoost
$\widetilde{\delta}_B^t$	practical stopping criterion for Binary ERLPBoost

Table 3.1: Notation for entropy-based boosting algorithms

solution in at most $O\left(\frac{\ln\left(\frac{N}{\nu}\right)}{\epsilon^2}\right)$ iteration, where N is the number of examples and ν is the capping parameter that constrains the weight on each example to be at most $\frac{1}{\nu}$. The range of ν is $[1, N]$ and the iteration bound tightens as ν increases. When $\nu = N$, the bound is zero because the only feasible distribution on the examples is the uniform distribution.

There are important similarities between LPBoost and the three algorithms presented in this chapter. Like LPBoost, these algorithms all maximize the soft margin. Also, these algorithms employ capping to make them robust to noise. Finally, these algorithms are totally corrective: the updated distribution $\mathbf{d}$ on the examples has small edge w.r.t. *all* of the past hypotheses.

The chapter is organized as follows. We will discuss SoftBoost, ERLPBoost, and Binary ERLPBoost. For each algorithm, we derive the stopping criterion, the Lagrangian dual, the system of linear equations that relate the primal and dual variables, and the iteration bound.

3.1 The SoftBoost Algorithm

The SoftBoost algorithm [87] is motivated by minimizing the relative entropy to the initial distribution subject to specially chosen constraints. The relative entropy between distribution $\mathbf{d}$ and the uniform distribution $\mathbf{d}^0$ is defined as $\Delta(\mathbf{d}, \mathbf{d}^0) := \sum_n d_n \ln \frac{d_n}{d_n^0}$. The relative entropy is minimized when $\mathbf{d} = \mathbf{d}^0$, so the optimization problem solved by SoftBoost picks the feasible distribution that is closest

Algorithm 3.1 SoftBoost with accuracy param. ϵ and capping parameter ν

1. **Input:** $S = \langle (x_1, y_1), \ldots, (x_N, y_N) \rangle$, desired accuracy ϵ, and capping parameter $\nu \in [1, N]$.

2. **Initialize:** $\mathbf{d}^0$ to the uniform distribution and $\widehat{\gamma}_0$ to 1.

3. **Do for** $t = 1, \ldots$

   (a) Send $\mathbf{d}^{t-1}$ to the oracle and obtain hypothesis h^t.

   Set $u_n^t = h^t(x_n) y_n$ and $\widehat{\gamma}_t = \min\{\widehat{\gamma}_{t-1}, \mathbf{u}^t \cdot \mathbf{d}^{t-1}\}$.

   (Assume $\mathbf{d}^{t-1} \cdot \mathbf{u}^t \geq g$, where edge guarantee g is unknown.)

   (b) Update[3]

   $$\mathbf{d}^t = \operatorname*{argmin}_{\mathbf{d}} \Delta(\mathbf{d}, \mathbf{d}^0), \quad \text{s.t. } \mathbf{u}^q \cdot \mathbf{d} \leq \widehat{\gamma}_t - \epsilon, \text{ for } 1 \leq q \leq t, \sum_n d_n = 1, \ \mathbf{d} \leq \frac{1}{\nu} \mathbf{1}.$$

   (c) **If** above infeasible or $\mathbf{d}^t$ contains a zero **then** $T = t$ and terminate.

4. **Output:** $f_{\mathbf{w}}(x) = \sum_{q=1}^{T} \mathbf{w}_q^T h^q(x)$, where the coefficients $\mathbf{w}_q^T$ are the dual variables of the above optimization problem at iteration T .

[3] When g is known, replace the upper bound $\widehat{\gamma}_t - \epsilon$ by $g - \epsilon$.

to uniform. In this way, the relative entropy mechanism prevents $\mathbf{d}$ from becoming overly concentrated on a few examples.

SoftBoost takes as input a sequence of examples $S = \langle (x_1, y_1), \ldots, (x_N, y_N) \rangle$, an accuracy parameter ϵ, and a capping parameter ν. The algorithm has an oracle available with unknown guarantee g. Its initial distribution $\mathbf{d}^0$ is uniform. In each iteration t, the algorithm prompts the oracle for a new base hypothesis, incorporates it into the constraint set, and updates its distribution $\mathbf{d}^{t-1}$ to $\mathbf{d}^t$ by minimizing the relative entropy subject to linear constraints:

$$\begin{aligned} \min_{\mathbf{d}} \quad & \Delta(\mathbf{d}, \mathbf{d}^0) \\ \text{s.t.} \quad & \mathbf{u}^q \cdot \mathbf{d} \leq \widehat{\gamma}_t - \epsilon, \text{ for } 1 \leq q \leq t, \\ & \textstyle\sum_n d_n = 1, \ \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}. \end{aligned} \tag{3.1}$$

We define $\widehat{\gamma}_t := \min_{1 \leq q \leq t} \mathbf{u}^q \cdot \mathbf{d}^{q-1}$. Because the set over which the minimum is taken gets larger with each iteration, $\widehat{\gamma}_t$ is non-increasing. Note that decreasing $\widehat{\gamma}_t$ amounts to tightening the constraints of the optimization problem. At the same time, a new constraint is added to the optimization problem at each iteration. This further reduces the size of the feasible set. SoftBoost terminates when the above optimization problem is infeasible or when $\mathbf{d}^t$ contains a zero.

Observe that removing the relative entropy term from the objective results in a feasibility problem for a linear program where the edges are upper bounded by $\widehat{\gamma}_t - \epsilon$. If we remove the relative entropy and minimize the upper bound on the edges, then we arrive at the optimization problem of LPBoost. In this case, logarithmic growth in the number of examples is no longer possible. The relative entropy in the objective ensures

that the probabilities of the examples are always proportional to their exponentiated negative soft margins (3.2). That is, more weight is put on the examples with low soft margin, which are the examples that are hard to classify. In contrast, recall from Chapter 2 that LPBoost puts all of the weight on examples of minimum soft margin.

SoftBoost is the same as TotalBoost except for the additional capping constraints. SoftBoost with $\nu = 1$ makes the capping constraints vacuous and recovers TotalBoost.

3.1.1 Lagrangian Dual of SoftBoost

In the following lemma, we compute the Lagrangian dual of the SoftBoost optimization problem. We make use of the Lagrangian dual in the iteration bound, but there is also some insight to be gained from the dual problem. First, as a byproduct of deriving the Lagrangian dual, we get a closed-form expression for the update on the $\mathbf{d}$ variables in terms of the dual variables $\mathbf{w}$ and $\boldsymbol{\psi}$. The second point of interest is the form of the Lagrangian dual itself. The corresponding dual for LPBoost tries to maximize the minimum soft margin. In contrast, the dual of the SoftBoost problem is a soft max.

Lemma 3.1 *The Lagrangian dual of* (3.1) *is*

$$\max_{\mathbf{w},\boldsymbol{\psi}} \ \widehat{\Theta}_S^t(\mathbf{w},\boldsymbol{\psi}), \quad \textit{s.t.}\ \mathbf{w} \geq \mathbf{0},\ \mathbf{w}\cdot\mathbf{1} = 1, \boldsymbol{\psi} \geq 0,$$

$$\textit{where}\ \widehat{\Theta}_S^t(\mathbf{w},\boldsymbol{\psi}) := -\ln \sum_{n=1}^{N} d_n^0 \exp(-\sum_{q=1}^{t} u_n^q w_q - \psi_n) - (\widehat{\gamma}_t - \epsilon)\sum_{q=1}^{t} w_q - \frac{1}{\nu}\sum_{n=1}^{N}\psi_n.$$

The optimal solution $\mathbf{d}^t$ *of* (3.6) *can be expressed in terms of the dual variables* $\mathbf{w}^t$ *and* $\boldsymbol{\psi}^t$ *as follows:*

$$d_n^t := \frac{d_n^0 \exp(-\sum_{q=1}^t u_n^q w_q^t - \psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\sum_{q=1}^t u_{n'}^q w_q^t - \psi_n^t))}. \tag{3.2}$$

Proof The Lagrangian of the minimization problem in (3.1) is

$$\begin{aligned} L^t(\underbrace{\mathbf{d}}_{\text{primal}}, \underbrace{\mathbf{w}, \boldsymbol{\psi}, \beta}_{\text{dual}}) = \Delta(\mathbf{d}, \mathbf{d}^0) + \sum_{q=1}^t w_q(\mathbf{u}^q \cdot \mathbf{d} - \widehat{\gamma}_t + \epsilon) \\ + \sum_{n=1}^N \psi_n(d_n - 1/\nu) + \beta(\mathbf{1} \cdot \mathbf{d} - 1). \end{aligned} \tag{3.3}$$

The dual is derived from the Lagrangian by plugging in the minimum value of the primal variables:

$$\Theta_S^t(\mathbf{w}, \boldsymbol{\psi}, \beta) := \inf_{\mathbf{d}} L^t(\mathbf{d}, \mathbf{w}, \boldsymbol{\psi}, \beta).$$

Differentiating w.r.t. $\mathbf{d}$ shows that the n-th component of the optimal $\mathbf{d}$ vector has the form

$$d^n = d_n^0 \exp(-\sum_{q=1}^t u_n^q w_q - \psi_n - \beta - 1). \tag{3.4}$$

By plugging the optimal $\mathbf{d}$ into the Lagrangian shown in (3.3), the dual function simplifies to

$$\Theta_S^t(\mathbf{w}, \boldsymbol{\psi}, \beta) = -\sum_{n=1}^N d_n^0 \exp(-\sum_{q=1}^t u_n^q w_q - \psi_n - \beta - 1) - \beta - (\widehat{\gamma}_t - \epsilon)\sum_{q=1}^t w_q - \frac{1}{\nu}\sum_{n=1}^N \psi_n.$$

This results in the following Lagrange dual:

$$\begin{aligned} &\max_{\mathbf{w}} \ \Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta) \\ &\text{s.t. } \mathbf{w} \geq \mathbf{0}, \ \boldsymbol{\psi} \geq 0, \end{aligned}$$

By differentiating $\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$, we can determine the optimal choice of β:

$$\beta^t(\mathbf{w}, \boldsymbol{\psi}) = -1 + \ln \sum_{n=1}^{N} d_n^0 \exp(-\sum_{q=1}^{t} u_n^q w_q - \psi_n).$$

Plugging this choice for β into (3.4) results in

$$\begin{aligned} d_n^t :=& d_n^t(\mathbf{w}^t, \boldsymbol{\psi}^t, \beta^t(\mathbf{w}^t, \boldsymbol{\psi}^t)) \\ =& \frac{d_n^0 \exp(-\sum_{q=1}^{t} u_n^q w_q^t - \psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\sum_{q=1}^{t} u_{n'}^q w_q^t - \psi_n^t)}. \end{aligned}$$

Once β is optimized, the Lagrangian becomes

$$\begin{aligned} &\Theta_S^t(\mathbf{w}, \boldsymbol{\psi}, \beta^t(\mathbf{w}, \boldsymbol{\psi})) = \widehat{\Theta}_S^t(\mathbf{w}, \boldsymbol{\psi}) \\ &:= -\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q - \eta \psi_n) - (\widehat{\gamma}_t - \epsilon) \sum_{q=1}^{t} w_q - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n. \end{aligned}$$

The dual problem now reduces to

$$\begin{aligned} \max_{\mathbf{w}, \boldsymbol{\psi}} \quad & \widehat{\Theta}_S^t(\mathbf{w}, \boldsymbol{\psi}) \\ \text{s.t.} \quad & \mathbf{w} \geq \mathbf{0},\ \boldsymbol{\psi} \geq 0. \end{aligned}$$

The primal objective is convex and the primal constraints are affine. Also, the uniform distribution is always a feasible solution. Therefore, Slater's condition [7] tells us that since this problem has a non-empty feasible set, strong duality holds and the values of the primal and the dual problems are the same. ■

3.1.2 Relationship Between the Primal and Dual Variables of SoftBoost

In the process of finding the Lagrangian dual of the SoftBoost problem, we were able to express the optimal $\mathbf{d}^t$ in terms of the dual variables $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$:

$$d_n^t = \frac{d_n^0 \exp(-\sum_{q=1}^t u_n^q w_q^t - \psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\sum_{q=1}^t u_{n'}^q w_q^t - \psi_n^t)}.$$

We now want to compute $\mathbf{w}^t$ and ψ^t given $\mathbf{d}^t$. It is possible to do so via the Karush-Kuhn-Tucker optimality conditions, which relate the primal and dual optimal solutions [7]. Let $\mathbf{e}_n$ be a unit vector with a 1 in the n^{th} component. Then for the SoftBoost algorithm, the KKT conditions are:

$$\mathbf{u}^q \cdot \mathbf{d}^t - \widehat{\gamma}_t + \epsilon \leq 0, \quad q = 1 \dots t \qquad\qquad d_n^t \leq \tfrac{1}{\nu}, \quad n = 1 \dots N$$

$$w_q^t \geq 0, \quad q = 1 \dots t \qquad\qquad \psi_n^t \geq 0, \quad n = 1 \dots N$$

$$w_q^t(\mathbf{u}^q \cdot \mathbf{d}^t - \widehat{\gamma}_t + \epsilon) = 0, \quad q = 1 \dots t \qquad\qquad \psi_n^t(d_n^t - \tfrac{1}{\nu}) = 0, \quad n = 1 \dots N$$

$$\sum_n d_n^t - 1 = 0, \quad n = 1 \dots N$$

$$\tfrac{1}{\eta} + \tfrac{1}{\eta} \ln\left(\frac{\mathbf{d}^t}{\mathbf{d}^0}\right) + \sum_{q=1}^t w_q^t \mathbf{u}^q + \sum_{n=1}^N \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0.$$

The inequality constraints make this a complicated set of equations to solve, but the problem can be considerably simplified by observing that $\mathbf{d}^t$, the optimal value of $\mathbf{d}$, is already known. Knowing $\mathbf{d}^t$ means we also know which inequality constraints are active and which are slack. Following the discussion in Appendix A, we can make use of this knowledge to find $\mathbf{w}$ and $\boldsymbol{\psi}$ by solving an equivalent simpler KKT system.

For a given $\mathbf{d}$, we define the sets of active constraints for $\mathbf{w}$ as $\mathcal{A}_{\mathbf{w}}(\mathbf{d}) := \{q \in 1 \dots t : \mathbf{u}^q \cdot \mathbf{d} = \widehat{\gamma}_t - \epsilon\}$ and $\boldsymbol{\psi}$ as $\mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}) := \{n \in 1 \dots N : d_n = \frac{1}{\nu}\}$. If $\mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)$ and

$\mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)$ are the optimal active sets, then (3.1) can be equivalently expressed as

$$\begin{aligned} \min_{\mathbf{d}} \quad & \Delta(\mathbf{d}, \mathbf{d}^0) \\ \text{s.t.} \quad & \mathbf{u}^q \cdot \mathbf{d} = \widehat{\gamma}_t - \epsilon \text{ for } q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t), \\ & \textstyle\sum_n d_n = 1,\ d_n = \frac{1}{\nu} \text{ for } n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t). \end{aligned}$$

Then the KKT optimality conditions for this problem are

$$\begin{aligned} & \textstyle\sum_n d_n^t = 1 \\ & d_n^t = \tfrac{1}{\nu} \text{ for } n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t) \\ & \mathbf{d}^t \cdot \mathbf{u}^q = \widehat{\gamma}_t - \epsilon \text{ for } q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t) \\ & \tfrac{1}{\eta} + \tfrac{1}{\eta} \ln\left(\tfrac{\mathbf{d}^t}{\mathbf{d}^0}\right) + \textstyle\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)} w_q \mathbf{u}^q + \sum_{n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)} \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0. \end{aligned}$$

Recall that in this scenario, we are given $\mathbf{d}^t$ and want to use this knowledge to get $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$. Because the first three of the above equations are simply solving for $\mathbf{d}^t$ and $\mathbf{d}^t$ is already known, we can ignore them. The last equation is what relates $\mathbf{d}^t$ to the dual variables. We can find $\mathbf{w}^t$, $\boldsymbol{\psi}^t$, and β^t by solving the linear system

$$\frac{1}{\eta} + \frac{1}{\eta} \ln\left(\frac{\mathbf{d}^t}{\mathbf{d}^0}\right) + \sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)} w_q \mathbf{u}^q + \sum_{n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)} \psi_n \mathbf{e}_n + \beta \mathbf{1} = 0.$$

The SoftBoost optimization problem is strictly convex and its feasible set is a closed convex set, so as long as the optimization problem is feasible, then it has a unique optimal solution [7]. As a result, the system of equations defined by the KKT conditions has an exact solution.

3.1.3 Iteration Bound for SoftBoost

In this section, we prove an upper bound on the number of iterations required by SoftBoost to achieve a soft margin that is within ϵ of guarantee g. This result

originally appeared in [87]. The iteration bound for SoftBoost is very similar to the bound proved for TotalBoost [91], differing only in the additional details related to capping. The main tools used in this iteration bound are the Pythagorean theorem for Bregman divergences and Pinsker's inequality.

Theorem 3.2 *SoftBoost terminates after at most* $\lceil \frac{2}{\epsilon^2} \ln(N/\nu) \rceil$ *iterations with a convex combination that is at most* ϵ *below the optimum value* g.

Proof We begin by observing that if the optimization problem at iteration t is infeasible, then $P_{LP}^t > \widehat{\gamma}_t - \epsilon \geq g - \epsilon$, where P_{LP}^t is the optimal solution to the soft margin linear programming problem at iteration t. The objective function $\Delta(\mathbf{d}, \mathbf{d}^0)$ is strictly convex in $\mathbf{d}$ and minimized at the interior point $\mathbf{d}^0$. If $\mathbf{d}^t$ contains a zero, then there is no optimal solution in the interior of the simplex. Hence, $P_{LP}^t = \widehat{\gamma}_t - \epsilon \geq g - \epsilon$.

Let $\mathcal{C}_t$ be the convex subset of probability vectors $\mathbf{d} \in \mathcal{S}^N$ satisfying $\mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$ and $\max_{m=1}^t \mathbf{d} \cdot \mathbf{u}^t \leq \widehat{\gamma}_t - \epsilon$. Notice that $\mathcal{C}_0$ is the N dimensional probability simplex where the components are capped to $\frac{1}{\nu}$. The distribution $\mathbf{d}^{t-1}$ at iteration $t-1$ is the projection of $\mathbf{d}^0$ onto the closed convex set $\mathcal{C}_{t-1}$. Because adding a new hypothesis in iteration t results in an additional constraint and $\widehat{\gamma}_t \leq \widehat{\gamma}_{t-1}$, we have $\mathcal{C}_t \subseteq \mathcal{C}_{t-1}$. If $t \leq T-1$, then our termination condition ensures that at iteration $t-1$, the set $\mathcal{C}_{t-1}$ has a feasible solution in the interior of the simplex. Also, $\mathbf{d}^0$ lies in the interior and $\mathbf{d}^t \in \mathcal{C}_t \subseteq \mathcal{C}_{t-1}$. These preconditions ensure that at iteration $t-1$, the projection $\mathbf{d}^{t-1}$ of $\mathbf{d}^0$ onto $\mathcal{C}_{t-1}$, exists and the Generalized Pythagorean Theorem for Bregman divergences

[15, 9] is applicable:

$$\Delta(\mathbf{d}^t, \mathbf{d}^0) - \Delta(\mathbf{d}^{t-1}, \mathbf{d}^0) \geq \Delta(\mathbf{d}^t, \mathbf{d}^{t-1}). \tag{3.5}$$

By Pinsker's inequality [60], $\Delta(\mathbf{d}^t, \mathbf{d}^{t-1}) \geq \frac{(||\mathbf{d}^t - \mathbf{d}^{t-1}||_1)^2}{2}$, and by Hölder's inequality, $||\mathbf{d}^{t-1} - \mathbf{d}^t||_1 \geq ||\mathbf{d}^{t-1} - \mathbf{d}^t||_1 ||\mathbf{u}^t||_\infty \geq \mathbf{d}^{t-1} \cdot \mathbf{u}^t - \mathbf{d}^t \cdot \mathbf{u}^t$. By the definition of $\widehat{\gamma}_t$, $\mathbf{d}^{t-1} \cdot \mathbf{u}^t \geq \widehat{\gamma}_t$. The constraints on the optimization problem assure that $\mathbf{d}^t \cdot \mathbf{u}^t \leq \widehat{\gamma}_t - \epsilon$ and thus $\mathbf{d}^{t-1} \cdot \mathbf{u}^t - \mathbf{d}^t \cdot \mathbf{u}^t \geq \widehat{\gamma}_t - (\widehat{\gamma}_t - \epsilon) = \epsilon$. We conclude that $\Delta(\mathbf{d}^t, \mathbf{d}^{t-1}) \geq \frac{\epsilon^2}{2}$ at iterations 1 through $T-1$. By summing (3.5) over the first $T-1$ iterations, we obtain

$$\Delta(\mathbf{d}^T, \mathbf{d}^0) - \Delta(\mathbf{d}^0, \mathbf{d}^0) \geq (T-1)\frac{\epsilon^2}{2}.$$

Since the left side is at most $\ln(N/\nu)$, the bound of the theorem follows. ■

When $\nu = 1$, then capping is vacuous and the SoftBoost algorithm becomes TotalBoost. Note that the upper bound $\ln(N/\nu)$ on the relative entropy decreases with ν. When $\nu = N$, then the distribution stays at $\mathbf{d}^0$ and the iteration bound is zero.

3.2 Entropy Regularized LPBoost

The SoftBoost algorithm maximized the soft margin and had an $O(\frac{1}{\epsilon^2}\ln\frac{N}{\nu})$, but it also had undesirable characteristics. As we showed in [87], the generalization error decreases slowly in early iterations. Also, the SoftBoost optimization problem had $\widehat{\gamma}_t - \epsilon$ in the edge constraints. The epsilon was necessary but inelegant. Entropy Regularized LPBoost has the same advantages as SoftBoost, but it has a simpler, more

Algorithm 3.2 Entropy Regularized LPBoost

1. **Input:** $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, accuracy parameter $\epsilon > 0$, regularization parameter $\eta > 0$, and smoothness parameter $\nu \in [1, N]$. The canonical value of $\eta = \frac{2}{\epsilon} \ln \frac{N}{\nu}$.

2. **Initialize:** $\mathbf{d}^0$ to the uniform distribution.

3. **Do for** $t = 1, \ldots$

   (a) Send $\mathbf{d}^{t-1}$ to oracle and obtain hypothesis h^t.

   Set $u_n^t = y_n h^t(\mathbf{x}_n)$

   Assume $\mathbf{u}^t \cdot \mathbf{d}^{t-1} \geq g$, where g need not be known.

   (b) Set $\hat{\delta}^t = \min_{q=1\ldots t} P^q(\mathbf{d}^{q-1}) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1})$,

   where $P^t(\mathbf{d}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0)$

   and $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) := -\frac{1}{\eta} \ln \sum_{n=1}^N d_n^0 \exp(-\eta(\sum_{q=1}^t u_n^q w_q + \psi_n)) - \frac{1}{\nu} \sum_{n=1}^N \psi_n$.

   (c) **If** $\hat{\delta}^t \leq \epsilon/2$ **then** set $T = t - 1$ and break.

   (d) **Else** update the distribution to $\mathbf{d}^t = \operatorname{argmin}_{\mathbf{d}} \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0)$.

   s.t. $d_n \leq 1/\nu$, for $n = 1 \ldots N$, $\sum_n d_n = 1$.

4. **Output:** $f_{\mathbf{w}}(\mathbf{x}) = \sum_{q=1}^T w_q^T h^q(\mathbf{x})$, where the coefficients w_q^T are the dual variables to the optimization problem at iteration T.

natural motivation.

In the minimization problem that motivates Entropy Regularized LPBoost (ERLPBoost), a relative entropy is added to the linear programming problem in (2.3):

$$\min_{\mathbf{d}\in\mathcal{S}^N,\, \mathbf{d}\le\frac{1}{\nu}\mathbf{1}} \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}.$$

As in the previous section, the initial distribution $\mathbf{d}^0$ is the uniform distribution. The factor $1/\eta$ is a trade-off parameter between the relative entropy and the maximum edge. The modified mini-max problem is defined as follows:

$$\begin{array}{c} \min \\ \mathbf{d}\cdot\mathbf{1} = 1 \\ \mathbf{d} \le \frac{1}{\nu}\mathbf{1} \end{array} \quad \underbrace{\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d},\mathbf{d}^0)}_{:=P^t(\mathbf{d})}. \tag{3.6}$$

Note that the constraint $\mathbf{d} \ge \mathbf{0}$ was dropped because the relative entropy is not defined for negative d_n, and therefore the constraints $\mathbf{d} \ge \mathbf{0}$ are enforced implicitly. The relative entropy term makes the objective function $P^t(\mathbf{d})$ strictly convex and therefore the minimization problem has a unique solution, which we denote as $\mathbf{d}^t$. We also define $P^t(\mathbf{d})$ when $t = 0$. In this case the maximum in (3.6) is over an empty set of hypotheses and is defined as -1 as before. Thus the minimization problem over $\mathbf{d}$, $P^0(\mathbf{d})$, becomes $-1 + \frac{1}{\eta}\Delta(\mathbf{d},\mathbf{d}^0)$, which is minimized at $\mathbf{d}^0$ and therefore $P^0(\mathbf{d}^0) := -1$.

The Entropy Regularized LPBoost algorithm, shown in Algorithm 3.2, predicts at trial t with this distribution $\mathbf{d}^t$. Note that in the statement of the algorithm we reformulated (3.6) into an equivalent convex optimization problem. If the regularization term $\frac{1}{\eta}\Delta(\mathbf{d},\mathbf{d}^0)$ is dropped from the optimization problem then this algorithm becomes

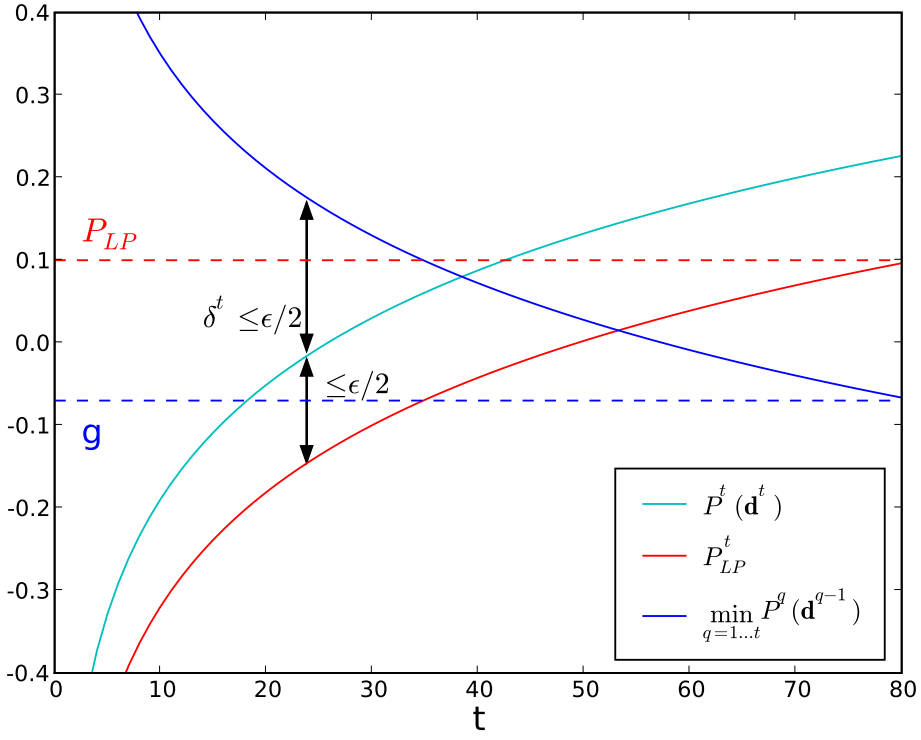


Figure 3.1: Depiction of the simple stopping criterion for ERLPBoost: $\delta^{T+1} \leq \epsilon/2$ implies $g - P^T_{\mathrm{LP}} \leq \epsilon$.

the original LPBoost, whose solution is not unique and depends on the LP solver that is used.

3.2.1 A Simple Stopping Criterion for ERLPBoost

The simple stopping criterion for ERLPBoost algorithm monitors

$$\delta^t := \min_{q=1,2,\ldots,t} P^q(\mathbf{d}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1})$$

and stops when $\delta^{T+1} \leq \epsilon/2$, for a predefined threshold $\epsilon > 0$. All relevant quantities used in the proof are depicted in Figure 3.1. The sequence $\langle P^t_{LP} \rangle$ approaches the line

P_{LP} from below and the sequence $\langle \min_{q=1\ldots t+1} P^q(\mathbf{d}^{q-1})\rangle$ approaches the line g from above. When $g < P_{\mathrm{LP}}$, then both sequences cross, and when $g = P_{\mathrm{LP}}$, then both sequences get arbitrarily close.

Lemma 3.3 *If $\eta \geq \frac{2}{\epsilon}\ln\frac{N}{\nu}$ in (3.6), then $\delta^{T+1} \leq \epsilon/2$ implies $g - P^T_{LP} \leq \epsilon$, where g is the guarantee of the oracle.*

Proof Since $\Delta(\mathbf{d}, \mathbf{d}^0) \leq \ln\frac{N}{\nu}$ and $\eta \geq \frac{2}{\epsilon}\ln\frac{N}{\nu}$, we have $\frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) \leq \epsilon/2$ and

$$P^T(\mathbf{d}^T) \leq P^T_{\mathrm{LP}} + \epsilon/2. \tag{3.7}$$

On the other hand, from the fact that $\Delta(\mathbf{d}, \mathbf{d}^0) \geq 0$ and the assumption on the oracle, we know that

$$g \leq \min_{q=1,2,\ldots,T+1} \mathbf{u}^q \cdot \mathbf{d}^{q-1} \leq \min_{q=1,2,\ldots,T+1} P^q(\mathbf{d}^{q-1}).$$

Subtracting $P^T(\mathbf{d}^T)$ and using the stopping criterion we have

$$g - P^T(\mathbf{d}^T) \quad \leq \quad \min_{q=1,2,\ldots,T+1} P^q(\mathbf{d}^{q-1}) - P^T(\mathbf{d}^T) = \delta^{T+1} \leq \epsilon/2.$$

Adding (3.7) to the above yields $g \leq P^T_{\mathrm{LP}} + \epsilon$. ■

When $\eta \geq \frac{2}{\epsilon}\ln\frac{N}{\nu}$, then the regularization term $\frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0)$ is at most $\epsilon/2$, implying that the values of the regularized problems in the example domain are always at most $\epsilon/2$ larger than the corresponding unregularized problems. The ERLPBoost algorithm produces a final weight vector $\mathbf{w}$ based on the dual of the regularized minimum-maximum edge problem (3.6) given in the next section. In [87] and [90] we compute

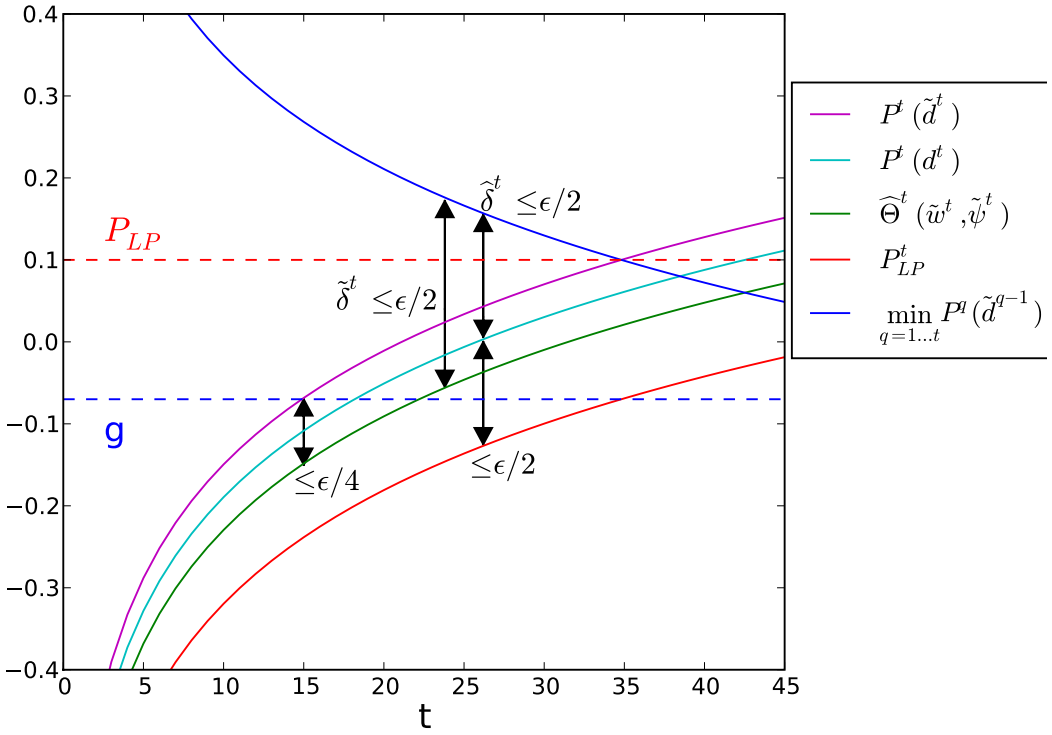


Figure 3.2: Depiction of the practical stopping criterion for ERLPBoost: $\widetilde{\delta}^{T+1}$ implies $g - P^T_{\text{LP}} \leq \epsilon$.

the final $\mathbf{w}$ via the dual of the unregularized optimization problem, but the results in Chapter 2 suggest this choice of $\mathbf{w}$ can result in poor performance.

3.2.2 A Practical Stopping Criterion for ERLPBoost

In practice, computing $\mathbf{d}^{t-1}$ to high precision is too computationally expensive. If the overall precision of our algorithm ϵ is not that high, then it can be argued that our intermediate solution $\mathbf{d}^{t-1}$ does not need to be solved with very high precision either. However, this requires a somewhat different stopping criterion, which was originally

proposed by Vishwanathan [84].

Recall that the simple stopping criterion for ERLPBoost monitors the quantity

$$\delta^t := \min_{q=1,2,\ldots,t} P^q(\mathbf{d}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1})$$

and terminates when $\delta^t \leq \epsilon/2$. Now suppose that $\mathbf{d}^{t-1}$ is not known to high precision. Analysis of following inequality shows that reducing the precision requirement for $\mathbf{d}^{t-1}$ introduces a problem. Let $\widetilde{\mathbf{d}}^{t-1}$ be a low precision approximation of $\mathbf{d}^{t-1}$ and let $\widetilde{\mathbf{w}}^{t-1}$ and $\widetilde{\boldsymbol{\psi}}^{t-1}$ be the dual variables that correspond to the $\widetilde{\mathbf{d}}^{t-1}$. Then by duality,

$$P^{t-1}(\widetilde{\mathbf{d}}^{t-1}) > \min_{\mathbf{d}} P^{t-1}(\mathbf{d}) = \max_{\mathbf{w},\boldsymbol{\psi}} \widehat{\Theta}^{t-1}(\mathbf{w},\boldsymbol{\psi}) > \widehat{\Theta}^{t-1}(\widetilde{\mathbf{w}}^{t-1},\widetilde{\boldsymbol{\psi}}^{t-1})$$

From this, it follows that using the low precision approximation $\widetilde{\mathbf{d}}^{t-1}$ will make δ^t too small, and this will cause the algorithm to terminate prematurely.

To be able to use ERLPBoost with low precision solution $\widetilde{\mathbf{d}}^{t-1}$, we introduce a more practical stopping criterion. At each iteration, we define the necessary precision of $\widetilde{\mathbf{d}}^{t-1}$ by requiring that $P^{t-1}(\widetilde{\mathbf{d}}^{t-1}) - \widehat{\Theta}^t(\widetilde{\mathbf{w}}^{t-1},\widetilde{\boldsymbol{\psi}}^{t-1}) \leq \epsilon/4$. The new stopping criterion monitors

$$\widetilde{\delta}^t := \min_{q=1,2,\ldots,t} P^q(\widetilde{\mathbf{d}}^{q-1}) - \widehat{\Theta}^{t-1}(\widetilde{\mathbf{w}}^{t-1},\widetilde{\boldsymbol{\psi}}^{t-1}),$$

stops when $\widetilde{\delta}^{T+1} \leq \epsilon/2$ for a predefined threshold $\epsilon > 0$. Recall that $\widehat{\Theta}^t$ is the Lagrangian dual of the ERLPBoost optimization problem shown in (3.10). When $\widetilde{\mathbf{d}}^{t-1} = \mathbf{d}^{t-1}$, it is clear that $\widetilde{\delta}^t = \delta^t$ because $P^{t-1}(\mathbf{d}^{t-1}) = \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1},\boldsymbol{\psi}^{t-1})$.

To prove iteration bounds for the practical stopping criterion, we need to introduce one more quantity. In Theorem 3.10, which proves the final iteration bound

for the simple stopping criterion, we require that $\delta^t > \epsilon/2$ for $t = 1 \dots T$ and $\delta^{T+1} \leq \epsilon/2$. We want to prove a similar theorem for the practical stopping criterion. Unfortunately, $\widetilde{\delta}^t > \epsilon/2$ does not imply that $\delta^t > \epsilon/2$ for $t = 1 \dots T$ and $\widetilde{\delta}^{T+1} \leq \epsilon/2$ does not imply $\delta^t \leq \epsilon/2$. The reason is that in this scenario, the low precision $\widetilde{\mathbf{d}}^{t-1}$ is known but the optimal value $\mathbf{d}^{t-1}$ is not. Consequently, the value of $\min_{q=1,2,\dots,t} P^q(\mathbf{d}^{q-1})$ is also unknown and it is not possible to bound it by $\min_{q=1,2,\dots,t} P^q(\widetilde{\mathbf{d}}^{q-1})$. Instead, we define

$$\widehat{\delta}^t := \min_{q=1,2,\dots,t} P^q(\widetilde{\mathbf{d}}^{q-1}) - P^{t-1}(d^{t-1}).$$

When we prove bounds for this stopping criterion, we will work with $\widehat{\delta}^t$ instead of δ^t. This stopping criterion is visualized in Figure 3.2. The main difference between this figure and Figure 3.1 are the two lines that upper and lower bound $P^t(\mathbf{d}^t)$.

The following lemma establishes the relationship between $\widetilde{\delta}$ and $\widehat{\delta}$ before and after termination. These relationships are used to prove iteration bounds ERLPBoost when we use the practical stopping criterion.

Lemma 3.4 *For $t = 1 \dots T$, $\widetilde{\delta}^t > \epsilon/2$ implies $\widehat{\delta}^t > \epsilon/4$. Furthermore, $\widetilde{\delta}^{T+1} \leq \epsilon/2$ implies $\widehat{\delta}^{T+1} \leq \epsilon/2$.*

Proof Using the fact that $\widehat{\Theta}^t(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}) < P^{t-1}(\mathbf{d}^{t-1}) < P^{t-1}(\widetilde{\mathbf{d}}^{t-1})$ we know that $P^{t-1}(\widetilde{\mathbf{d}}^{t-1}) - \widehat{\Theta}^t(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}) \leq \epsilon/4$ implies

$$P^{t-1}(\mathbf{d}^{t-1}) - \widehat{\Theta}^t(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}) \leq \epsilon/4. \tag{3.8}$$

From the termination condition we also know that for $t \leq T$,

$$\min_{q=1,2,\dots,t} P^q(\widetilde{\mathbf{d}}^{q-1}) - \widehat{\Theta}^{t-1}(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}) = \widetilde{\delta}^t > \epsilon/2.$$

Subtracting (3.8) from the above equation yields

$$\min_{q=1,2,\ldots,t} P^q(\widetilde{\mathbf{d}}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1}) = \widehat{\delta}^t > \epsilon/4.$$

The last part of this lemma comes from the fact that $\widehat{\Theta}^t(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}) < P^{t-1}(\mathbf{d}^{t-1})$, which implies

$$\underbrace{\min_{q=1,2,\ldots,t} P^{T+1}(\widetilde{\mathbf{d}}^T) - P^T(\mathbf{d}^T)}_{\widehat{\delta}^{T+1}} \leq \underbrace{\min_{q=1,2,\ldots,t} P^{T+1}(\widetilde{\mathbf{d}}^T) - \widehat{\Theta}^{t-1}(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1})}_{\widetilde{\delta}^{T+1}} \leq \epsilon/2.$$

■

We now want to prove that $\widetilde{\delta}^{T+1} \leq \epsilon/2$ implies $g - P^T_{\mathrm{LP}} \leq \epsilon$.

Lemma 3.5 *If $\eta \geq \frac{2}{\epsilon} \ln \frac{N}{\nu}$ in (3.6), then $\widetilde{\delta}^{T+1} \leq \epsilon/2$ implies $g - P^T_{LP} \leq \epsilon$, where g is the guarantee of the oracle.*

Proof Since $\Delta(\mathbf{d}, \mathbf{d}^0) \leq \ln \frac{N}{\nu}$ for all $\mathbf{d} \in \mathcal{S}^N$ and $\eta \geq \frac{2}{\epsilon} \ln \frac{N}{\nu}$, we have $\frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) \leq \epsilon/2$ and

$$\widehat{\Theta}^T(\widetilde{\mathbf{w}}^T, \widetilde{\boldsymbol{\psi}}^T) \leq P^T(\mathbf{d}^T) \leq P^T_{\mathrm{LP}} + \epsilon/2. \tag{3.9}$$

On the other hand, from the fact that $\Delta(\mathbf{d}, \mathbf{d}^0) \geq 0$ and the assumption on the oracle, we know that

$$g \leq \min_{q=1,2,\ldots,T+1} \mathbf{u}^q \cdot \widetilde{\mathbf{d}}^{q-1} \leq \min_{q=1,2,\ldots,T+1} P^q(\widetilde{\mathbf{d}}^{q-1}).$$

Subtracting $\widehat{\Theta}^T(\widetilde{\mathbf{w}}^T, \widetilde{\boldsymbol{\psi}}^T)$ and using the stopping criterion we have

$$g - \widehat{\Theta}^T(\widetilde{\mathbf{w}}^T, \widetilde{\boldsymbol{\psi}}^T) \quad \leq \quad \min_{q=1,2,\ldots,T+1} P^q(\widetilde{\mathbf{d}}^{q-1}) - \widehat{\Theta}^T(\widetilde{\mathbf{w}}^T, \widetilde{\boldsymbol{\psi}}^T) = \widetilde{\delta}^{T+1} \leq \epsilon/2.$$

Adding (3.9) to the above yields $g \leq P_{\mathrm{LP}}^T + \epsilon$. ■

3.2.3 Lagrangian Dual of ERLPBoost

In this section we compute the Lagrangian dual of (3.6) and discuss the dual relationship between the the problem of optimizing the distribution on the examples versus optimizing the distribution on the current set of hypotheses. For the purpose of this proof, we call the former problem the primal and the latter problem the dual.

Lemma 3.6 *The Lagrangian dual of* (3.6) *is*

$$\max_{\mathbf{w},\boldsymbol{\psi}} \widehat{\Theta}^t(\mathbf{w},\boldsymbol{\psi}), \quad s.t.\ \mathbf{w} \geq \mathbf{0},\ \mathbf{w}\cdot\mathbf{1} = 1, \boldsymbol{\psi} \geq 0, \tag{3.10}$$

$$\textit{where } \widehat{\Theta}^t(\mathbf{w},\boldsymbol{\psi}) := -\frac{1}{\eta}\ln\sum_{n=1}^{N} d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \psi_n)) - \frac{1}{\nu}\sum_{n=1}^{N}\psi_n.$$

The optimal solution $\mathbf{d}^t$ *of* (3.6) *can be expressed in terms of the dual variables* $\mathbf{w}^t$ *and* $\boldsymbol{\psi}^t$ *as follows:*

$$d_n^t := \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q^t + \psi_n^t))}{\sum_{n'} d_{n'}^0 \exp(-\eta(\sum_{q=1}^{t} u_{n'}^q w_q^t + \psi_n^t))}. \tag{3.11}$$

Furthermore, the value of the primal is equal to the value of the dual. Also, for the optimal primal solution $\mathbf{d}^t$ *and optimal dual solution* $\mathbf{w}^t$*,*

$$\sum_{q=1}^{t} w_q^t\ \mathbf{u}^q \cdot \mathbf{d}^t = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t.$$

Proof We start with an equivalent form of (3.6), the optimization problem used in Entropy Regularized LPBoost. In this reformulation, we introduce the variable γ and

the inequality constraints $\mathbf{u}^q \cdot \mathbf{d} \leq \gamma$. This is equivalent to the $\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}$ term from (3.6). The resulting optimization problem is:

$$\begin{aligned} \min_{\mathbf{d},\gamma} \quad & \gamma + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0), \\ \text{s.t.} \quad & \mathbf{u}^q \cdot \mathbf{d} \leq \gamma, \text{ for } 1 \leq q \leq t, \\ & d_n \leq 1/\nu, \text{ for } 1 \leq n \leq N, \\ & \sum_n d_n = 1. \end{aligned}$$

The Lagrangian for this minimization problem is

$$\begin{aligned} L^t(\underbrace{\mathbf{d}, \gamma}_{\text{primal}}, \underbrace{\mathbf{w}, \boldsymbol{\psi}, \beta}_{\text{dual}}) = \gamma + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) + \sum_{q=1}^{t} w_q(\mathbf{u}^q \cdot \mathbf{d} - \gamma) \\ + \sum_{n=1}^{N} \psi_n(d_n - 1/\nu) + \beta(\mathbf{1} \cdot \mathbf{d} - 1). \end{aligned} \tag{3.12}$$

The dual is derived from the Lagrangian by plugging in the minimum value of the primal variables:

$$\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta) := \inf_{\mathbf{d},\gamma} L^t(\mathbf{d}, \gamma, \mathbf{w}, \boldsymbol{\psi}, \beta).$$

Because the Lagrangian is linear w.r.t. γ and a linear function is unbounded below unless it is zero everywhere, the result is the following implicit constraint:

$$\frac{\partial L^t}{\partial \gamma} = 1 - \mathbf{1} \cdot \mathbf{w} = 0.$$

If we enforce the constraint $\mathbf{1} \cdot \mathbf{w} = 1$, then γ vanishes from the Lagrangian. Differentiating the Lagrangian w.r.t. $\mathbf{d}$ shows that the n-th component of the optimal $\mathbf{d}$ vector has the form

$$d^n = d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q - \eta\psi_n - \eta\beta - 1). \tag{3.13}$$

By plugging the optimal $\mathbf{d}$ into the Lagrangian (3.12) and enforcing the constraint $\mathbf{1} \cdot \mathbf{w} = 1$, the dual function simplifies to

$$\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta) = -\frac{1}{\eta} \sum_{n=1}^{N} d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q - \eta\psi_n - \eta\beta - 1) - \beta - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n.$$

This results in the following Lagrange dual:

$$\max_{\mathbf{w}, \boldsymbol{\psi}, \beta} \ \Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta) \tag{3.14}$$

$$\text{s.t. } \mathbf{w} \geq \mathbf{0}, \mathbf{w} \cdot \mathbf{1} = 1, \boldsymbol{\psi} \geq 0, \tag{3.15}$$

By differentiating $\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$ we can determine the optimal choice of β:

$$\beta^t(\mathbf{w}, \boldsymbol{\psi}) = -\frac{1}{\eta} + \frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q - \eta\psi_n). \tag{3.16}$$

Plugging this choice for β into (3.13) results in

$$\begin{aligned} d_n^t :=& d_n^t(\mathbf{w}^t, \boldsymbol{\psi}^t, \beta^t(\mathbf{w}^t, \boldsymbol{\psi}^t)) \\ =& \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q^t - \psi_n^t))}{\sum_{n'} d_{n'}^0 \exp(-\eta(\sum_{q=1}^{t} u_{n'}^q w_q^t - \psi_n^t))}. \end{aligned}$$

Once β is optimized, the Lagrangian becomes

$$\begin{aligned} &\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta^t(\mathbf{w}, \boldsymbol{\psi})) = \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) \\ &:= -\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q - \eta\psi_n) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n. \end{aligned}$$

The dual problem now reduces to

$$\begin{aligned} &\max_{\mathbf{w}, \boldsymbol{\psi}} \ \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) \\ &\text{s.t.} \quad \mathbf{w} \geq \mathbf{0}, \ \mathbf{w} \cdot \mathbf{1} = 1, \boldsymbol{\psi} \geq 0. \end{aligned}$$

The primal objective is convex and the primal constraints are affine. Also, the uniform distribution is always a feasible solution. Therefore, Slater's condition tells us that since this problem has a non-empty feasible set, strong duality holds and the values of the primal and the dual problems are the same.

To prove the last part of the lemma, we observe that when the optimal value of every parameter is plugged into the Lagrangian, its value is equal to that of the optimal value of the original objective function. More precisely,

$$P^t(\mathbf{d}^t) = L^t(\mathbf{d}^t, \gamma^t, \mathbf{w}^t, \boldsymbol{\psi}^t, \beta^t),$$

where $\mathbf{d}^t$ and γ^t are the optimal primal parameters that satisfy the primal constraints and $\mathbf{w}^t,\ \boldsymbol{\psi}^t,\ \beta^t$ are the optimal dual parameters that satisfy the dual constraint. We now develop a simplified expression for $L^t(\mathbf{d}^t, \gamma^t, \mathbf{w}^t, \boldsymbol{\psi}^t, \beta^t)$.

First observe that the primal constraint $(\mathbf{1} \cdot \mathbf{d}^t - 1) = 0$ and the dual constraint $\mathbf{1} \cdot \mathbf{w}^t = 1$ cause β^t and γ^t, respectively, to vanish from the Lagrangian L^t. Furthermore, the KKT conditions [7] imply that the product $\psi_n^t(d_n^t - 1/\nu) = 0$, and this also eliminates $\boldsymbol{\psi}^t$ from L^t. We are left with the partial Lagrangian

$$P^t(\mathbf{d}^t) = L^t(\mathbf{d}^t, \mathbf{w}^t) = \frac{1}{\eta}\Delta(\mathbf{d}^t, \mathbf{d}^0) + \sum_{q=1}^{t} w_q^t \mathbf{u}^q \cdot \mathbf{d}^t.$$

Together with the definition (3.6) of $P^t(\mathbf{d}^t)$, this implies that

$$\sum_{q=1}^{t} w_q^t \mathbf{u}^q \cdot \mathbf{d}^t = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t.$$

■

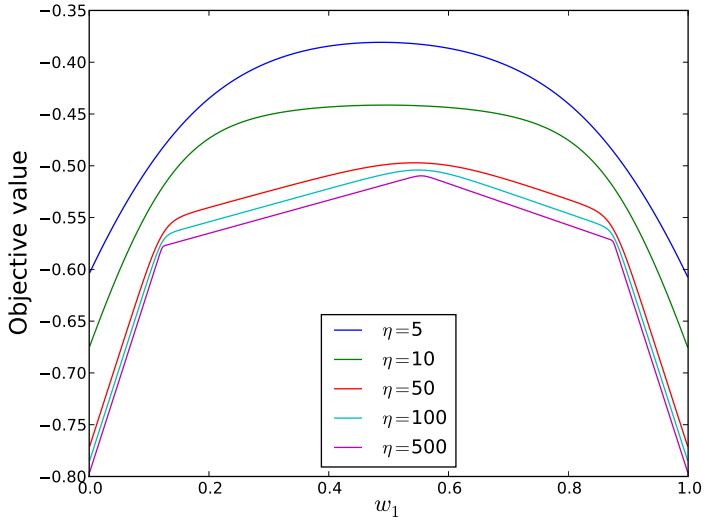


Figure 3.3: $\widehat{\Theta}^t(\mathbf{w})$, the objective function of ERLPBoost in the $\mathbf{w}$ domain vs. w_1 on a simple problem of two hypotheses and four examples. It is concave and differentiable. This is in the uncapped case $\nu = 1$ because otherwise it would not reduce to a 2-dimensional problem.

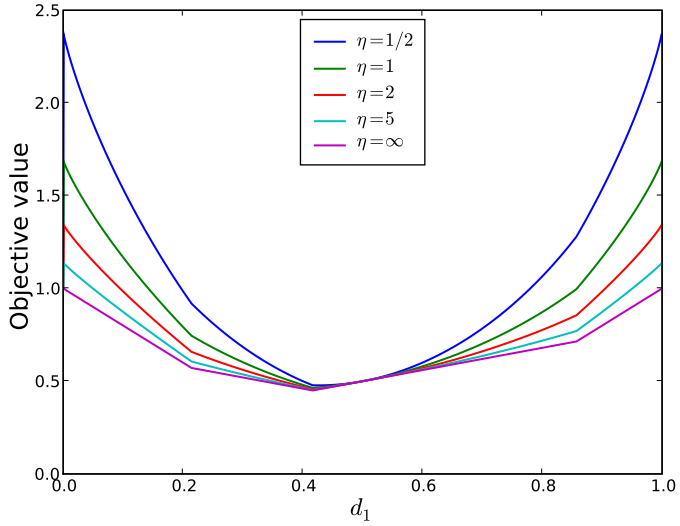


Figure 3.4: $P^t(\mathbf{d})$, the objective function of ERLPBoost in the $\mathbf{d}$ domain, vs. d_1 on a simple problem of four hypotheses and two examples. This function is piecewise continuous and it is not differentiable. This is in the uncapped case $\nu = 1$ because otherwise it would not reduce to a 2-dimensional problem.

Note that $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ is a "smoothed" minimum soft margin of the examples for the linear combination $\mathbf{w}$ of the first t hypotheses. As $\eta \to \infty$, this margin becomes the minimum soft margin. Figure 3.3 plots $\widehat{\Theta}^t(\mathbf{w})$ against w_1 for a simple dataset with four examples and two hypotheses in the uncapped case ($\nu = 1$). Note that the $\boldsymbol{\psi}$ variable does not appear when $\nu = 1$. Because $\mathbf{w}$ is a vector of length 2 and $w_2 = 1 - w_1$, this can be reduced to a 2-dimension problem of $\widehat{\Theta}^t(\mathbf{w})$ vs. w_1. As η increases, the curvature of the objective function decreases and when $\eta = \infty$, $\widehat{\Theta}^t(\mathbf{w})$ is a concave piecewise linear function.

A smoothing effect can also be seen in the $\mathbf{d}$ domain. Whereas LPBoost puts its entire weight onto the examples with minimum soft margin w.r.t. the current hypothesis set $\{h^1, \ldots, h^t\}$, Entropy Regularized LPBoost spreads the weight to examples with higher soft margins by taking the soft min of these margins. The degree of the smoothing is controlled by η. As $\eta \to \infty$, Entropy Regularized LPBoost reverts to an instantiation of LPBoost (i.e. all weight is put on examples with minimum soft margin). Figure 3.4 plots the value of $P^t(\mathbf{d})$ in the uncapped case for a simple problem of four hypotheses and two examples. Because $d_2 = 1 - d_1$, this can be reduced to a 2-dimensional problem of $P^t(\mathbf{d})$ vs. d_1.

3.2.4 Relationship Between Primal and Dual Variables for ERLPBoost

In the process of finding the Lagrangian dual of the ERLPBoost problem, we were able to express the optimal distribution $\mathbf{d}^t$ in terms of $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$:

$$d_n^t := \frac{d_n^0 \exp(-\eta(\sum_{q=1}^t u_n^q w_q^t + \psi_n^t))}{\sum_{n'} d_{n'}^0 \exp(-\eta(\sum_{q=1}^t u_{n'}^q w_q^t + \psi_n^t))}.$$

We now want to know how to find $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$ given $\mathbf{d}^t$. This can be done by solving the system of equations defined by the Karush-Kuhn-Tucker (KKT) optimality conditions, which relate the primal and dual optimal solutions. For the ERLPBoost algorithm, the KKT conditions relating the primal and dual variables are:

$$\mathbf{d}^t \cdot \mathbf{u}^q - \gamma^t \leq 0, \quad q = 1 \dots t \qquad\qquad d_n^t \leq \tfrac{1}{\nu}, \quad n = 1 \dots N$$

$$w_q^t \geq 0, \quad q = 1 \dots t \qquad\qquad \psi_n^t \geq 0, \quad n = 1 \dots N$$

$$w_q^t(\mathbf{u}^q \cdot \mathbf{d}^t - \gamma^t) = 0, \quad q = 1 \dots t \qquad\qquad \psi_n^t(d_n^t - \tfrac{1}{\nu}) = 0, \quad n = 1 \dots N$$

$$\textstyle\sum_n d_n^t - 1 = 0, \quad n = 1 \dots N \qquad\qquad \sum_{q=1}^t w_q^t - 1 = 0, \quad q = 1 \dots t$$

$$\tfrac{1}{\eta}\mathbf{1} + \tfrac{1}{\eta} \ln\left(\tfrac{\mathbf{d}^t}{\mathbf{d}^0}\right) + \textstyle\sum_{q=1}^t w_q^t \mathbf{u}^q + \sum_{n=1}^N \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0$$

where $\mathbf{d}/\mathbf{d}^0$ indicates componentwise division and $\mathbf{e}_n$ is a unit vector whose n^{th} component is 1. Note that the last two equations come from the condition that the gradient of the Lagrangian must be zero. For completeness, the Lagrangian of the ERLPBoost optimization problem is

$$L^t(\underbrace{\mathbf{d}, \gamma}_{\text{primal}}, \underbrace{\mathbf{w}, \boldsymbol{\psi}, \beta}_{\text{dual}}) = \gamma + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) + \sum_{q=1}^t w_q(\mathbf{u}^q \cdot \mathbf{d} - \gamma) + \sum_{n=1}^N \psi_n(d_n - 1/\nu) + \beta(\mathbf{1} \cdot \mathbf{d} - 1).$$

Observe that these equations now require $\mathbf{w}$ to be normalized, which was not the case for SoftBoost. Also recall that β is the Lagrange multiplier for constraint that $\mathbf{d}$ must sum to 1.

Because the optimal value of the primal variables, d^t and γ^t, is already known, we also know which constraints are active. Following the discussion in Appendix A and Chapter 3.1.2, we can use this knowledge to simplify the problem. For a given $\mathbf{d}$, we define the sets of active constraints for $\mathbf{w}$ as $\mathcal{A}_{\mathbf{w}}(\mathbf{d}) := \{q \in 1 \dots t : \mathbf{u}^q \cdot \mathbf{d} = \gamma^t\}$ and $\boldsymbol{\psi}$ as $\mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}) := \{n \in 1 \dots N : d_n = \frac{1}{\nu}\}$. If $\mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)$ and $\mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)$ are the optimal active sets, then (3.6) can be equivalently expressed as

$$\begin{aligned} \min_{\mathbf{d},\gamma} \quad & \gamma + \tfrac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) \\ \text{s.t.} \quad & \mathbf{d} \cdot \mathbf{u}^q = \gamma \text{ for } q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t), \\ & \textstyle\sum_n d_n = 1,\ d_n = \frac{1}{\nu} \text{ for } n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t). \end{aligned}$$

Then the KKT optimality conditions for the ERLPBoost problem at iteration t can be simplified to

$$\begin{aligned} & \textstyle\sum_n d_n^t = 1 \\ & d_n^t = \tfrac{1}{\nu} \text{ for } n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t) \\ & \mathbf{d}^t \cdot \mathbf{u}^q = \gamma^t \text{ for } q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t) \\ & \textstyle\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)}^t w_q^t = 1. \\ & \tfrac{1}{\eta}\mathbf{1} + \tfrac{1}{\eta} \ln\left(\tfrac{\mathbf{d}^t}{\mathbf{d}^0}\right) + \textstyle\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)} w_q \mathbf{u}^q + \sum_{n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)} \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0 \end{aligned}$$

The first three equations are solving for $\mathbf{d}^t$ and γ^t, but since these are known, these equations can be ignored. Therefore, solving for $\mathbf{w}^t$, $\boldsymbol{\psi}^t$, and β^t reduces to solving

the linear system defined by

$$\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)}^{t} w_q^t = 1.$$

$$\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)} w_q \mathbf{u}^q + \sum_{n \in \mathcal{A}_{\psi}(\mathbf{d}^t)} \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = -\frac{1}{\eta}\mathbf{1} - \frac{1}{\eta} \ln\left(\frac{\mathbf{d}^t}{\mathbf{d}^0}\right)$$

The ERLPBoost optimization problem is strictly convex and its feasible set is a closed convex set, so as long as the optimization problem is feasible, then it has a unique optimal solution [7]. As a result, the system of equations defined by the KKT conditions has an exact solution.

3.2.5 Iteration Bound for ERLPBoost

We now bound the number of iterations T needed before the value of the ERLPBoost optimization problem gets within ϵ of the guarantee g. For clarity, the number of iterations corresponds to the number of hypotheses incorporated into the final linear combination, rather than calls to the oracle. The number of calls to the oracle is $T + 1$ but the number of hypotheses in the final linear combination is T. In other words, the hypothesis h^{T+1} obtained in the last call to the oracle is discarded.

In this section, we actually prove two iteration bounds for ERLPBoost. One will use the simple stopping criterion of Chapter 3.2.1 and the other will use the practical stopping criterion of Chapter 3.2.2. We will see that the bound for the former is tighter than the bound for the latter by a factor of two.

Our first technical lemma bounds the increase $P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1})$ by a quadratic term. This lemma is used in iteration bounds for both of the stopping criteria.

Lemma 3.7 *If* $\eta \geq \frac{1}{2}$, *then* $P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1}) \geq \frac{1}{8\eta}(P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}))^2$.

Proof First observe that $P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^{t-1} - \max_{q=1,2,\ldots,t-1} \mathbf{u}^q \cdot \mathbf{d}^{t-1}$. Clearly the first max is at least as large as the second. If both are the same, then the lemma trivially holds because $P^t(\mathbf{d}^{t-1}) = P^{t-1}(\mathbf{d}^{t-1})$. If $P^t(\mathbf{d}^{t-1}) > P^{t-1}(\mathbf{d}^{t-1})$, then the first max equals $\mathbf{u}^t \cdot \mathbf{d}^{t-1}$. We can also rewrite the second max by invoking Lemma 3.6 with $t-1$ instead of t, obtaining

$$P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}) = \mathbf{u}^t \cdot \mathbf{d}^{t-1} - \sum_{q=1}^{t-1} w_q^{t-1} \mathbf{u}^q \cdot \mathbf{d}^{t-1} := (\underbrace{\mathbf{u}^t - \sum_{q=1}^{t-1} w_q^{t-1} \mathbf{u}^q}_{:=\mathbf{x}}) \cdot \mathbf{d}^{t-1}.$$

We still need to show that when $\mathbf{x} \cdot \mathbf{d}^{t-1} \geq 0$, $P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1}) \geq \frac{1}{8\eta}(\mathbf{x} \cdot \mathbf{d}^{t-1})^2$.

By Lemma 3.6, the value of the optimization problem defining Entropy Regularized LPBoost, $P^t(\mathbf{d})$, equals the value of its dual problem. We begin by lower bounding the increase of this value between successive iterations. Let $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$ denote optimal parameters for the dual problem at iteration t. Because the dual is a maximization problem, $\widehat{\Theta}^t(\mathbf{w}^t, \boldsymbol{\psi}^t) \geq \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ for any suboptimal $\mathbf{w} \in \mathcal{S}^t$ and $\boldsymbol{\psi} \geq \mathbf{0}$. For our lower bound on the value we replace $\boldsymbol{\psi}^t$ by the suboptimal previous value $\boldsymbol{\psi}^{t-1}$ and

$\mathbf{w}^t$ by $\mathbf{w}^t(\alpha) = (1-\alpha)\begin{bmatrix} \mathbf{w}^{t-1} \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$, where $\alpha \in [0,1]$:

$$
\begin{aligned}
P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1}) = & \ \widehat{\Theta}^t(\mathbf{w}^t, \boldsymbol{\psi}^t) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1}) \\
& \geq \widehat{\Theta}^t(\mathbf{w}^t(\alpha), \boldsymbol{\psi}^{t-1}) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1}) \\
& = -\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp\left(-\eta \sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta\alpha u_n^t + \eta\, \alpha \sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta \psi_n^{t-1} \right) \\
& \quad + \frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp\left(-\eta \sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta \psi_n^{t-1} \right) \\
& \overset{(3.11)}{=} -\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^{t-1} \exp\left(-\eta\alpha \underbrace{\left(u_n^t - \sum_{q=1}^{t-1} u_n^q w_q^{t-1} \right)}_{:=x_n} \right).
\end{aligned}
$$

This holds for any $\alpha \in [0,1]$. Since $x_n \in [-2,2]$, then $\exp(x_n) \leq$

$\frac{2+x_n}{4}\exp(-2\eta\alpha) + \frac{2-x_n}{4}\exp(2\eta\alpha)$ and this lets us lower bound the above as

$$
\begin{aligned}
& -\frac{1}{\eta} \ln\left(\frac{2 + \mathbf{x}\cdot\mathbf{d}^{t-1}}{4} \exp\left(-2\eta\alpha\right) + \frac{2 - \mathbf{x}\cdot\mathbf{d}^{t-1}}{4} \exp\left(2\eta\alpha\right) \right) \\
& \qquad = 2\alpha - \frac{1}{\eta} \ln\left(1 - \frac{2 - \mathbf{x}\cdot\mathbf{d}^{t-1}}{4} (1 - \exp(4\eta\alpha) \right).
\end{aligned}
$$

By applying the following inequality from [37]

$$
\forall c \in [0,1] \text{ and } r \in \mathbb{R} : \ -\ln(1 - c(1 - e^r)) \geq -cr - \frac{r^2}{8},
$$

the above can be lower bounded by $2\alpha - \frac{2-\mathbf{x}\cdot\mathbf{d}^{t-1}}{4}\ 4\alpha - \frac{16\eta\alpha^2}{8}$. This is maximized at $\alpha = \frac{\mathbf{x}\cdot\mathbf{d}^{t-1}}{4\eta}$ which lies in [0,1] because $\mathbf{x}\cdot\mathbf{d}^{t-1} \in [0,2]$ and $\eta \geq \frac{1}{2}$. Plugging this α into the above, we get $\frac{(\mathbf{x}\cdot\mathbf{d}^{t-1})^2}{8\eta}$ as desired. ■

Using this bound on progress, we prove iteration bounds for ERLPBoost for the simple and the practical stopping criteria respectively. We begin with the simple stopping criterion described in Chapter 3.2.1. Recall the definition of δ^t that is monitored by the simple stopping criterion:

$$\delta^t = \min_{q=1,2,\ldots,t} P^q(\mathbf{d}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1}).$$

As done in [78], we now prove a quadratic recurrence for δ^t.

Lemma 3.8 *If $\eta \geq 1/2$ and $\delta^t \geq 0$, then $\delta^t - \delta^{t+1} \geq \frac{(\delta^t)^2}{8\eta}$, for $1 \leq t \leq T$.*

Proof We prove this recurrence by bounding the inequality of the previous lemma from above and below. We upper bound the l.h.s. via

$$P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1}) \leq \underbrace{\min_{1\leq q\leq t} P^q(\mathbf{d}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1})}_{\delta^t} - \underbrace{\min_{1\leq q\leq t+1} P^q(\mathbf{d}^{q-1}) - P^t(\mathbf{d}^t)}_{\delta^{t+1}}.$$

To lower bound the r.h.s. of the same inequality, first observe that

$$P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}) \geq \min_{1\leq q\leq t} P^q(\mathbf{d}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1}) = \delta^t,$$

Since we assumed $\delta^t \geq 0$, we can lower bound the r.h.s. as

$$\frac{(P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}))^2}{8\eta} \geq \frac{(\delta^t)^2}{8\eta}.$$

■

The lemma requires $\delta^t \geq 0$. The stopping criterion actually ensures that $\delta^t > \frac{\epsilon}{2}$, for $1 \leq t \leq T$. Instead of using a recurrence relation, the standard approach

would be to show that the value of the underlying optimization problem drops by at least a constant. See e.g. [91, 87] for examples for this type of proof. More precisely, in our case we have

$$P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}) \geq \frac{(\delta^t)^2}{8\eta} \geq \frac{1}{32\eta}\epsilon^2.$$

Unfortunately, for our solution to get ϵ-close to the guarantee g, we need that η is inversely proportional to ϵ and therefore this proof technique only yields the iteration bound of $O(\frac{1}{\epsilon^3}\ln\frac{N}{\nu})$. We shall now see that our recurrence method leads to an improved iteration bound of $O(\frac{1}{\epsilon^2}\ln\frac{N}{\nu})$, which is optimal in the hard margin case (when $\nu = 1$).

Lemma 3.9 *Let $\langle\delta^1, \delta^2, \ldots\rangle$ be a sequence of non-negative numbers satisfying the following recurrence, for $t \geq 1$: $\delta^t - \delta^{t+1} \geq c(\delta^t)^2$, where $c > 0$ is a positive constant. Then for all integers $t \geq 1$,*

$$\frac{1}{c(t-1+\frac{1}{\delta^1 c})} \geq \delta^t.$$

This is Sublemma 5.4 of [1] which is is easily proved by induction.

Theorem 3.10 *If $\eta = \max(\frac{2}{\epsilon}\ln\frac{N}{\nu}, \frac{1}{2})$ in (3.6), then Entropy Regularized LPBoost terminates in*

$$T \leq \max(\frac{32}{\epsilon^2}\ln\frac{N}{\nu}, \frac{8}{\epsilon})$$

iterations with a final convex combination of hypotheses for which $g - P^t_{LP} \leq \epsilon$.

Proof Since $\eta \geq \frac{1}{2}$, we can apply Lemma 3.9 with $c = \frac{1}{8\eta}$. This give us $\delta^t \leq \frac{8\eta}{t-1+\frac{8\eta}{\delta^1}}$, which can be rearranged to $t \leq \frac{8\eta}{\delta^t} - \frac{8\eta}{\delta^1} + 1 < \frac{8\eta}{\delta^t}$, since $\eta \geq \frac{1}{2}$ and $0 \leq \delta^1 \leq 1$. By the

stopping criterion, $\delta^T > \epsilon/2$ and $\delta^{T+1} \leq \epsilon/2$. Thus the above inequality implies that

$$T < \frac{8\eta}{\delta^T} \leq \frac{16\eta}{\epsilon}. \tag{3.17}$$

By Lemma 3.3, $\delta^{T+1} \leq \epsilon/2$ implies tolerance ϵ if $\eta \geq \frac{2}{\epsilon} \ln \frac{N}{\nu}$. Plugging $\eta = \max(\frac{2}{\epsilon} \ln \frac{N}{\nu}, \frac{1}{2})$ into the above inequality results in $T \leq \max(\frac{32}{\epsilon^2} \ln \frac{N}{\nu}, \frac{8}{\epsilon})$. Because the aborted iteration $T+1$ is not counted, we arrive at the iteration bound of the theorem. ■

Note that $\frac{2}{\epsilon} \ln \frac{N}{\nu} \geq 1/2$ iff $\epsilon \leq 4 \ln \frac{N}{\nu}$. As pointed out before, when $\eta \to \infty$ then Entropy Regularized LPBoost morphs into a version of LPBoost. Notice that the iteration bound (3.17) is linear in η. Therefore as $\eta \to \infty$, the iteration bound becomes vacuous.

We now prove a similar iteration bound for the practical stopping criterion of Chapter 3.2.2. We will see that the bound we get with the practical stopping criterion will be looser than that of Theorem 3.10 by a factor of two. Recall that this stopping criterion monitors

$$\widetilde{\delta}^t := \min_{q=1,2,\ldots,t} P^q(\widetilde{\mathbf{d}}^{q-1}) - \widehat{\Theta}^{t-1}(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}),$$

and stops when $\widetilde{\delta}^{T+1} \leq \epsilon/2$ for a predefined threshold $\epsilon > 0$. However, the real quantity of interest is $\widehat{\delta}^t$, which is defined as

$$\widehat{\delta}^t := \min_{q=1,2,\ldots,t} P^q(\widetilde{\mathbf{d}}^{q-1}) - P^{t-1}(\mathbf{d}^{t-1}).$$

Observe that $\widehat{\delta}^t$ is extremely similar to δ^t. The difference between the two is that the former uses $\min_{q=1,2,\ldots,t} P^q(\mathbf{d}^{q-1})$ and the latter uses $\min_{q=1,2,\ldots,t} P^q(\widetilde{\mathbf{d}}^{q-1})$.

In the following lemma, we prove a quadratic recurrence for $\widehat{\delta}^t$. This lemma is analogous to Lemma 3.8, which was proved for the simple stopping criterion.

Lemma 3.11 *If* $\eta \geq 1/2$ *and* $\widehat{\delta}^t \geq 0$, *then* $\widehat{\delta}^t - \widehat{\delta}^{t+1} \geq \frac{(\widehat{\delta}^t)^2}{8\eta}$, *for* $1 \leq t \leq T$.

This lemma requires $\widehat{\delta}^t > 0$. According to Lemma 3.4, the practical stopping criterion ensures that $\widehat{\delta}^t > \epsilon/4$ for $t = 1 \ldots T$. The proof is omitted because it is identical to that of Lemma 3.8 except that $\min_{q=1,2,\ldots,t} P^q(\widetilde{\mathbf{d}}^{q-1})$ appears in place of $\min_{q=1,2,\ldots,t} P^q(\mathbf{d}^{q-1})$.

The following theorem contains the final iteration bound for the practical stopping criterion. This theorem is an adaptation of Theorem 3.10 for the practical stopping criterion. The factor of two difference between the two bounds comes from the fact that for $t = 1 \ldots T$, $\delta^t > \epsilon/2$ while $\widehat{\delta}^t > \epsilon/4$.

Theorem 3.12 *If* $\eta = \max(\frac{2}{\epsilon} \ln \frac{N}{\nu}, \frac{1}{2})$ *in* (3.6) *and the practical stopping criterion from Chapter 3.2.2 is used, then Entropy Regularized LPBoost terminates in*

$$T \leq \max(\frac{64}{\epsilon^2} \ln \frac{N}{\nu}, \frac{16}{\epsilon})$$

iterations with a final convex combination of hypotheses for which $g - P^t_{LP} \leq \epsilon$.

The proof is omitted because it is nearly identical to Theorem 3.10. The only difference is that in this proof, $\widehat{\delta}^t > \epsilon/4$ for $t = 1 \ldots T$. On the other hand, in Theorem 3.10 we had that $\delta^t > \epsilon/2$. This is the source of the difference between the two bounds.

3.2.6 Alternative Analysis of ERLPBoost

An equivalent derivation of Lemma 3.7, which bounds the increase $P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1})$ in successive iterations, can be found using the unnormalized relative entropy. The unnormalized relative entropy is defined as

$$\Delta_U := \sum_{n=1}^{N} (d_n \ln \frac{d_n}{d_n^0} - d_n + d_n^0).$$

Recall the definition of the relative entropy: $\Delta(\mathbf{d}, \mathbf{d}_0) := \sum_{n=1}^{N} d_n \ln(d_n/d_n^0)$. When $\mathbf{d}$ and $\mathbf{d}_0$ are probability distributions, $\Delta_U(\mathbf{d}, \mathbf{d}^0) = \Delta(\mathbf{d}, \mathbf{d}^0)$. The optimization problem associated with the unnormalized relative entropy regularization is

$$\begin{array}{c} \min \\ \sum_{n=1}^{N} d_n = 1 \\ \mathbf{d} \leq \mathbf{1}/\nu \end{array} \quad \underbrace{\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta} \Delta_U(\mathbf{d}, \mathbf{d}^0)}_{:= P_U^t(\mathbf{d})}. \tag{3.18}$$

Normalization plays a central role in our analysis. Notice that the normalization constraint of the above optimization problem requires $\mathbf{d}$ to be a probability distribution. As a result, $\Delta_U(\mathbf{d}, \mathbf{d}_0) = \Delta(\mathbf{d}, \mathbf{d}_0)$, so the value of this optimization problem will be identical to (3.6) for all feasible points and thus, they must arrive at the same solution. If the $\sum_{n=1}^{N} d_n = 1$ constraint were not present in the unnormalized entropy problem, then the two optimization problems would be different. As we will see in the next section, the Lagrangians duals of (3.6) and (3.18) appear to be different. However, when we optimize out the Lagrange multiplier that corresponds to the normalization constraint in the the dual of(3.18), it becomes identical to the dual of (3.6). In our analysis, we work with the Lagrangian dual of the unnormalized entropy in the intermediate stage,

before we optimize out the Lagrange multiplier. This is what causes the analysis in this section to differ from the analysis of the previous section. This is interesting because the iteration bound for Binary ERLPBoost, covered in Chapter 3.3, follows directly from the analysis of the unnormalized entropy.

3.2.6.1 Lagrangian Dual of Unnormalized Entropy Problem

We now derive the Lagrangian dual of the unnormalized relative entropy. This is interesting because $P^t(\mathbf{d}^t) = P_U^t(\mathbf{d}^t)$ when $\mathbf{d} \in \mathcal{S}^N$, but the Lagrangians of these two optimization problems are different. Consequently, the Lagrangian duals of these optimization problems, $\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$ and $\Theta_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$ respectively, are also of different form. In particular, $\Theta^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$ is the log partition function and $\Theta_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$ is the partition function *without* the logarithm. However, once we substitute in the optimal value for the dual variable β, we show that $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) = \widehat{\Theta}_U^t(\mathbf{w}, \boldsymbol{\psi})$.

Lemma 3.13 *The Lagrangian dual of* (3.18) *is*

$$\max_{\mathbf{w}, \boldsymbol{\psi}, \beta} \ \widehat{\Theta}_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta), \quad s.t. \ \mathbf{w} \geq \mathbf{0}, \ \mathbf{w} \cdot \mathbf{1} = 1, \boldsymbol{\psi} \geq \mathbf{0}, \beta \in \mathbb{R}.$$

where

$$\widehat{\Theta}_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta) := \frac{1}{\eta} \left(-\sum_{n=1}^{N} d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \psi_n + \beta)) + d_n^0 \right) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n - \beta. \quad (3.19)$$

The optimal solution $\mathbf{d}^t$ *of* (3.18) *can be expressed in terms of the dual variables* $\mathbf{w}^t$ *as follows:*

$$d_n^t := d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \psi_n + \beta)) \quad (3.20)$$

Furthermore, if the optimal β is substituted into $\widehat{\Theta}_U^t(\mathbf{w}, \beta)$, the result is

$$\widehat{\Theta}_U^t(\mathbf{w}, \boldsymbol{\psi}) = -\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \psi_n)) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n.$$

This is equal to the Lagrangian dual of the normalized entropy problem.

Finally, the value of the primal is equal to the value of the dual. Also, for the optimal primal solution $\mathbf{d}^t$ and optimal dual solution $\mathbf{w}^t$,

$$\sum_{q=1}^{t} w_q^t \, \mathbf{u}^q \cdot \mathbf{d}^t = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t.$$

Proof Equation (3.18) can be rewritten as

$$\begin{aligned} \min_{\mathbf{d},\gamma} \quad & \gamma + \frac{1}{\eta} \sum_{n=1}^{N} (d_n \ln \frac{d_n}{d_n^0} - d_n + d_n^0) \\ s.t. \quad & \mathbf{d} \cdot u^q \leq \gamma \text{ for } q = 1 \ldots t \\ & \mathbf{d} \leq \mathbf{1}/\nu, \; \sum_n d_n = 1. \end{aligned} \tag{3.21}$$

The Lagrangian of this optimization problem is

$$\begin{aligned} L_U(\mathbf{d}, \gamma, \mathbf{w}, \boldsymbol{\psi}, \beta) = \quad & \gamma + \tfrac{1}{\eta} \textstyle\sum_{n=1}^{N} (d_n \ln \frac{d_n}{d_n^0} - d_n + d_n^0) + \sum_{q=1}^{t} w_q (\mathbf{d} \cdot u^q - \gamma) \\ & + \textstyle\sum_{n=1}^{N} \psi_n (d_n - \frac{1}{\nu}) + \beta (\sum_{n=1}^{N} d_n - 1). \end{aligned}$$

Because the Lagrangian is linear w.r.t. γ and a linear function is unbounded below unless it is zero everywhere, the result is the following implicit constraint:

$$\frac{\partial L_U}{\partial \gamma} = 1 - \sum_{q=1}^{t} w_q = 0.$$

The partial derivative of L_U w.r.t. d_n is

$$\frac{\partial L_U}{\partial d_n} = \frac{1}{\eta} (\ln \frac{d_n}{d_n^0}) + \sum_{q=1}^{t} w_q u_n^q + \psi_n + \beta.$$

The update for d_n takes the form $d_n = d_n^0 \exp(-\eta(\sum_{q=1}^t w_q u_n^q + \psi_n + \beta))$. Plugging the update back into the Lagrangian results in the dual problem

$$\Theta_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta) := \max_{\mathbf{w}, \boldsymbol{\psi}, \beta} \quad \frac{1}{\eta}\left(-\sum_{n=1}^N d_n^0 \exp(-\eta(\sum_{q=1}^t w_q u_n^q + \psi_n + \beta)) + d_n^0\right) - \frac{1}{\nu}\sum_{n=1}^N \psi_n - \beta$$
$$s.t. \qquad \sum_{q=1}^t w_q = q, \ \boldsymbol{\psi} \geq 0.$$

This is not the same as the dual for the normalized relative entropy, shown in (3.14). However, after we substitute in the optimal choice for β, the Lagrangian duals become identical. By differentiating $\Theta_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$, we can determine the optimal choice for β:

$$\beta(\mathbf{w}, \boldsymbol{\psi}) = \frac{1}{\eta} \ln \sum_{n=1}^N d_n^0 \exp(-\eta(\sum_{q=1}^t u_n^q w_q + \psi_n)).$$

Note that this is not the same as the optimal β for normalized entropy, shown in (3.16). Once β is optimized, the result is

$$\widehat{\Theta}_U^t(\mathbf{w}, \boldsymbol{\psi}) = -\frac{1}{\eta} \ln \sum_{n=1}^N d_n^0 \exp(-\eta(\sum_{q=1}^t w_q u_n^q + \psi_n)) - \frac{1}{\nu}\sum_{n=1}^N \psi_n.$$

This is identical to (3.10), the dual of the normalized relative entropy.

The final step in this proof is to show that

$$\sum_{q=1}^t w_q^t \ \mathbf{u}^q \cdot \mathbf{d}^t = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t.$$

We do so by showing that the l.h.s. and the r.h.s. of the above equation are both equal to γ. From (3.21), it is clear that $\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t = \gamma$. Because $\mathbf{w}^t$ is the optimal dual variable, we know that complementary slackness applies. This means that either $\mathbf{u}^q \cdot \mathbf{d}^t = \gamma$ or $w_q^t = 0$. We also know that $\sum_{q=1}^t w_q^t = 1$. The l.h.s. of the above equation is therefore equivalent to $\gamma \sum_{q=1}^t w_q^t = \gamma$. ■

3.2.6.2 Alternate Bound on Progress

In Lemma 3.7, we bounded the increase $P^t(\mathbf{d}^t) - P^t(\mathbf{d}^{t-1})$ in terms of a quadratic term. Because $P_U^t(\mathbf{d}) = P^t(\mathbf{d})$ for all $\mathbf{d} \in \mathcal{S}^N$, we can use the unnormalized entropy to prove an alternate version of Lemma 3.7 by bounding the increase of $P_U^t(\mathbf{d}^t) - P_U^t(\mathbf{d}^{t-1})$. Recall that the Lagrangian dual of $P^t(\mathbf{d})$ is identical to dual of $P_U^t(\mathbf{d})$ when β is optimized out, but if β is not optimized out, they differ. By using the unnormalized entropy without optimizing β, we can arrive at the same bound via a different path. Most importantly, the inequalities used in this proof turn out to be useful in proving bounds for Binary ERLPBoost, discussed in Chapter 3.3.

Lemma 3.14 *If $\eta \geq 1/2$, then $P^t(\mathbf{d}^t) - P^{t-1}(\mathbf{d}^{t-1}) \geq \frac{1}{8\eta}(P^t(\mathbf{d}^{t-1}) - P^{t-1}(\mathbf{d}^{t-1}))^2$.*

Proof Because $P_U^t(\mathbf{d}) = P^t(\mathbf{d})$ for all $\mathbf{d} \in \mathcal{S}^N$, it suffices to bound the increase of $P_U^t(\mathbf{d}^t) - P_U^t(\mathbf{d}^{t-1})$, the value of the objective function for the unnormalized entropy. First observe that $P_U^t(\mathbf{d}^{t-1}) - P_U^{t-1}(\mathbf{d}^{t-1}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^{t-1} - \max_{q=1,2,\ldots,t-1} \mathbf{u}^q \cdot \mathbf{d}^{t-1}$. Clearly the first max is at least as large as the second. If both are the same, then the lemma trivially holds because $P_U^t(\mathbf{d}^{t-1}) = P_U^{t-1}(\mathbf{d}^{t-1})$. If $P_U^t(\mathbf{d}^{t-1}) > P_U^{t-1}(\mathbf{d}^{t-1})$, then the first max equals $\mathbf{u}^t \cdot \mathbf{d}^{t-1}$. We can also rewrite the second max by invoking Lemma 3.13 with $t-1$ instead of t, obtaining

$$P_U^t(\mathbf{d}^{t-1}) - P_U^{t-1}(\mathbf{d}^{t-1}) = \mathbf{u}^t \cdot \mathbf{d}^{t-1} - \sum_{q=1}^{t-1} w_q^{t-1} \mathbf{u}^q \cdot \mathbf{d}^{t-1} := \underbrace{(\mathbf{u}^t - \sum_{q=1}^{t-1} w_q^{t-1} \mathbf{u}^q)}_{:=\mathbf{x}} \cdot \mathbf{d}^{t-1}.$$

We still need to show that when $\mathbf{x} \cdot \mathbf{d}^{t-1} \geq 0$, $P_U^t(\mathbf{d}^t) - P_U^{t-1}(\mathbf{d}^{t-1}) \geq \frac{1}{8\eta}(\mathbf{x} \cdot \mathbf{d}^{t-1})^2$.

By Lemma 3.13, the value of $P_U^t(\mathbf{d})$ at its optimal solution equals the value of its dual problem. We begin by lower bounding the increase of this value between successive iterations. Let $\mathbf{w}^t$, $\boldsymbol{\psi}^t$, and β^t denote optimal parameters for (3.19), the dual problem at iteration t. Because the dual is a maximization problem, $\widehat{\Theta}_U^t(\mathbf{w}^t, \boldsymbol{\psi}^t, \beta^t) \geq \widehat{\Theta}_U^t(\mathbf{w}, \boldsymbol{\psi}, \beta)$ for any suboptimal $\mathbf{w} \in \mathcal{S}^t$, $\boldsymbol{\psi} \geq \mathbf{0}$, and β. For our lower bound on the value we replace $\boldsymbol{\psi}^t$ by the suboptimal previous value $\boldsymbol{\psi}^{t-1}$, we replace β^t by β^{t-1}, and we replace $\mathbf{w}^t$ by $\mathbf{w}^t(\alpha) = (1-\alpha)\begin{bmatrix} \mathbf{w}^{t-1} \\ \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{0} \\ \\ 1 \end{bmatrix}$, where $\alpha \in [0,1]$:

$$
\begin{aligned}
P_U^t(\mathbf{d}^t) - P_U^{t-1}(\mathbf{d}^{t-1}) = \; & \widehat{\Theta}_U^t(\mathbf{w}^t, \boldsymbol{\psi}^t, \beta^t) - \widehat{\Theta}_U^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1}, \beta^{t-1}) \\
\geq \; & \widehat{\Theta}_U^t(\mathbf{w}^t(\alpha), \boldsymbol{\psi}^{t-1}, \beta^{t-1}) - \widehat{\Theta}_U^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1}, \beta^{t-1}) \\
= \; & -\frac{1}{\eta} \sum_{n=1}^{N} d_n^0 \exp(-\eta(1-\alpha) \sum_{q=1}^{t} w_q u_n^q - \eta\alpha u_n^t - \eta\psi_n^{t-1} - \eta\beta^{t-1}) \\
& + \frac{1}{\eta} \sum_{n=1}^{N} d_n^0 \exp(-\eta \sum_{q=1}^{t} w_q u_n^q - \eta\psi_n^{t-1} - \eta\beta^{t-1}) \\
= \; & \frac{1}{\eta} \left(1 - \sum_{n=1}^{N} d_n^{t-1} \exp(-\eta\alpha \underbrace{(u_n^t - \sum_{q=1}^{t} w_q u_n^q)}_{x_n}) \right) \qquad (3.22) \\
\geq \; & \frac{1}{\eta} \left(1 - \sum_{n=1}^{N} d_n^{t-1} \left(\frac{2+x_n}{4} \exp(-2\eta\alpha) + \frac{2-x_n}{4} \exp(2\eta\alpha) \right) \right) \\
= \; & \frac{1}{\eta} \left(1 - \frac{2 + \mathbf{d}\cdot\mathbf{x}}{4} \exp(-2\eta\alpha) - \frac{2 - \mathbf{d}\cdot\mathbf{x}}{4} \exp(2\eta\alpha) \right) \qquad (3.23)
\end{aligned}
$$

First observe that the inequality holds for any $\alpha \in [0,1]$, so choose $\alpha = \mathbf{d}\cdot\mathbf{x}/(4\eta)$. It is easy to see that this value of α is valid as long as $\eta \geq 1/2$. Also, for notational simplicity, we introduce the change of variable $z = \mathbf{d}\cdot\mathbf{x}$. Making these substitutions

and rearranging (3.23) results in

$$\begin{aligned} & \frac{1}{\eta}\left(1-\frac{1}{2}(\exp(z/2)+\exp(-z/2))+\frac{z}{4}(\exp(z/2)-\exp(-z/2))\right) \\ = \quad & \frac{1}{\eta}\left(1-\cosh(z/2)+\frac{z}{2}\sinh(z/2)\right). \end{aligned} \tag{3.24}$$

We now reason that the above expression is lower bounded by $\frac{z^2}{8\eta}$ which gives us the promised lower bound of $\frac{1}{8\eta}(P_U^t(\mathbf{d}^{t-1}) - P_U^{t-1}(\mathbf{d}^{t-1}))^2$ and completes the proof of the theorem.

To see this observe that the derivative of the expression (3.24) minus $\frac{z^2}{8\eta}$ is $\frac{z}{4\eta}\cosh(z/2) - \frac{z}{4\eta}$ which is 0 at $z = 0$ and is antisymmetric around 0. Thus the expression (3.24) minus $\frac{z^2}{8\eta}$ has a minimum at $z = 0$. It is easy to check that the value of the minimum is 0 and therefore expression (3.24) is lower bounded by $\frac{z^2}{8\eta}$. ■

3.3 Binary Entropy Regularized LPBoost

In this section we present *Binary Entropy Regularized LPBoost* (Binary ERLPBoost). We motivate this algorithm by showing that at each iteration it solves an optimization problem of lower dimension than ERLPBoost. At iteration t, ERLPBoost finds updated distribution $\mathbf{d}^t$ that minimizes (3.6). In practice, we actually find $\mathbf{d}^t$ by minimizing $-\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ in (3.10) for $\mathbf{w}$ and $\boldsymbol{\psi}$ and then substituting these values into (3.11). The key observation is that the capping constraints enforced by ERLPBoost make the $\boldsymbol{\psi}$ variables appear in the $\mathbf{w}$ domain, and this greatly increases the dimensionality of the problem. Because $\mathbf{w}$ is of dimension t and $\boldsymbol{\psi}$ is of dimension N, ERLPBoost

solves an optimization problem of dimension of $t+N$, where $N \gg t$. On the other hand, Binary ERLPBoost incorporates the capping constraints directly into the binary relative entropy. Therefore, the $\boldsymbol{\psi}$ variables are not needed. The normalization constraint can be expressed in closed form in ERLPBoost, but not in Binary ERLPBoost. As a result, Binary ERLPBoost has only has $t+1$ variables. Optimization problems scale poorly with the number of input variables, so using Binary ERLPBoost could potentially be more computationally efficient.

If we replace the relative entropy regularizer with the binary entropy, we can reduce the optimization problem to one of $t+1$ variables. Since $N \gg t$ and optimization problems scale poorly with the number of input variables, this could potentially result in greatly improved computational efficiency.

In the minimization problem that motivates Binary Entropy Regularized LPBoost, shown in Algorithm 3.3, a binary relative entropy is added to the linear programming problem in the example domain. The binary entropy of a single random variable can be interpreted as the entropy of a Bernoulli trial with probability of success d, and is defined as $-d\ln d-(1-d)log(1-d)$. The binary relative entropy between probability distributions $\mathbf{d}$ and $\mathbf{d}^0$ is defined as

$$\Delta_2(\mathbf{d},\mathbf{d}^0)=\sum_{n=1}^{N} d_n \ln\frac{d_n}{d_n^0}+\sum_{n=1}^{N}(1-d_n)\ln\frac{1-d_n}{1-d_n^0}.$$

The Binary ERLPBoost algorithm must solve an optimization problem involving this quantity at every iteration, but with the constraints that $0 \leq d_n \leq \frac{1}{\nu}$ for $n=1\ldots N$. Note that this constitutes $2N$ interval constraints. Since N is large, eliminating these

constraints would be helpful.

There is a generalization of the binary relative entropy that restricts the elements of $\mathbf{d}$ and $\mathbf{d}^0$ to lie within two real numbers L and M, where $L < M$ [12]:

$$\Delta_{2,L,M}(\mathbf{d}, \mathbf{d}^0) = \sum_{n=1}^{N} (d_n - L) \ln \frac{d_n - L}{d_n^0 - L} + \sum_{n=1}^{N} (M - d_n) \ln \frac{M - d_n}{M - d_n^0}.$$

Using this generalization allows us to eliminate the $2N$ interval constraints, which should therefore result in a more efficient optimization problem. In our case, we want the restriction $0 \leq d \leq \frac{1}{\nu}$, so the form of the binary entropy that we want to use in our optimization problem would look like

$$\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) = \sum_{n=1}^{N} d_n \ln \frac{d_n}{d_n^0} + \sum_{n=1}^{N} (\frac{1}{\nu} - d_n) \ln \frac{\frac{1}{\nu} - d_n}{\frac{1}{\nu} - d_n^0}.$$

The modified mini-max problem is defined as follows:

$$\min_{\mathbf{d} \in \mathcal{S}^N} \underbrace{\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta} \Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0)}_{:= \mathbf{P}_B^t(\mathbf{d})}. \tag{3.25}$$

The factor $1/\eta$ is a trade-off parameter between the relative entropy and the maximum edge.

3.3.1 A Simple Stopping Criterion for Binary ERLPBoost

The stopping criterion for Binary ERLPBoost requires that $\eta \geq \frac{2}{\epsilon} \left(\ln \frac{N}{\nu} + 1 \right)$. Since $\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) \leq \ln \frac{N}{\nu} + 1$, our condition ensures that $\frac{1}{\eta} \Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) \leq \epsilon$ for all $\mathbf{d} \in \mathcal{S}^N$. This is slightly different from the stopping criterion for ERLPBoost, which requires $\eta \geq \frac{2}{\epsilon} \ln \frac{N}{\nu}$.

Algorithm 3.3 Binary Entropy Regularized LPBoost

1. **Input:** $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, accuracy parameter $\epsilon > 0$, regularization parameter $\eta > 0$, and smoothness parameter $\nu \in [1, N]$. The canonical value of $\eta = \frac{2}{\epsilon}(\ln \frac{N}{\nu} + 1)$.

2. **Initialize:** $\mathbf{d}^0$ to the uniform distribution.

3. **Do for** $t = 1, \ldots$

   (a) Send $\mathbf{d}^{t-1}$ to oracle and obtain hypothesis h^t.

   Set $u_n^t = y_n h^t(\mathbf{x}_n)$

   Assume $\mathbf{u}^t \cdot \mathbf{d}^{t-1} \geq g$, where g need not be known.

   (b) Set $\widetilde{\delta}_B^t = \min_{q=1\ldots t} \mathbf{P}^q(\mathbf{d}^{q-1}) - \widehat{\Theta}_B^{t-1}(\mathbf{w}^{t-1}, \beta^{t-1})$,

   where $\mathbf{P}^t(\mathbf{d}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0)$

   and $\widehat{\Theta}_B^t(\mathbf{w}, \beta) := -\frac{1}{\eta\nu} \sum_{n=1}^N \ln(1 - \nu\mathbf{d}_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^t w_q u_n^q + \beta))) - \beta$.

   (c) **If** $\widetilde{\delta}_B^t \leq \epsilon/2$ **then** set $T = t - 1$ and break.

   (d) **Else** Update the distribution to

$$\mathbf{d}^t = \operatorname*{argmin}_{\mathbf{d} \in \mathcal{S}^N} \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0).$$

4. **Output:** $f_{\mathbf{w}}(\mathbf{x}) = \sum_{q=1}^T w_q^T h^q(\mathbf{x})$, where the coefficients w_q^T are the dual variables to the above optimization problem at iteration T.

The simple stopping criterion for Binary ERLPBoost algorithm monitors

$$\delta_B^t := \min_{q=1,2,\ldots,t} P_B^q(\mathbf{d}^{q-1}) - P_B^{t-1}(\mathbf{d}^{t-1})$$

and stops when $\delta_B^{T+1} \leq \epsilon/2$, for a predefined threshold $\epsilon > 0$. This stopping criterion is comparable to the simple stopping criterion for ERLPBoost that was covered in Chapter 3.2.1.

Lemma 3.15 *If* $\eta \geq \frac{2}{\epsilon}\left(\ln \frac{N}{\nu} + 1\right)$ *in* (3.25), *then* $\delta_B^{T+1} \leq \epsilon/2$ *implies* $g - P_{LP}^T \leq \epsilon$, *where* g *is the guarantee of the oracle.*

Because the proof of this lemma is identical to Lemma 3.3, it has been omitted.

3.3.2 A Practical Stopping Criterion for Binary ERLPBoost

We now present a practical stopping criterion for Binary ERLPBoost that enables us to use a low precision estimate of $\mathbf{d}^{t-1}$ to improve the computational efficiency of the algorithm. This stopping criterion is comparable to the one proposed in Chapter 3.2.2 for ERLPBoost. The primary difference between this stopping criterion and the one in Chapter 3.2.2 is the value of η. As we mentioned in the previous section, the stopping criterion for Binary ERLPBoost requires that $\eta \geq \frac{2}{\epsilon}\left(\ln \frac{N}{\nu} + 1\right)$. Since $\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) \leq \ln \frac{N}{\nu} + 1$, our condition ensures that $\frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) \leq \epsilon$ for all $\mathbf{d} \in \mathcal{S}^N$. This is slightly different from the stopping criterion for ERLPBoost, which requires $\eta \geq \frac{2}{\epsilon} \ln \frac{N}{\nu}$.

Let $\widetilde{\mathbf{d}}^{t-1}$ be a low precision approximation of $\mathbf{d}^{t-1}$ and let $\widetilde{\mathbf{w}}^{t-1}$ and $\widetilde{\boldsymbol{\psi}}^{t-1}$ be the dual variables that correspond to the $\widetilde{\mathbf{d}}^{t-1}$. As in Chapter 3.2.2, we define the

necessary precision of $\widetilde{\mathbf{d}}^{t-1}$ by requiring that $P_B^{t-1}(\widetilde{\mathbf{d}}^{t-1}) - \widehat{\Theta}_B^t(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}) \leq \epsilon/4$. This stopping criterion monitors

$$\widetilde{\delta}_B^t := \min_{q=1,2,\ldots,t} P_B^q(\widetilde{\mathbf{d}}^{q-1}) - \widehat{\Theta}_B^{t-1}(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}),$$

stops when $\widetilde{\delta}^{T+1} \leq \epsilon/2$ for a predefined threshold $\epsilon > 0$. Note that $\widehat{\Theta}_B^t$ is the Lagrangian dual of the Binary ERLPBoost optimization problem shown in (3.26).

To prove an iteration bound for Binary ERLPBoost using this stopping criterion, we need to introduce one more quantity. In Theorem 3.21, which proves the final iteration bound for the simple stopping criterion, we require that $\delta_B^t > \epsilon/2$ for $t = 1 \ldots T$ and $\delta_B^{T+1} \leq \epsilon/2$. We want to prove a similar theorem for the practical stopping criterion. Unfortunately, $\widetilde{\delta}_B^t > \epsilon/2$ does not imply that $\delta_B^t > \epsilon/2$ for $t = 1 \ldots T$ and $\widetilde{\delta}_B^{T+1} \leq \epsilon/2$ does not imply $\delta_B^t \leq \epsilon/2$. The reason is that in this scenario, the low precision $\widetilde{\mathbf{d}}^{t-1}$ is known but the optimal value $\mathbf{d}^{t-1}$ is not. Consequently, the value of $\min_{q=1,2,\ldots,t} P_B^q(\mathbf{d}^{q-1})$ is also unknown and it is not possible to bound it by $\min_{q=1,2,\ldots,t} P_B^q(\widetilde{\mathbf{d}}^{q-1})$. To overcome this difficulty, we define

$$\widehat{\delta}_B^t := \min_{q=1,2,\ldots,t} P_B^q(\widetilde{\mathbf{d}}^{q-1}) - P_B^{t-1}(d^{t-1}).$$

When we prove bounds for this stopping criterion, we will work with $\widehat{\delta}_B^t$ instead of δ_B^t.

This next lemma establishes the relationship between $\widetilde{\delta}_B$ and $\widehat{\delta}_B$ before and after termination. These relationships are used to prove iteration bounds Binary ERLPBoost when we use the practical stopping criterion introduced in this section.

Lemma 3.16 *For $t = 1 \ldots T$, $\widetilde{\delta}_B^t > \epsilon/2$ implies $\widehat{\delta}_B^t > \epsilon/4$. Furthermore, $\widetilde{\delta}_B^{T+1} \leq \epsilon/2$ implies $\widehat{\delta}_B^{T+1} \leq \epsilon/2$.*

The proof is omitted because it is identical to the proof of Lemma 3.4.

We now want to prove that $\widetilde{\delta}_B^{T+1} \leq \epsilon/2$ implies $g - P_{\mathrm{LP}}^T \leq \epsilon$.

Lemma 3.17 *If* $\eta \geq \frac{2}{\epsilon}(\ln \frac{N}{\nu} + 1)$ *in* (3.25)*, then* $\widetilde{\delta}_B^{T+1} \leq \epsilon/2$ *implies* $g - P_{LP}^T \leq \epsilon$*, where* g *is the guarantee of the oracle.*

Because it is identical to that of Lemma 3.5, this proof is omitted as well.

3.3.3 Lagrangian Dual of Binary ERLPBoost

In this section we derive the Lagrangian dual of the binary relative entropy optimization problem. This dual differs from the previous duals in two important respects. In all of the previous algorithms, an optimal value for β, the Lagrange multiplier for the normalization constraint, had a closed form. This value could be substituted into the dual to get a closed form solution for $\mathbf{d}^t$ in terms of $\mathbf{w}$ and $\boldsymbol{\psi}$ alone. In contrast, the binary entropy problem does not allow for a similar closed form solution.

Another important difference between ERLPBoost and Binary ERLPBoost is the number of dual variables. The dual variables for ERLPBoost are $\mathbf{w}$ and $\boldsymbol{\psi}$. Because $\mathbf{w}$ is of dimension t and $\boldsymbol{\psi}$ is of dimension N, we must solve an optimization problem of dimension $t + N$. In contrast, the dual variables for Binary ERLPBoost are $\mathbf{w}$ and β, where $\mathbf{w}$ is of dimension t and β is a scalar. Consequently, the Lagrangian dual of the Binary ERLPBoost problem has a total of $t + 1$ variables, which is much less than $t + N$.

Lemma 3.18 *The Lagrangian dual of* (3.25) *is*

$$\max_{\mathbf{w},\beta} \ \widehat{\Theta}_B^t(\mathbf{w},\beta), \quad s.t. \ \mathbf{w} \geq \mathbf{0}, \ \mathbf{w}\cdot\mathbf{1} = 1, \beta \in \mathbb{R}. \tag{3.26}$$

$$\text{where } \widehat{\Theta}_B^t(\mathbf{w},\beta) := -\frac{1}{\eta\nu}\sum_{n=1}^{N} \ln(1-\nu\mathbf{d}_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))) - \beta.$$

The optimal solution $\mathbf{d}^t$ *of* (3.25) *can be expressed in terms of the dual variables* $\mathbf{w}^t$ *as follows:*

$$d_n^t := \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))}{1-\nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))} \tag{3.27}$$

Furthermore, the value of the primal is equal to the value of the dual. Also, for the optimal primal solution $\mathbf{d}^t$ *and optimal dual solution* $\mathbf{w}^t$*,*

$$\sum_{q=1}^{t} w_q^t \ \mathbf{u}^q \cdot \mathbf{d}^t = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t.$$

Proof The optimization problem in (3.25) can be rewritten equivalently as

$$\begin{aligned} \min_{\gamma,\mathbf{d}} \quad & \gamma + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d},\mathbf{d}^0) \\ s.t. \quad & \mathbf{d}\cdot\mathbf{u}^q \leq \gamma \ \text{ for } \ q = 1\ldots t \\ & \sum_{n=1}^{n} d_n = 1 \ \text{ for } \ n = 1\ldots N \end{aligned}$$

The Lagrangian of this optimization problem at iteration t is

$$L^t = \gamma + \frac{1}{\eta}\left[\sum_{n=1}^{N} d_n \ln\frac{d_n}{d_n^0} + \sum_{n=1}^{N}(\frac{1}{\nu}-d_n)\ln\frac{\frac{1}{\nu}-d_n}{\frac{1}{\nu}-d_n^0}\right] + \sum_{q=1}^{t} w_q(\mathbf{d}\cdot\mathbf{u}^q - \gamma) + \beta(\sum_{n=1}^{N} d_n - 1) \tag{3.28}$$

The dual is derived from the Lagrangian by plugging in the minimum value of the primal variables:

$$\Theta_B^t(\mathbf{w},\beta) := \inf_{\mathbf{d},\gamma} L^t(\mathbf{d},\gamma,\mathbf{w},\beta).$$

Because the Lagrangian is linear w.r.t. γ and a linear function is unbounded below unless it is zero everywhere, the result is the following implicit constraint:

$$\frac{\partial L^t}{\partial \gamma} = 1 - \mathbf{1} \cdot \mathbf{w} = 0.$$

Therefore, if we enforce the constraint $\mathbf{1} \cdot \mathbf{w} = 1$, then γ vanishes from the Lagrangian. Differentiating the Lagrangian w.r.t. $\mathbf{d}$ results in

$$\frac{\partial L}{\partial d_n} = \frac{1}{\eta}\left[\ln \frac{d_n}{d_n^0} - \ln \frac{\frac{1}{\nu} - d_n}{\frac{1}{\nu} - d_n^0}\right] + \sum_{q=1}^{t} w_q u_n^q + \beta.$$

The update for d_n takes the form

$$d_n^{t+1} = \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))}.$$

Plugging this update back into (3.28) results in the dual

$$\widehat{\Theta}_B^t(\mathbf{w}, \beta) = -\frac{1}{\eta\nu}\sum_{n=1}^{N} \ln(1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta)) - \beta.$$

The dual problem now reduces to

$$\begin{aligned} \max_{\mathbf{w},\beta} \quad & \widehat{\Theta}_B^t(\mathbf{w}, \beta) \\ \text{s.t.} \quad & \mathbf{w} \geq \mathbf{0},\ \mathbf{w} \cdot \mathbf{1} = 1,\ \beta \in \mathbb{R}. \end{aligned}$$

The final step in this proof is to show that

$$\sum_{q=1}^{t} w_q^t\ \mathbf{u}^q \cdot \mathbf{d}^t = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t.$$

We do so by showing that the l.h.s. and the r.h.s. of the above equation are both equal to γ. From (3.21), it is clear that $\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^t = \gamma$. Because $\mathbf{w}^t$ is the optimal dual variable, we know that complementary slackness applies. This means that either

$\mathbf{u}^q \cdot \mathbf{d}^t = \gamma$ or $w_q^t = 0$. We also know that $\sum_{q=1}^t w_q^t = 1$. The l.h.s. of the above equation is therefore equivalent to $\gamma \sum_{q=1}^t w_q^t = \gamma$. ■

It is useful to visualize the optimization problems in Binary ERLPBoost in the $\mathbf{d}$ and $\mathbf{w}$ domains. Figure 3.5 plots the objective function $P^t(\mathbf{d}^t)$ of (3.25) for a simple problem of two examples and four hypotheses. This is analogous to Figure 3.4, which shows the objective function for ERLPBoost. Note that these figures are extremely similar. Because $\mathbf{d}$ is a vector of dimension two and the components must sum to one, $d_2 = 1 - d_1$. Consequently, this reduces to a two dimensional problem. The objective function is strictly convex, and it is easy to see that decreasing η increases the amount of regularization and therefore the curvature of the function. Similarly, Figure 3.6 illustrates the effect of the capping parameter ν, which constrains $d_1, d_2 \leq 1/\nu$. The corresponding plot for ERLPBoost is not interesting: changing ν does not change the value of the objective function, just the domain.

It is also possible to visualize Binary ERLPBoost in the $\mathbf{w}$ domain. Figure 3.7 plots the objective function of (3.26) for a simple problem of four examples and two hypotheses. As before, $\mathbf{w}$ is a vector of length two and the components must sum to one, so $w_2 = 1 - w_1$. Therefore we can again reduce this to a two dimensional problem. The objective function is strictly concave, and it is easy to see that decreasing η increases the amount of regularization and therefore the curvature of the function. Similarly, Figure 3.8 illustrates the effect of the capping parameter ν. However, the

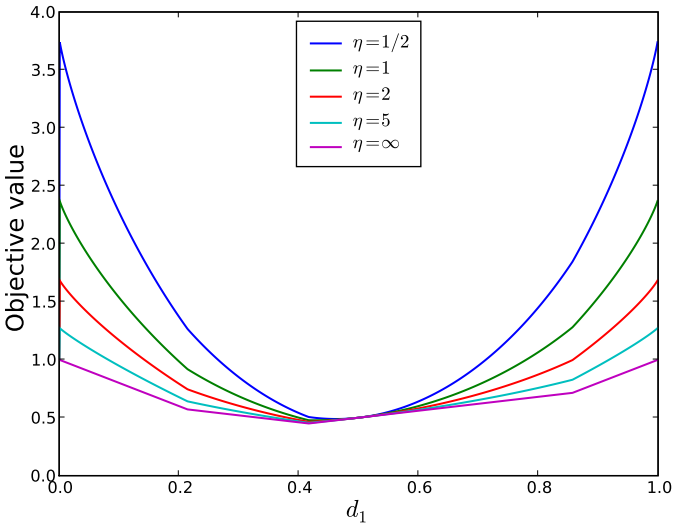


Figure 3.5: The effect of regularization on Binary ERLPBoost in the **d** domain. This function is piecewise continuous and it is not differentiable.

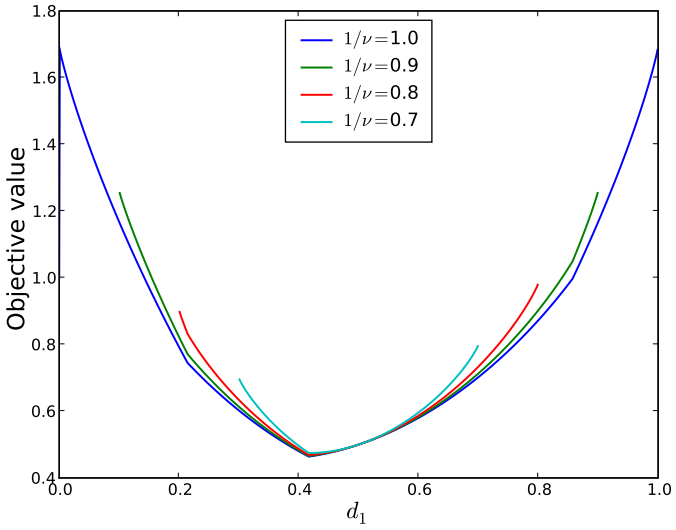


Figure 3.6: The effect of capping on Binary ERLPBoost in the **d** domain. This function is piecewise continuous and it is not differentiable.

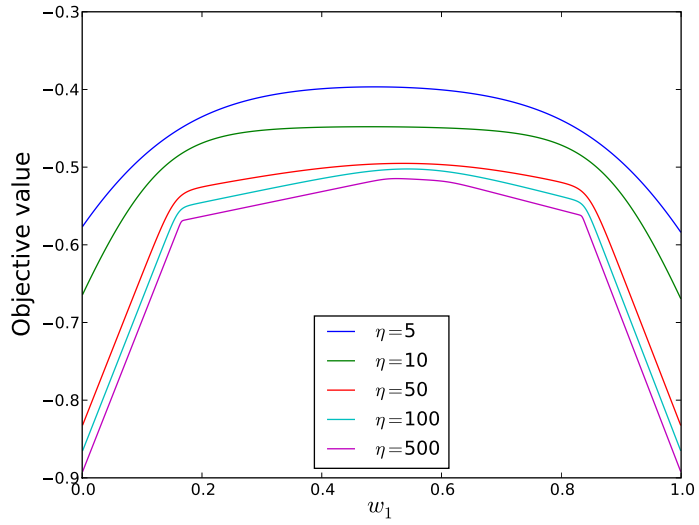


Figure 3.7: The effect of regularization on Binary ERLPBoost in the **w** domain for $\frac{1}{\nu} = 0.8$. This function is the Lagrangian dual of the function in the **d** domain. It is concave and differentiable.

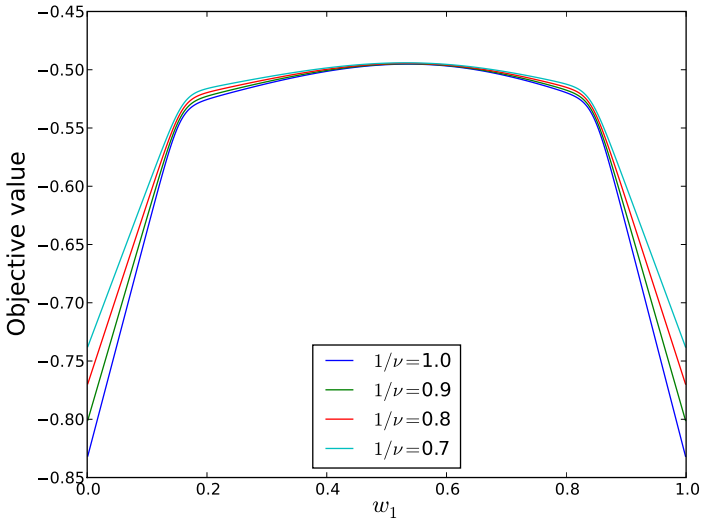


Figure 3.8: The effect of capping on Binary ERLPBoost in the **w** domain.This function is the Lagrangian dual of the function in the **d** domain. It is concave and differentiable.

effect of changing ν in the $\mathbf{w}$ domain is somewhat harder to interpret. The corresponding plot for ERLPBoost is not available because the $\boldsymbol{\psi}$ variables in the $\mathbf{w}$ domain make it impossible to visualize even the simplest problem in two dimensions. The fact that Binary ERLPBoost lacks $\boldsymbol{\psi}$ variables in the $\mathbf{w}$ domain – the Lagrange multipliers for the capping constraints – is the primary argument for using it.

3.3.4 Relationship Between Primal and Dual Variables for Binary ERLPBoost

Given the optimal $\mathbf{d}^t$ for Binary ERLPBoost, we now describe how to get $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$. The approach is the similar to the one derived for SoftBoost, ERLPBoost, and Unnormalized ERLPBoost. This approach relies on the active set method, covered in Appendix A, to eliminate the inequality constraints so that the KKT conditions can be solved as a linear system.

For the Binary ERLPBoost algorithm, the KKT conditions relating the primal and dual variables are:

$$\begin{aligned}
&\mathbf{d}^t \cdot \mathbf{u}^q - \gamma^t \leq 0, \;\; q = 1 \dots t && d_n^t \leq \tfrac{1}{\nu}, \;\; n = 1 \dots N \\
&w_q^t \geq 0, \;\; q = 1 \dots t && \psi_n^t \geq 0, \;\; n = 1 \dots N \\
&w_q^t(\mathbf{u}^q \cdot \mathbf{d}^t - \gamma^t) = 0, \;\; q = 1 \dots t && \psi_n^t(d_n^t - \tfrac{1}{\nu}) = 0, \;\; n = 1 \dots N \\
&\textstyle\sum_n d_n^t - 1 = 0, \;\; n = 1 \dots N && \textstyle\sum_{q=1}^t w_q^t = 1, \;\; q = 1 \dots t \\
&\frac{1}{\eta}\left[\ln\left(\frac{\mathbf{d}^t}{\mathbf{d}^0}\right) - \ln\left(\frac{\frac{1}{\nu} - \mathbf{d}^t}{\frac{1}{\nu} - \mathbf{d}^0}\right)\right] + \sum_{q=1}^t w_q^t \mathbf{u}^q + \sum_{n=1}^N \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0,
\end{aligned}$$

where $\mathbf{e}_n$ is a unit vector whose n^{th} component is 1. Note that the last two equations come from the condition that the gradient of the Lagrangian must be zero.

Because the optimal value of the primal variables, d^t and γ^t, is already known, we also know which constraints are active. Following the discussion in Appendix A and Chapter 3.1.2, we can use this knowledge to simplify the problem. For a given $\mathbf{d}$, we define the sets of active constraints for $\mathbf{w}$ as $\mathcal{A}_{\mathbf{w}}(d) := \{q \in 1 \dots t : \mathbf{u}^q \cdot \mathbf{d} = \gamma^t\}$ and $\boldsymbol{\psi}$ as $\mathcal{A}_{\boldsymbol{\psi}}(d) := \{n \in 1 \dots N : d_n = \frac{1}{\nu}\}$. Then the KKT optimality conditions for the Unnormalized ERLPBoost problem at iteration t can be simplified to

$$\sum_n d_n^t = 1$$

$$d_n^t = \tfrac{1}{\nu} \text{ for } n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)$$

$$\mathbf{d}^t \cdot \mathbf{u}^q = \gamma \text{ for } q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)$$

$$\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)}^t w_q^t = 1.$$

$$\tfrac{1}{\eta}\left[\ln\left(\frac{\mathbf{d}^t}{\mathbf{d}^0}\right) - \ln\left(\frac{\frac{1}{\nu}-\mathbf{d}^t}{\frac{1}{\nu}-\mathbf{d}^0}\right)\right] + \sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)} w_q \mathbf{u}^q + \sum_{n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)} \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0$$

The first three equations are solving for $\mathbf{d}^t$ and γ^t, but since these are known, these equations can be ignored. Therefore, solving for $\mathbf{w}^t$, $\boldsymbol{\psi}^t$, and β^t reduces to solving the linear system defined by

$$\sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)}^t w_q^t = 1.$$

$$\tfrac{1}{\eta}\left[\ln\left(\frac{\mathbf{d}^t}{\mathbf{d}^0}\right) - \ln\left(\frac{\frac{1}{\nu}-\mathbf{d}^t}{\frac{1}{\nu}-\mathbf{d}^0}\right)\right] + \sum_{q \in \mathcal{A}_{\mathbf{w}}(\mathbf{d}^t)} w_q \mathbf{u}^q + \sum_{n \in \mathcal{A}_{\boldsymbol{\psi}}(\mathbf{d}^t)} \psi_n^t \mathbf{e}_n + \beta^t \mathbf{1} = 0$$

The Binary ERLPBoost optimization problem is strictly convex and its feasible set is a closed convex set, so as long as the optimization problem is feasible, then it has a unique optimal solution [7]. As a result, the system of equations defined by the KKT conditions has an exact solution.

3.3.5 Iteration Bound for Binary ERLPBoost

We now bound the number of iterations T needed before the value of the Binary ERLPBoost optimization problem gets within ϵ of the guarantee g. This section contains two iteration bounds. One bound assumes that we use the simple stopping criterion and the other assumes that we use the practical stopping criterion.

We begin by bounding the increase $P_B^t(\mathbf{d}^t) - P_B^{t-1}(\mathbf{d}^{t-1})$ in successive iterations by a quadratic term. This lemma follows from Lemma 3.14, where we lower bounded the increase in the objective function for the unnormalized entropy by

$$\frac{1}{\eta}\left(1 - \sum_{n=1}^{N} d_n^{t-1} \exp(-\eta\alpha x_n)\right).$$

The same expression can be shown to lower bound the increase of the objective function for the optimization problem solved by Binary ERLPBoost. This result is used in the iteration bounds for both stopping criteria.

Lemma 3.19 *For Binary Entropy Regularized LPBoost, if $\eta \geq 1/2$, then*

$$P_B^t(\mathbf{d}^t) - P_B^{t-1}(\mathbf{d}^{t-1}) \geq \frac{1}{8\eta}(P_B^t(\mathbf{d}^{t-1}) - P_B^{t-1}(\mathbf{d}^{t-1}))^2.$$

Proof First observe that

$$P_B^t(\mathbf{d}^{t-1}) - P_B^{t-1}(\mathbf{d}^{t-1}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}^{t-1} - \max_{q=1,2,\ldots,t-1} \mathbf{u}^q \cdot \mathbf{d}^{t-1}.$$

Clearly the first max is at least as large as the second. If both are the same, then the lemma trivially holds because $P_B^t(\mathbf{d}^{t-1}) = P_B^{t-1}(\mathbf{d}^{t-1})$. If $P_B^t(\mathbf{d}^{t-1}) > P_B^{t-1}(\mathbf{d}^{t-1})$, then the first max equals $\mathbf{u}^t \cdot \mathbf{d}^{t-1}$. We can also rewrite the second max by invoking Lemma

3.13 with $t-1$ instead of t, obtaining

$$P_B^t(\mathbf{d}^{t-1}) - P_B^{t-1}(\mathbf{d}^{t-1}) = \mathbf{u}^t \cdot \mathbf{d}^{t-1} - \sum_{q=1}^{t-1} w_q^{t-1}\mathbf{u}^q \cdot \mathbf{d}^{t-1} := \underbrace{(\mathbf{u}^t - \sum_{q=1}^{t-1} w_q^{t-1}\mathbf{u}^q)}_{:=\mathbf{x}} \cdot \mathbf{d}^{t-1}.$$

We still need to show that when $\mathbf{x} \cdot \mathbf{d}^{t-1} \geq 0$, $P_B^t(\mathbf{d}^t) - P_B^{t-1}(\mathbf{d}^{t-1}) \geq \frac{1}{8\eta}(\mathbf{x} \cdot \mathbf{d}^{t-1})^2$.

By Lemma 3.18, the optimal value of the optimization problem defining Binary Entropy Regularized LPBoost, $P_B^t(\mathbf{d}^t)$, equals the optimal value of its dual problem. We begin by lower bounding the increase of this value between successive iterations. Let $\mathbf{w}^t$ and β^t denote optimal parameters for the dual problem at iteration t. Because the dual is a maximization problem, $\Theta_B^t(\mathbf{w}^t, \beta^t) \geq \Theta_B^t(\mathbf{w}, \beta)$ for any suboptimal $\mathbf{w} \in \mathcal{S}^t$ and $\beta \in \mathbb{R}$. For our lower bound on the value we replace β^t by the suboptimal previous value β^{t-1} and $\mathbf{w}^t$ by $\mathbf{w}^t(\alpha) = (1-\alpha)\begin{bmatrix} \mathbf{w}^{t-1} \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$, where $\alpha \in [0,1]$:

$$
\begin{aligned}
P_B^t(\mathbf{d}^t) - P_B^{t-1}(\mathbf{d}^{t-1}) =\ & \widehat{\Theta}_B^t(\mathbf{w}^t, \beta^t) - \widehat{\Theta}_B^{t-1}(\mathbf{w}^{t-1}, \beta^{t-1}) \\
\geq\ & \widehat{\Theta}_B^t(\mathbf{w}^t(\alpha), \beta^{t-1}) - \widehat{\Theta}_B^{t-1}(\mathbf{w}^{t-1}, \beta^{t-1}) \\
=\ & -\frac{1}{\nu\eta}\sum_{n=1}^{N} \ln\left(1 - \nu d_n^0 + \nu d_n^0 \exp\left(-\eta\sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta\alpha u_n^t + \eta\alpha\sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta\beta^{t-1}\right)\right) \\
& + \frac{1}{\nu\eta}\sum_{n=1}^{N} \ln\left(1 - \nu d_n^0 + \nu d_n^0 \exp\left(-\eta\sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta\beta^{t-1}\right)\right) \\
=\ & -\frac{1}{\nu\eta}\sum_{n=1}^{N} \ln\left(\frac{1 - \nu d_n^0}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta\sum_{q=1}^{t-1} u_n^q w_q^{t-1})}\right. \\
& \left. + \frac{\nu d_n^0 \exp\left(-\eta\sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta\alpha u_n^t + \eta\alpha\sum_{q=1}^{t-1} u_n^q w_q^{t-1}\right)}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta\sum_{q=1}^{t-1} u_n^q w_q^{t-1})}\right) \\
=\ & -\frac{1}{\nu\eta}\sum_{n=1}^{N} \ln\left(1 - \nu d_n^{t-1} + \nu d_n^{t-1}\exp(-\eta\alpha x_n)\right)
\end{aligned}
$$

Using the inequality $\ln(1+x) \leq x$, we get that

$$
-\frac{1}{\nu\eta}\sum_{n=1}^{N} \ln\left(1 - \nu d_n^{t-1} + \nu d_n^{t-1}\exp(-\eta\alpha x_n)\right) \geq \frac{1}{\eta}\sum_{n=1}^{N}\left(d_n^{t-1} - d_n^{t-1}\exp(-\eta\alpha x_n)\right)
$$

The r.h.s. of this last inequality is identical to Equation (3.22) in Lemma 3.14, and so this proof follows from that one. ■

We now prove iteration bounds for Binary ERLPBoost for the simple and the practical stopping criteria respectively. Since the recursion we proved in Lemma 3.19 for the Binary ERLPBoost is of the same form as the one proved in Lemma 3.7 for the ERLPBoost, all of the subsequent analysis is virtually identical. In fact, the only difference in the final bounds comes from the different value of regularization param-

eter η required to achieve the termination condition. Recall that for ERLPBoost, the canonical value of $\eta = \frac{2}{\epsilon} \ln \frac{N}{\nu}$ and for Binary ERLPBoost $\eta = \frac{2}{\epsilon}(\ln \frac{N}{\nu} + 1)$. Because the analysis is the same, we will give a complete statement of the following lemmas and theorems, but their proofs will be omitted.

We begin with the simple stopping criterion described in Chapter 3.3.1. Recall the definition of δ_B^t that is monitored by the simple stopping criterion:

$$\delta_B^t = \min_{q=1,2,\ldots,t} P_B^q(\mathbf{d}^{q-1}) - P_B^{t-1}(\mathbf{d}^{t-1}).$$

The following lemma gives a quadratic recurrence for δ_B^t that is identical to the one proved for ERLPBoost in Lemma 3.8.

Lemma 3.20 *If $\eta \geq 1/2$ and $\delta_B^t \geq 0$, then $\delta_B^t - \delta_B^{t+1} \geq \frac{(\delta_B^t)^2}{8\eta}$, for $1 \leq t \leq T$.*

By making use of Lemma 3.19, Lemma 3.20, and Lemma 3.9, we arrive at the final iteration bound for Binary ERLPBoost using the simple stopping criterion. The proof of this theorem is of the same form as Theorem 3.10.

Theorem 3.21 *If $\eta = \max(\frac{2}{\epsilon}(\ln \frac{N}{\nu} + 1), \frac{1}{2})$ in (3.25) and we use the simple stopping criterion of Chapter 3.3.1, then Binary ERLPBoost terminates in*

$$T \leq \max(\frac{32}{\epsilon^2}(\ln \frac{N}{\nu} + 1), \frac{8}{\epsilon})$$

iterations with a final convex combination of hypotheses for which $g - P_{LP}^t \leq \epsilon$.

We now prove a similar iteration bound for Binary ERLPBoost using the practical stopping criterion of Chapter 3.3.2. We will see that the bound we get with

the practical stopping criterion is looser than that of Theorem 3.21 by a factor of two. Moreover, it is identical to the bound in Theorem 3.10 for ERLPBoost with the practical stopping criterion. Recall that this stopping criterion monitors

$$\widetilde{\delta}_B^t := \min_{q=1,2,\ldots,t} P_B^q(\widetilde{\mathbf{d}}^{q-1}) - \widehat{\Theta}_B^{t-1}(\widetilde{\mathbf{w}}^{t-1}, \widetilde{\boldsymbol{\psi}}^{t-1}),$$

stops when $\widetilde{\delta}_B^{T+1} \leq \epsilon/2$ for a predefined threshold $\epsilon > 0$. However, the real quantity of interest is $\widehat{\delta}_B^t$, which is defined as

$$\widehat{\delta}_B^t := \min_{q=1,2,\ldots,t} P_B^q(\widetilde{\mathbf{d}}^{q-1}) - P_B^{t-1}(\mathbf{d}^{t-1}).$$

In the following lemma, we prove a quadratic recurrence for $\widehat{\delta}_B^t$.

Lemma 3.22 *If $\eta \geq 1/2$ and $\widehat{\delta}_B^t \geq 0$, then $\widehat{\delta}_B^t - \widehat{\delta}_B^{t+1} \geq \frac{(\widehat{\delta}_B^t)^2}{8\eta}$, for $1 \leq t \leq T$.*

The proof is virtually identical to that of Lemma 3.8. This lemma requires $\widehat{\delta}_B^t > 0$. According to Lemma 3.16, the practical stopping criterion ensures that $\widehat{\delta}_B^t > \epsilon/4$ for $t = 1 \ldots T$.

The following theorem contains the final iteration bound for Binary ERLPBoost using the practical stopping criterion. This theorem is a version of Theorem 3.21 that corresponds to the practical stopping criterion. The factor of two difference between the two bounds comes from the fact that for $t = 1 \ldots T$, $\delta_B^t > \epsilon/2$ while $\widehat{\delta}_B^t > \epsilon/4$.

Theorem 3.23 *If $\eta = \max(\frac{2}{\epsilon} \ln \frac{N}{\nu}, \frac{1}{2})$ in (3.6) and the practical stopping criterion from Chapter 3.3.2 is used, then Binary ERLPBoost terminates in*

$$T \leq \max(\frac{64}{\epsilon^2} \ln \frac{N}{\nu}, \frac{16}{\epsilon})$$

iterations with a final convex combination of hypotheses for which $g - P_{LP}^t \leq \epsilon$.

Chapter 4

Implementation

In this chapter, we discuss the implementation of the algorithms we use in our experiments. First we cover the ERLPBoost and Binary ERLPBoost algorithms. For these algorithms, the most difficult step is the optimization problem they solve at each iteration. We discuss these algorithms together because despite their differences, their optimization problems are solved in the same way. The next algorithm that we cover, originally proposed in [77], is extremely similar to ERLPBoost except that instead of updating the weights on all of the hypotheses at each iteration, it only updates the weight on the *last* hypothesis. We will refer to this algorithm as Corrective ERLPBoost. This algorithm uses relative entropy regularization, but it is easy to derive a similar algorithm that uses a binary entropy regularizer instead. We call this last algorithm Corrective Binary ERLPBoost.

ERLPBoost	
domain	optimization problem
$\mathbf{d}$	$\min_{\mathbf{d}} \quad \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0)$ $s.t. \quad \mathbf{d} \cdot \mathbf{1} = 1,\ \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$
$\mathbf{w}$	$\max_{\mathbf{w},\boldsymbol{\psi}} \quad \underbrace{-\frac{1}{\eta}\ln\sum_{n=1}^{N} d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \psi_n)) - \frac{1}{\nu}\sum_{n=1}^{N}\psi_n}_{\widehat{\Theta}^t(\mathbf{w},\boldsymbol{\psi})}$ $s.t. \quad \mathbf{w} \geq \mathbf{0},\ \ \mathbf{w} \cdot \mathbf{1} = 1,\ \ \boldsymbol{\psi} \geq \mathbf{0}$

Binary ERLPBoost	
domain	optimization problem
$\mathbf{d}$	$\min_{\mathbf{d}} \quad \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0)$ $s.t. \quad \mathbf{d} \cdot \mathbf{1} = 1,\ \mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$
$\mathbf{w}$	$\max_{\mathbf{w},\beta} \quad \underbrace{-\frac{1}{\eta\nu}\sum_{n=1}^{N}\ln(1-\nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))) - \beta}_{\widehat{\Theta}_B^T(\mathbf{w},\beta)}$ $s.t. \quad \mathbf{w} \geq \mathbf{0},\ \mathbf{w} \cdot \mathbf{1} = 1, \beta \in \mathbb{R}.$

Table 4.1: ERLPBoost and Binary ERLPBoost optimization problems in the $\mathbf{d}$ and $\mathbf{w}$ domains. The objective functions are not differentiable in the $\mathbf{d}$ domain, but they are in the $\mathbf{w}$ domain.

4.1 ERLPBoost and Binary ERLPBoost

The ERLPBoost algorithm is given in Algorithm 3.2 from Chapter 3.2 and the Binary ERLPBoost algorithm is given in Algorithm 3.3 from Chapter 3.3. In both algorithms, the most difficult step to implement is the optimization problem used to find updated $\mathbf{d}^t$. In this section we will discuss in detail how to solve this kind of optimization problem. First we will explain why we chose to work in the $\mathbf{w}$ domain. We then motivate LANCELOT [19], the augmented Lagrangian method that was used to solve the optimization problem[1]. Finally, we provide a detailed description of how we solve the bound-constrained sub-problem of the augmented Lagrangian method with the BLMVM[2] algorithm [5]. BLMVM requires the user to provide two things: an initial estimate of the variables and a user-defined function to compute the value and gradient of the augmented Lagrangian.

In the $\mathbf{d}$ domain, ERLPBoost solves the optimization problem defined in (3.6) and Binary ERLPBoost solves the problem defined in (3.25). For clarity, both of these problems are summarized in Table 4.1. Unfortunately, because of the max, neither of the objective functions is differentiable in the $\mathbf{d}$ domain. However, Table 4.1 also contains the optimization problems solved by ERLPBoost and Binary ERLPBoost in the $\mathbf{w}$ domain, which are the Lagrangian duals of the problems in the $\mathbf{d}$ domain. Notice that in the $\mathbf{w}$ domain, both objective functions are differentiable. Furthermore, as we discussed in Chapter 3, it is extremely easy to recover $\mathbf{d}^t$ given $\mathbf{w}^t$ and $\boldsymbol{\psi}^t$ for

[1] LANCELOT is an acronym for Large And Nonlinear Constrained Extended Lagrangian Optimization Techniques.

[2] BLMVM is an acronym for Bound-constrained Limited Memory Variable Metric

ERLPBoost. It is equally easy to recover $\mathbf{d}^t$ given $\mathbf{w}^t$ and β^t for Binary ERLPBoost. It therefore makes sense to solve the optimization problems in the $\mathbf{w}$ domain.

Both of these optimization problems have special structure that we can take advantage of. Note that the inequality constraints are all bounds on the variables. The only additional constraint is a single equality constraint, $\mathbf{w} \cdot \mathbf{1} = 1$. If the equality constraint were not there, this optimization problem would be a simple bound-constrained optimization problem. There are many algorithms that can solve this type of problem, for example LBFGS-B [13] and BLMVM [5].

To handle the equality constraint, the experiments in our thesis use the augmented Lagrangian method from the LANCELOT package [19, 58]. This algorithm is designed to solve problems of the form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ s.t. \quad & c(\mathbf{x}) = 0 \\ & l\mathbf{1} \leq \mathbf{x} \leq u\mathbf{1}. \end{aligned}$$

If we set $\mathbf{x} = [\mathbf{w}, \boldsymbol{\psi}]$, $c(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{1} - 1)$, and $f(\mathbf{x}) = \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ or $\widehat{\Theta}^t_B(\mathbf{w}, \boldsymbol{\psi})$, we can see that this problem has the same form as that of ERLPBoost and Binary ERLPBoost.

To motivate the augmented Lagrangian method, let us first consider the simple approach of incorporating the equality constraint into the objective function in the form of a penalty term. The resulting optimization problem is

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) + \frac{1}{2\mu} c(\mathbf{x})^2 \\ s.t. \quad & l\mathbf{1} \leq \mathbf{x} \leq u\mathbf{1}, \end{aligned}$$

where μ is a penalty parameter given by the user. Observe that a sufficiently small value of μ will push $c(\mathbf{x})$ toward zero and that this optimization problem can be solved with any bound-constrained optimization algorithm. The penalty method solves the above optimization problem with a decreasing sequence of μ values until the equality constraint is satisfied to sufficient precision. It has been established that $c_i(\mathbf{x}^t) \approx \lambda^* \mu^t$, where λ^* is the optimal Lagrange multiplier for the equality constraint while $\mathbf{x}^t$ and μ^t are the values of the variables at iteration t [58]. Unfortunately, the problem becomes ill conditioned when μ becomes too small, so we introduce an alternative method that does not have this problem.

The augmented Lagrangian method is related to the penalty method, but it reduces the possibility of ill conditioning by introducing Lagrange multiplier estimates. The augmented Lagrangian function is

$$L_A(\mathbf{x}, \lambda, \mu) := f(\mathbf{x}) - \lambda c(\mathbf{x}) + \frac{1}{2\mu} c(\mathbf{x})^2$$

and resulting optimization problem is

$$\begin{aligned} \min_x \ & L_A(\mathbf{x}, \lambda, \mu) \\ s.t. \ & l\mathbf{1} \leq \mathbf{x} \leq u\mathbf{1}, \end{aligned} \tag{4.1}$$

where λ is the Lagrange multiplier of the equality constraint and μ is the penalty parameter. Both of these values are given by the user, and the only variable being optimized is $\mathbf{x}$. This problem can be solved with a bound-constrained optimization solver as well. The first-order optimality conditions imply that $c_i(\mathbf{x}) \approx \mu^t(\lambda^* - \lambda^t)$, where λ^* is the optimal Lagrange multiplier and λ^t is its estimate at iteration t [58].

Thus, the augmented Lagrangian approach pushes $c_i(\mathbf{x})$ to zero more efficiently than the penalty method. Theorem 17.5 of [58] states that given λ^*, there exists a threshold value $\bar{\mu}$ such that for all $\mu \leq \bar{\mu}$, the variable x^* (the minimizer of the original optimization problem) is also the minimizer of (4.1).

We have therefore reduced the problem to finding sufficiently good estimates of λ and μ. The LANCELOT algorithm accomplishes this by solving (4.1) with a sequence of estimates of μ and λ until the equality constraint is satisfied to sufficient precision. The implementation follows the description in [19]. In particular, the initial parameter values are the ones described in [19]. The only deviation from the algorithm description occurs when we solve (4.1). Instead of the trust region method used by LANCELOT, we use the BLMVM algorithm from TAO [5].

We now discuss the use of BLMVM to minimize the bound-constrained augmented Lagrangian problem for ERLPBoost and Binary ERLPBoost. First, BLMVM requires an initial estimate of $\mathbf{w}$ and $\boldsymbol{\psi}$ for ERLPBoost and an initial estimate of $\mathbf{w}$ and β for Binary ERLPBoost. In the first iteration, we initialize $\mathbf{w}^0$ to uniform. For ERLPBoost we also initialize $\boldsymbol{\psi}^0 = \mathbf{0}$, while for Binary ERLPBoost we initialize $\beta^0 = 0$. In subsequent iterations, a common technique is to initialize a variable with its value from the previous iteration. At iteration t, we initialize $\boldsymbol{\psi}^t = \boldsymbol{\psi}^{t-1}$ for ERLPBoost and $\beta^t = \beta^{t-1}$ for Binary ERLPBoost. The difficulty is that the length of $\mathbf{w}$ increases by one at each iteration, so we cannot simply set $\mathbf{w}^t = \mathbf{w}^{t-1}$. In our implementation, we solve this problem by mimicking a single iteration of the corrective algorithm, which

will be defined in Algorithm 4.1 in the next section. We initialize

$$w_t^t = \max\left(0, \min\left(1, \frac{(\mathbf{u}^t - \sum_{q=0}^{t-1} w_q^{t-1}\mathbf{u}^q)\cdot \mathbf{d}^{t-1}}{\eta||\mathbf{u}^t - \sum_{q=0}^{t-1} w_q^t \mathbf{u}^q||_\infty^2}\right)\right),$$

and we set

$$w_q^{t-1} = (1 - w_t^t)w_q^{t-1} \text{ for } q = 1 \ldots t-1.$$

BLMVM also requires a user-defined function that computes the value of the augmented Lagrangian and and its gradient. We now explain how these values are computed, beginning with ERLPBoost. The value of the original objective function, $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ is in Table 4.1. The augmented Lagrangian value is

$$L_A(\mathbf{w}, \boldsymbol{\psi}, \lambda, \mu) = \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) + \lambda(\sum_{q=1}^{t} w_q - 1) + \frac{1}{2\mu}(\sum_{q=1}^{t} w_q - 1)^2.$$

The gradient of the augmented Lagrangian consists of its partial derivative w.r.t $\mathbf{w}$ and $\boldsymbol{\psi}$. The partial derivative of the augmented Lagrangian w.r.t. $\mathbf{w}$ is

$$\frac{\partial L_A(\mathbf{w}, \boldsymbol{\psi}, \lambda, \mu)}{\partial w_q} = \frac{\partial \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})}{\partial w_q} + \lambda + \frac{1}{\mu}(\sum_{q=1}^{t} w_q - 1),$$

where

$$\frac{\partial \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})}{\partial w_q} = \sum_{n=1}^{N} \frac{d_n^0 u_n^q \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \psi_n))}{\sum_{n'=1}^{N} d_{n'}^0 u_{n'}^q \exp(-\eta(\sum_{q=1}^{t} u_{n'}^q w_q + \psi_{n'}))}.$$

Since $\boldsymbol{\psi}$ does not appear in the equality constraint, the partial derivative of the augmented Lagrangian w.r.t. $\boldsymbol{\psi}$ is equal to

$$\frac{\partial L_A(\mathbf{w}, \boldsymbol{\psi}, \lambda, \mu)}{\partial \psi_n} = \frac{\partial \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})}{\partial \psi_n} = \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \psi_n))}{\sum_{n=1}^{N} d_n^0 u_n^q \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \psi_n))}.$$

We now derive the value of the augmented Lagrangian and its gradient for Binary ERLPBoost. The value of the original objective function, $\widehat{\Theta}_B^t(\mathbf{w}, \beta)$ is in Table 4.1.

The augmented Lagrangian value is

$$L_A(\mathbf{w}, \beta, \lambda, \mu) = \widehat{\Theta}_B^t(\mathbf{w}, \beta) + \lambda(\sum_{q=1}^{t} w_q - 1) + \frac{1}{2\mu}(\sum_{q=1}^{t} w_q - 1)^2.$$

The partial derivative of the augmented Lagrangian w.r.t. $\mathbf{w}$ is

$$\frac{\partial L_A(\mathbf{w}, \beta, \lambda, \mu)}{\partial w_q} = \frac{\partial \widehat{\Theta}_B^t(\mathbf{w}, \beta)}{\partial w_q} + \lambda + \frac{1}{\mu}(\sum_{q=1}^{t} w_q - 1),$$

where

$$\frac{\partial \widehat{\Theta}_B^t(\mathbf{w}, \beta)}{\partial w_q} = \sum_{n=1}^{N} \frac{d_n^0 u_n^q \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \beta))}{\sum_{n'=1}^{N} 1 - \nu d_{n'}^0 + \nu d_{n'}^0 \exp(-\eta(\sum_{q=1}^{t} u_{n'}^q w_q + \beta))}.$$

As before, since β does not appear in the equality constraint, the gradient of the augmented Lagrangian w.r.t. β is equal to

$$\frac{\partial L_A(\mathbf{w}, \beta, \lambda, \mu)}{\partial \beta} = \frac{\partial \widehat{\Theta}_B^t(\mathbf{w}, \beta)}{\partial \beta} = \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \beta))}{\sum_{n=1}^{N} 1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} u_n^q w_q + \beta))}.$$

Finally, the initial value of the variables, the value of the objective function, and the value of the gradient are given to BLMVM, which is documented in [5]. The remaining steps in ERLPBoost and Binary ERLPBoost are straightforward and do not require further documentation.

4.2 Corrective ERLPBoost

Shalev-Schwartz and Singer [77] proposed an algorithm that is similar to ERLPBoost, but it differs in two respects. First, instead of updating the weights on all of the hypotheses at each iteration, it only updates the weight on the *last* hypothesis

Algorithm 4.1 Corrective ERLPBoost (based on [77])

1. **Input:** $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, accuracy parameter $\epsilon > 0$, regularization parameter $\eta > 0$, and smoothness parameter $\nu \in [1, N]$.

2. **Initialize:** $\mathbf{d}^0$ to the uniform distribution,

   $\mathbf{w}^0 = \mathbf{0}$, $\boldsymbol{\psi}^0 = \mathbf{0}$, $P_C^1(\mathbf{d}^0) = 1$ and $\widehat{\Theta}_C^0(\mathbf{w}^0, \boldsymbol{\psi}^0) = -1$.

3. **Do for** $t = 1, \ldots$

   (a) Send $\mathbf{d}^{t-1}$ to oracle and obtain hypothesis h^t. Set $u_n^t = y_n h^t(\mathbf{x}_n)$.

   (b) Set $w_t^t = \max\left(0, \min\left(1, \frac{(\mathbf{u}^t - \sum_{q=0}^{t-1} w_q^{t-1}\mathbf{u}^q)\cdot\mathbf{d}^{t-1}}{\eta||\mathbf{u}^t - \sum_{q=0}^{t-1} w_q^t \mathbf{u}^q||_\infty^2}\right)\right)$.

   (c) Set $w_q^t = (1 - w_t^t) w_q^{t-1}$ for $q = 1 \ldots t-1$.

   (d) Set $\delta^t = \min_{q=1\ldots t} P^q(\mathbf{d}^{q-1}) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1})$,

   where $P^t(\mathbf{d}) = \sum_{q=1}^t w_q^t(\mathbf{u}^q \cdot \mathbf{d}) + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0)$ and

   $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) := -\frac{1}{\eta}\ln\sum_{n=1}^N d_n^0 \exp(-\eta(\sum_{q=1}^t u_n^q w_q + \psi_n)) - \frac{1}{\nu}\sum_{n=1}^N \psi_n$.

   (e) **If** $\delta^t \leq \epsilon/2$ **then** set $T = t-1$ and **break**.

   (f) Update the distribution to $d_n^t = d_n^0 \exp(-\eta \sum_{q=1}^t u_n^q w_q^t)$.

   (g) Project $\mathbf{d}^t$ using Algorithm 4.2 and define $C = \{n : d_n^t = \frac{1}{\nu}\}$.

   (h) Set $Z^t = \frac{1}{(1-|C|/\nu)} \sum_{n \notin C} d_n^0 \exp(-\eta \sum_{q=1}^t u_n^q w_q^t)$.

   (i) Set $\psi_n^t = \begin{cases} -\sum_{q=1}^t u_n^q w_q^t - \frac{1}{\eta}\ln\left(\frac{Z^t}{d_n^0 \nu}\right) & \text{if } n \in C \\ 0 & \text{if } n \notin C \end{cases}$.

4. **Output:** $f_{\mathbf{w}}(\mathbf{x}) = \sum_{q=1}^T w_q^T h^q(\mathbf{x})$.

Algorithm 4.2 Project Simplex [77]

1. **Input:** distribution $\mathbf{d} \in \mathcal{S}^N$ and capping parameter $\nu \in [1, N]$.
2. Sort $\mathbf{d}$ in descending order and store in $\mathbf{d}'$.
3. **Initialize:** $Z = \sum_{n=1}^{N} d'_n$
4. **Do for** $n = 1, \ldots N$
   (a) $\theta = \frac{1-\nu(n-1)}{Z}$
   (b) **If** $\theta d'_n \leq \nu$ **then** break
   (c) $Z = Z - d'_n$
5. **Output:** $\hat{\mathbf{d}}$ s.t. $\hat{d_n} = \min\{\nu, \theta d_n\}$.

and rescales the weights on the previous hypotheses accordingly. Also, instead of updating the weights on the examples and the hypotheses at the same time, this algorithm updates them separately. We call this algorithm Corrective ERLPBoost, and its details are given in Algorithm 4.1. In this section, we first explain the motivation for this algorithm in terms of its totally corrective counterpart, ERLPBoost. We then present a stopping criterion for Corrective ERLPBoost. This is the main difference between our implementation and the original algorithm description in [77], which does not contain a stopping criterion. Finally, we present an optimization problem that simultaneously updates $\mathbf{d}$ and $\mathbf{w}$ and whose update is equivalent to the update in Algorithm 4.1.

Before we proceed, let us clarify the notation we will use in the following two

sections. We refer to the updated weight on hypothesis t at iteration t as w_t^t, and if the value is not updated, we call this weight w_t. Next, observe that in Algorithm 4.1, Corrective ERLPBoost first chooses w_t^t and then sets $w_q^t = (1-w_t^t)w_q^{t-1}$ for $q = 1 \ldots t-1$. These values define vector $\mathbf{w}^t$.

To understand how Corrective ERLPBoost and ERLPBoost are related, we analyze the way Corrective ERLPBoost chooses the updated weight $\mathbf{w}_t^t$ for hypothesis t. This choice is motivated by Lemma 3.7, which lower bounds the increase of the ERLPBoost objective function at each iteration via the Lagrangian dual. Recall that because the ERLPBoost dual is a maximization problem, $\widehat{\Theta}^t(\mathbf{w}^t, \boldsymbol{\psi}^t) \geq \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ for any suboptimal $\mathbf{w} \in \mathcal{S}^t$ and $\boldsymbol{\psi} \geq \mathbf{0}$. For our lower bound on the value we replaced $\boldsymbol{\psi}^t$ by the suboptimal previous value $\boldsymbol{\psi}^{t-1}$ and $\mathbf{w}^t$ by

$$\mathbf{w}^t(\alpha) = (1-\alpha)\begin{bmatrix} \mathbf{w}^{t-1} \\ \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{0} \\ \\ 1 \end{bmatrix}, \text{ where } \alpha \in [0,1].$$

This substitution results in the following lower bound from Lemma 3.7:

$$\begin{aligned}
P^t(\mathbf{d}^t) \quad - P^{t-1}(\mathbf{d}^{t-1}) = \; & \widehat{\Theta}^t(\mathbf{w}^t, \boldsymbol{\psi}^t) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1}) \\
\geq \quad & \widehat{\Theta}^t(\mathbf{w}^t(\alpha), \boldsymbol{\psi}^{t-1}) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1}) \\
= \quad & -\tfrac{1}{\eta} \ln \textstyle\sum_{n=1}^N d_n^{t-1} \exp\left(-\eta\alpha\left(u_n^t - \sum_{q=1}^{t-1} u_n^q w_q^{t-1}\right)\right) \qquad (4.2) \\
\geq \quad & \alpha(\mathbf{u}^t - \textstyle\sum_{q=1}^{t-1} \mathbf{u}^q w_q^{t-1}) \cdot \mathbf{d} - 2\eta\alpha^2. \qquad (4.3)
\end{aligned}$$

The best possible choice that Corrective ERLPBoost could make for w_t^t is the value of α that maximizes (4.2), but there is no closed form solution for α. Instead, Corrective ERLPBoost chooses α to maximize a lower bound on this expression. Observe that

tighter lower bound means more progress per iteration, which results in faster convergence. A naive choice for α would be the maximizer of (4.3), but we will show that a better choice of α is found in[77].

In the analysis of [77], (4.3) is replaced by

$$\alpha(\mathbf{u}^t - \sum_{q=1}^{t-1} \mathbf{u}^q w_q^{t-1}) \cdot \mathbf{d} - 2\eta\alpha^2 ||\mathbf{u}^t - \sum_{q=1}^{t-1} \mathbf{u}^q w_q^{t-1}||_\infty^2. \tag{4.4}$$

Corrective ERLPBoost sets w_t^t to the value of α that maximizes the above expression because as Figure 4.1 shows, this is a tighter lower bound on (4.2). Figure 4.1 compares the value of Equation (4.2) with its two lower bounds shown in (4.3) and (4.4) respectively. This figure is for a simple problem of two examples and two hypotheses in the uncapped case. We must use the uncapped case because otherwise, the $\boldsymbol{\psi}$ variables would make it impossible to visualize the problem in two dimensions. According to Figure 4.1, neither lower bound is tight. However, the lower bound of (4.4), which is used by Corrective ERLPBoost, is tighter than the lower bound of (4.3), which is used in Lemma 3.7.

A potential advantage of the corrective algorithm is that the update on the weights performed at each iteration has a complexity of $O(N \ln(N))$, which is much faster than solving a complicated constrained optimization problem. Note that it is the projection step that is $O(N \ln(N))$, and the update could be done in linear time by using the projection algorithm in Herbster and Warmuth [38]. While the corrective algorithm is always simpler and faster on an iteration by iteration basis, it requires more iterations than its totally corrective counterpart. Despite this, Corrective ERLPBoost

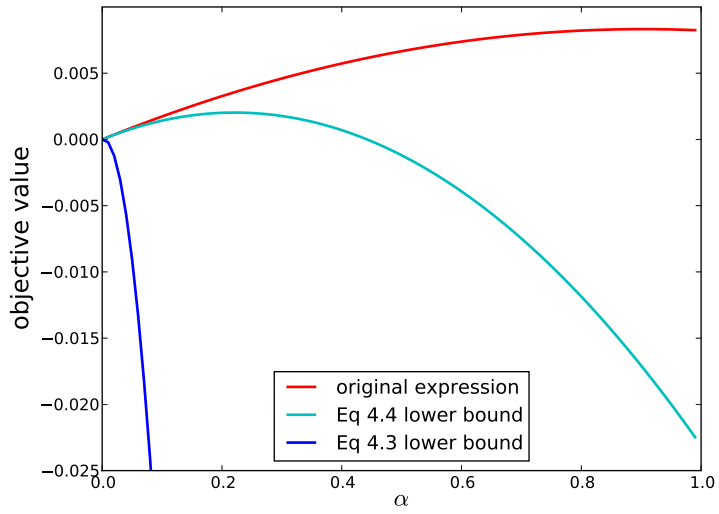


Figure 4.1: The original $\widehat{\Theta}^t(\mathbf{w}^t(\alpha), \boldsymbol{\psi}^{t-1}) - \widehat{\Theta}^{t-1}(\mathbf{w}^{t-1}, \boldsymbol{\psi}^{t-1})$ is plotted against the lower bounds in (4.4) from [77] and (4.3) from Lemma 3.7. The lower bound in (4.4) is tighter, but it is not optimal.

has an iteration bound of $O(\frac{1}{\epsilon^2} \ln \frac{N}{\nu})$, the same iteration bound as ERLPBoost. This is possible because the iteration bound is loose.

4.2.1 Stopping Criterion for Corrective ERLPBoost

Our implementation of Corrective ERLPBoost, shown in Algorithm 4.1, differs from the algorithm proposed in [77] in only one respect: the stopping criterion. The algorithm in [77] has no stopping criterion. Instead, it was proved that after $\Omega(\frac{1}{\epsilon^2} \ln(\frac{N}{\nu}))$, then $P_{LP}^t - \widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi}) \leq \epsilon$, where $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ (defined in (3.10)) is the Lagrangian dual of the ERLPBoost objective function $P^t(\mathbf{d})$. Observe that in the analysis of the corrective algorithm, the convergence is measured using the totally corrective optimization problem.

Unfortunately, $\Omega(\frac{1}{\epsilon^2}\ln(\frac{N}{\nu}))$ is very large number of iterations, and it is likely that the algorithm will require far fewer iterations to reach a precision of ϵ. For this reason, we chose to modify the algorithm defined in [77] to use the practical ERLP-Boost stopping criterion defined in Chapter 3.2.2. Corrective ERLPBoost monitors the quantity

$$\widetilde{\delta}^t = \min_{q=1\ldots t} P^q(\mathbf{d}^{q-1}) - \widehat{\Theta}^{t-1}(w_{t-1}^{t-1}, \boldsymbol{\psi}^{t-1}),$$

where $P^t(\mathbf{d})$ is given in (3.6) and $\widehat{\Theta}^t(\mathbf{w}, \boldsymbol{\psi})$ is given in (3.10). The algorithm terminates when when $\widetilde{\delta}^t \leq \frac{\epsilon}{2}$.

Every variable in the above expression is known except $\boldsymbol{\psi}$. This is because in the projection step, Algorithm 4.2 is given the unnormalized distribution whose components are $d_n^0 \exp(-\eta \sum_{q=1}^t w_q u_n^q)$ and returns a distribution whose components are in the capped simplex, but $\boldsymbol{\psi}$ is never computed in the process. We will now get a closed form expression for the ψ_n^t corresponding to the projected distribution. This was originally derived by Vishwanathan [85].

After setting $w_q^t = (1-w_t^t)w_q^{t-1}$ for $q = 1\ldots t-1$, the update $\mathbf{d}$ on the examples becomes

$$d_n^t = \underbrace{\frac{d_n^0 \exp(-\eta \sum_{q=1}^t u_n^q w_q^t - \eta\psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\eta \sum_{q=1}^t u_n^q w_q^t - \eta\psi_n^t)}}_{Z^t}. \tag{4.5}$$

The KKT conditions imply that the product $\psi_n^t(d_n^t - \frac{1}{\nu}) = 0$. This means that either $\psi_n^t = 0$ and $d_n^t < \frac{1}{\nu}$ or $d_n^t = \frac{1}{\nu}$ and $\psi_n^t \geq 0$. Let us define the set of indices of active

constraints $C = \{n : d_n^t = \frac{1}{\nu}\}$. We know that $\sum_{n=1}^{N} d_n = 1$, so

$$\sum_{n \notin C} d_n^t = 1 - \frac{1}{\nu}|C|.$$

Since $\psi_n^t = 0$ for $n \notin C$,

$$\sum_{n \notin C} d_n^t = \sum_{n \notin C} \frac{d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q^t - \eta \psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q^t - \eta \psi_n^t)}$$

The above expression can be computed from quantities that are already known. Combining the previous two expressions yields

$$Z^t = \frac{\sum_{n \notin C} d_n^t}{1 - \frac{1}{\nu}|C|}.$$

Once we have Z^t, getting ψ_n^t is straightforward. If $n \notin C$, $\psi_n^t = 0$. If $n \in C$, then by using the form of the update in (4.5) along with the fact that $d_n^t = \frac{1}{\nu}$, it is clear that ψ_n^t satisfies

$$d_n^0 \exp(-\eta \sum_{q=1}^{t} u_n^q w_q^t - \eta \psi_n^t) = \frac{Z^t}{\nu}.$$

Solving for ψ_n^t gives us

$$\psi_n^t = -\sum_{q=1}^{t} u_n^q w_q^t - \frac{1}{\eta} \ln(\frac{Z^t}{d_n^0 \nu}).$$

4.2.2 Alternative Corrective ERLPBoost Optimization Problem

Recall from Algorithm 4.1 that Corrective ERLPBoost updates the $\mathbf{d}$ and w_t variables separately. We now present an optimization problem that optimizes $\mathbf{d}$ and w_t jointly and whose optimal value $\mathbf{d}^t$ is of the exact same form as (4.5). The optimal w_t^t is the dual variable to an equality constraint; it is not determined heuristically as it was in Algorithm 4.1.

At iteration t, we define the following optimization problem:

$$\begin{aligned} \min \quad & \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0). \\ \mathbf{d} \cdot \mathbf{1} &= 1 \\ \mathbf{d} &\leq \tfrac{1}{\nu}\mathbf{1} \\ (\mathbf{u}^t - \textstyle\sum_{q=1}^{t-1} w_q^{t-1}\mathbf{u}^q) \cdot \mathbf{d} &= 0 \end{aligned} \tag{4.6}$$

The last constraint ensures that the edge w.r.t. the latest hypothesis is equal to the edge w.r.t. the convex combination of the previous hypotheses. It is instructive to compare this optimization problem with that of ERLPBoost. Recall from Chapter 3.2 that the optimization problem solved by ERLPBoost at each iteration is

$$\begin{aligned} \min \quad & \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0). \\ \mathbf{d} \cdot \mathbf{1} &= 1 \\ \mathbf{d} &\leq \tfrac{1}{\nu}\mathbf{1} \end{aligned}$$

Notice the difference in the first terms of their respective objective functions. The corrective algorithm minimizes $(\mathbf{u}^t \cdot \mathbf{d})$, while the totally corrective algorithm minimizes $\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}$.

We now derive the Lagrangian dual of (4.6), and in the process, we show that the form of the update on $\mathbf{d}$ is equivalent to (4.5).

Lemma 4.1 *The Lagrangian dual of* (4.6) *is*

$$\max_{w_t, \boldsymbol{\psi}} \ \widehat{\Theta}_C^t(w_t, \boldsymbol{\psi}) \ \textit{s.t.} \quad \boldsymbol{\psi} \geq 0, \ \textit{where} \tag{4.7}$$

$$\widehat{\Theta}_C^t(w_t, \boldsymbol{\psi}) := -\frac{1}{\eta} \ln \sum_{n=1}^{N} d_n^0 \exp(-\eta(1 - w_t) \sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta w_t u_n^t - \eta\psi_n)) - \frac{1}{\nu} \sum_{n=1}^{N} \psi_n.$$

The optimal solution $\mathbf{d}^t$ *of* (4.6) *can be expressed in terms of the dual variables* w_t^t *and* $\boldsymbol{\psi}^t$ *as follows:*

$$d_n^t := \frac{d_n^0 \exp(-\eta(1-w_t^t)\sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta w_t^t u_n^t - \eta\psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\eta(1-w_t^t)\sum_{q=1}^{t-1} u_n^q w_q^{t-1} - \eta w_t^t u_n^t - \eta\psi_n^t))}. \tag{4.8}$$

Furthermore, when we introduce the change of variable $w_q^t = (1-w_t^t)w_q^{t-1}$*, the optimal solution* $\mathbf{d}^t$ *becomes*

$$d_n^t := \frac{d_n^0 \exp(-\eta\sum_{q=1}^{t} u_n^q w_q^t - \eta\psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\eta\sum_{q=1}^{t} u_n^q w_q^t - \eta\psi_n^t))}.$$

Proof The Lagrangian for this minimization problem in (4.6) is

$$\begin{aligned} L^t(\underbrace{\mathbf{d}}_{\text{primal}}, \underbrace{\alpha, \boldsymbol{\psi}, \beta}_{\text{dual}}) = \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) + \alpha(\sum_{q=1}^{t-1} w_q(\mathbf{u}^q \cdot \mathbf{d}^t) - \mathbf{u}^t \cdot \mathbf{d}^t) \\ + \sum_{n=1}^{N} \psi_n(d_n - 1/\nu) + \beta(\mathbf{1}\cdot\mathbf{d} - 1). \end{aligned}$$

To get the form of the update to match that of [77], let us do a change of variable and substitute $w_t = 1-\alpha$. The new Lagrangian is

$$\begin{aligned} L^t(\underbrace{\mathbf{d}}_{\text{primal}}, \underbrace{w_t, \boldsymbol{\psi}, \beta}_{\text{dual}}) = \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta}\Delta(\mathbf{d}, \mathbf{d}^0) + (1-w_t)(\sum_{q=1}^{t-1} w_q(\mathbf{u}^q \cdot \mathbf{d}^t) - \mathbf{u}^t \cdot \mathbf{d}^t) \\ + \sum_{n=1}^{N} \psi_n(d_n - 1/\nu) + \beta(\mathbf{1}\cdot\mathbf{d} - 1). \end{aligned} \tag{4.9}$$

The dual is derived from the Lagrangian by plugging in the minimum value of the primal variables:

$$\Theta_C^t(w_t, \boldsymbol{\psi}, \beta) := \inf_{\mathbf{d}} L^t(\mathbf{d}, w_t, \boldsymbol{\psi}, \beta).$$

Differentiating the Lagrangian w.r.t. $\mathbf{d}$ shows that the n-th component of the optimal

d vector has the form

$$d_n^t = d_n^0 \exp(-\eta(1-w_t)\sum_{q=1}^{t-1} u_n^q w_q - \eta w_t u_n^t - \eta\psi_n - \eta\beta - 1). \tag{4.10}$$

By plugging the optimal **d** into the Lagrangian (4.9), the dual function simplifies to

$$\Theta_C^t(w_t, \boldsymbol{\psi}, \beta) = -\frac{1}{\eta}\sum_{n=1}^{N} d_n^0 \exp(-\eta(1-w_t)\sum_{q=1}^{t-1} u_n^q w_q - \eta w_t u_n^t - \eta\psi_n - \eta\beta - 1) - \beta - \frac{1}{\nu}\sum_{n=1}^{N}\psi_n.$$

This results in the following Lagrange dual:

$$\max_{\boldsymbol{\psi},\beta} \ \Theta_C^t(w_t, \boldsymbol{\psi}, \beta)$$

$$\text{s.t. } \boldsymbol{\psi} \geq 0,$$

By differentiating $\Theta_C^t(w_t, \boldsymbol{\psi}, \beta)$ we can determine the optimal choice of β:

$$\beta^t(w_t, \boldsymbol{\psi}) = -\frac{1}{\eta} + \frac{1}{\eta}\ln\sum_{n=1}^{N} d_n^0 \exp(-\eta(1-w_t)\sum_{q=1}^{t-1} u_n^q w_q - \eta w_t u_n^t - \eta\psi_n).$$

Plugging this choice for β into (4.10) results in

$$\begin{aligned} d_n^t :=& d_n^t(w_t, \boldsymbol{\psi}^t, \beta^t(\boldsymbol{\psi}^t)) \\ =& \frac{d_n^0 \exp(-\eta(1-w_t)\sum_{q=1}^{t-1} u_n^q w_q - \eta w_t u_n^t - \eta\psi_n^t)}{\sum_{n'} d_{n'}^0 \exp(-\eta(1-w_t)\sum_{q=1}^{t-1} u_n^q w_q - \eta w_t u_n^t - \eta\psi_n^t)}. \end{aligned}$$

Once β is optimized, the Lagrangian becomes

$$\begin{aligned} &\Theta_C^t(w_t, \boldsymbol{\psi}, \beta^t(\boldsymbol{\psi})) = \widehat{\Theta}_C^t(w_t, \boldsymbol{\psi}) \\ &:= -\frac{1}{\eta}\ln\sum_{n=1}^{N} d_n^0 \exp(-\eta(1-w_t)\sum_{q=1}^{t-1} u_n^q w_q - \eta w_t u_n^t - \eta\psi_n) - \frac{1}{\nu}\sum_{n=1}^{N}\psi_n. \end{aligned}$$

The dual problem now reduces to

$$\max_{w_t,\boldsymbol{\psi}} \quad \widehat{\Theta}_C^t(w_t, \boldsymbol{\psi})$$

$$\text{s.t.} \quad \boldsymbol{\psi} \geq 0.$$

The primal objective is convex and the primal constraints are affine. Also, the uniform distribution is always a feasible solution. Therefore, Slater's condition tells us that since this problem has a non-empty feasible set, strong duality holds and the values of the primal and the dual problems are the same.

■

4.3 Corrective Binary ERLPBoost

The algorithm defined in [77] uses a relative entropy regularizer, but it is easy to derive a similar algorithm that uses a binary entropy regularizer. We call this algorithm Corrective Binary ERLPBoost, and its details are given in Algorithm 4.3. In this section, we first explain the motivation for this algorithm in terms of its totally corrective counterpart, Binary ERLPBoost. We then discuss the stopping criterion and the computation of β, the variable that ensures that the updated $\mathbf{d}^t$ is normalized. We will see that the two primary differences between Corrective ERLPBoost and Corrective Binary ERLPBoost are the update on the weights and the normalization mechanism. Finally, we present an optimization problem that updates the weights on the examples and the hypotheses jointly and whose update has the same form as that of Algorithm 4.3.

Corrective Binary ERLPBoost approximates Binary ERLPBoost in the same way that the algorithm in the previous section approximates ERLPBoost. This algorithm finds an updated w_t^t based only on the current hypothesis, whereas Binary

ERLPBoost updates the entire $\mathbf{w}$ vector based on all previous hypotheses. The choice of w_t^t is motivated by Lemma 3.19, which lower bounds the increase of the Binary ERLPBoost objective function at each iteration via the Lagrangian dual. Because the Binary ERLPBoost dual is a maximization problem, $\widehat{\Theta}_B^t(\mathbf{w}^t, \beta^t) \geq \widehat{\Theta}_B^t(\mathbf{w}, \beta)$ for any suboptimal $\mathbf{w} \in \mathcal{S}^t$ and $\beta \in \mathbb{R}$. For our lower bound, we replaced β^t by the suboptimal previous value β^{t-1} and $\mathbf{w}^t$ by $\mathbf{w}^t(\alpha) = (1-\alpha)\begin{bmatrix} \mathbf{w}^{t-1} \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$, where $\alpha \in [0,1]$. This substitution results in the following lower bound:

$$\begin{aligned} P_B^t(\mathbf{d}^t) \quad & -P_B^{t-1}(\mathbf{d}^{t-1}) = \widehat{\Theta}_B^t(\mathbf{w}^t, \beta^t) - \widehat{\Theta}_B^{t-1}(\mathbf{w}^{t-1}, \beta^{t-1}) \\ & \geq \widehat{\Theta}_B^t(\mathbf{w}^t(\alpha), \beta^{t-1}) - \widehat{\Theta}_B^{t-1}(\mathbf{w}^{t-1}, \beta^{t-1}) \\ & = -\tfrac{1}{\nu\eta} \textstyle\sum_{n=1}^N \ln\left(1 - \nu d_n^{t-1} + \nu d_n^{t-1} \exp(-\eta\alpha(u_n^t - \sum_{q=1}^{t-1} u_n^q w_q^{t-1}))\right) \\ \geq \quad & \alpha(\mathbf{u}^t - \textstyle\sum_{q=1}^{t-1} \mathbf{u}^q w_q^{t-1}) \cdot \mathbf{d} - 2\eta\alpha^2. \end{aligned}$$

Note that the lower bound for Binary ERLPBoost is identical to that of ERLPBoost.

The updated w_t for Corrective Binary ERLPBoost is based on the same substitution of suboptimal variables. Recall from the previous section that the analysis of [77] uses a tighter lower bound, (4.4). The same lower bound holds in the case of binary entropy regularization, so we set w_t^t to the value of α that maximizes (4.4) in the interval $[0,1]$:

$$w_t^t = \max\left(0, \min\left(1, \frac{(\mathbf{u}^t - \sum_{q=0}^{t-1} w_q^{t-1}\mathbf{u}^q) \cdot \mathbf{d}^{t-1}}{\eta||\mathbf{u}^t - \sum_{q=0}^{t-1} w_q^t \mathbf{u}^q||_\infty^2}\right)\right).$$

Before continuing, it is worth noting that Corrective Binary ERLPBoost does not project $\mathbf{d}^t$ into the capped simplex in the same way that Corrective ERLPBoost does. Instead, the capping is handled implicitly by the binary relative entropy and

normalization is achieved by finding the right value of β via binary search, as shown in Algorithm 4.4.

4.3.1 Stopping Criterion for Corrective Binary ERLPBoost

Recall that in the analysis of [77], the convergence of the corrective algorithm is measured using the totally corrective optimization problem. We therefore chose to modify Corrective Binary ERLPBoost to use the practical stopping criterion of Binary ERLPBoost, described in Chapter 3.3.2. Corrective Binary ERLPBoost monitors the quantity

$$\widetilde{\delta}_B^t = \min_{q=1\ldots t} \mathbf{P}^q(\mathbf{d}^{q-1}) - \widehat{\Theta}_B^{t-1}(\mathbf{w}^{t-1}, \beta^{t-1}),$$

where

$$\mathbf{P}^t(\mathbf{d}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0)$$

and

$$\widehat{\Theta}_B^t(\mathbf{w}, \beta) := -\frac{1}{\eta\nu}\sum_{n=1}^{N} \ln(1 - \nu\mathbf{d}_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))) - \beta.$$

The algorithm terminates when $\widetilde{\delta}_B^t \leq \frac{\epsilon}{2}$.

Every variable in the above expression is known except β, for which there is no closed form expression. Instead, we find β via binary search in Algorithm 4.4. In contrast, for Corrective ERLPBoost we were able to find closed-form expression for the $\boldsymbol{\psi}$ variables. The binary search is computationally expensive, so for this reason we conjecture that Corrective Binary ERLPBoost will be slower than Corrective ERLPBoost.

Algorithm 4.3 Corrective Binary ERLPBoost

1. **Input:** $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, accuracy parameter $\epsilon > 0$, smoothness parameter $\nu \in [1, N]$, and regularization parameter $\eta > 0$. The canonical value of $\eta = \frac{2}{\epsilon}(\ln \frac{N}{\nu} + 1)$.

2. **Initialize:** $\mathbf{d}^0$ to the uniform distribution,

   $\mathbf{w}^0 = \mathbf{0}$, $\beta^0 = 0$, $P^1_{CB}(\mathbf{d}^0) = 1$ and $\widehat{\Theta}^0_{CB}(\mathbf{w}^0, \beta^0) = -1$.

3. **Do for** $t = 1, \ldots$

   (a) Send $\mathbf{d}^{t-1}$ to oracle and obtain hypothesis h^t. Set $u^t_n = y_n h^t(\mathbf{x}_n)$.

   (b) Set $w^t_t = \max\left(0, \min\left(1, \frac{(\mathbf{u}^t - \sum_{q=0}^{t-1} w^{t-1}_q \mathbf{u}^q) \cdot \mathbf{d}^{t-1}}{\eta ||\mathbf{u}^t - \sum_{q=0}^{t-1} w^t_q \mathbf{u}^q||^2_\infty}\right)\right)$.

   (c) Set $w^t_q = (1 - w^t_t) w^{t-1}_q$ for $q = 1 \ldots t-1$.

   (d) Set $\widetilde{\delta}^t_B = \min_{q=1\ldots t} \mathbf{P}^q(\mathbf{d}^{q-1}) - \widehat{\Theta}^{t-1}_B(\mathbf{w}^{t-1}, \beta^{t-1})$,

   where $\mathbf{P}^t(\mathbf{d}) = \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0)$

   and $\widehat{\Theta}^t_B(\mathbf{w}, \beta) := -\frac{1}{\eta\nu} \sum_{n=1}^N \ln(1 - \nu\mathbf{d}^0_n + \nu d^0_n \exp(-\eta(\sum_{q=1}^t w_q u^q_n + \beta))) - \beta$.

   (e) **If** $\delta^t_{CB} \leq \epsilon/2$ **then** set $T = t - 1$ and **break**.

   (f) Find β via Algorithm 4.4

   (g) Update the distribution to

   $$d^t_n := \frac{d^0_n \exp(-\eta(\sum_{q=1}^t w^t_q u^q_n + \beta))}{1 - \nu d^0_n + \nu d^0_n \exp(-\eta(\sum_{q=1}^t w^t_q u^q_n + \beta))}. \tag{4.11}$$

4. **Output:** $f_{\mathbf{w}}(\mathbf{x}) = \sum_{q=1}^T w^T_q h^q(\mathbf{x})$.

Algorithm 4.4 Binary Search for β

1. **Input:** $\mathbf{w}$, ν, η, $\mathbf{u}^q$ for $q = 1 \dots t$, tolerance τ,

   and maximum number of bisections T.

2. **Initialize:** $\beta_l < 0 < \beta_u$.

3. **Define:** $s(\beta) := \sum_{n=1}^{N} \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q u_n^q + \beta))}$

4. **While** $s(\beta_l) \geq 1$, set $\beta_l = 2\beta_l$.

5. **While** $s(\beta_u) \leq 1$, set $\beta_u = 2\beta_u$.

6. **Do for** $t = 1, \dots T$

   (a) $\beta = \beta_l + \frac{1}{2}(\beta_l + \beta_u)$

   (b) **If** $|s(\beta)| \leq \tau$ **then** break

   (c) **If** $s(\beta) < 1$ **then** set $\beta_u = \beta$, **Else** set $\beta_l = \beta$

7. **Output:** β

4.3.2 Alternative Corrective Binary ERLPBoost Optimization Problem

Algorithm 4.3 shows that Corrective Binary ERLPBoost updates the $\mathbf{d}$ and w_t variables separately. We now present an optimization problem that optimizes $\mathbf{d}$ and w_t simultaneously and whose update on $\mathbf{d}^t$ is equivalent to (4.11) in Algorithm 4.3. Note that in this optimization problem, w_t is the dual variable to an equality constraint; it is not determined heuristically as it was in Algorithm 4.3.

At iteration t, we define the following optimization problem:

$$\begin{aligned} \min \quad & \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0). \\ \mathbf{d} \cdot \mathbf{1} &= 1 \\ (\mathbf{u}^t - \textstyle\sum_{q=1}^{t-1} w_q \mathbf{u}^q) \cdot \mathbf{d} &= 0 \end{aligned} \tag{4.12}$$

The last constraint balances the edge w.r.t. the latest hypothesis and the edge w.r.t. the convex combination of the previous hypotheses. It is instructive to compare this optimization problem with that of Binary ERLPBoost. Recall from Chapter 3.3 that the at each iteration Binary ERLPBoost solves the following optimization problem:

$$\begin{aligned} \min \quad & \max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0). \\ \mathbf{d} \cdot \mathbf{1} &= 1 \end{aligned}$$

These problems also differ in the first term of their respective objective functions. The corrective algorithm minimizes $\sum_{q=1}^{t} w_q(\mathbf{u}^q \cdot \mathbf{d})$, while the totally corrective algorithm minimizes $\max_{q=1,2,\ldots,t} \mathbf{u}^q \cdot \mathbf{d}$.

We now derive the Lagrangian dual of (4.12), and in the process, we show that the form of the update on $\mathbf{d}$ is equivalent to (4.11).

Lemma 4.2 *The Lagrangian dual of* (4.12) *is*

$$\max_{w_t,\beta} \ \widehat{\Theta}^t_{CB}(w_t, \boldsymbol{\psi}), \quad s.t. \ \beta \in \mathbb{R}, \ \textit{where} \tag{4.13}$$

$$\widehat{\Theta}^t_{CB}(w_t, \beta) := -\frac{1}{\eta\nu}\sum_{n=1}^{N} \ln(1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(w_t u_n^t + (1-w_t)\sum_{q=1}^{t-1} w_q^{t-1} u_n^q + \beta))) - \beta.$$

The optimal solution $\mathbf{d}^t$ *of* (4.12) *can be expressed in terms of the dual variables* w_t^t *and* β^t *as follows:*

$$d_n^t := \frac{d_n^0 \exp(-\eta w_t^t u_n^t - \eta(1-w_t^t)\sum_{q=1}^{t-1} w_q^{t-1} u_n^q - \eta\beta^t)}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta w_t^t u_n^t - \eta(1-w_t^t)\sum_{q=1}^{t-1} w_q^{t-1} u_n^q - \eta\beta^t)} \tag{4.14}$$

Furthermore, when we introduce the change of variable $w_q^t = w_t^t w_q^{t-1}$*, the update becomes*

$$d_n^t := \frac{d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q^t u_n^q + \beta^t))}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(\sum_{q=1}^{t} w_q^t u_n^q + \beta^t))}.$$

This is the same form as the update in Algorithm 4.3.

Proof The Lagrangian for this minimization problem in (4.12) is

$$L^t(\underbrace{\mathbf{d}}_{\text{primal}}, \underbrace{\alpha, \beta}_{\text{dual}}) = \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) + \alpha(\sum_{q=1}^{t-1} w_q^{t-1}(\mathbf{u}^q \cdot \mathbf{d}^t) - \mathbf{u}^t \cdot \mathbf{d}^t)$$
$$+ \beta(\mathbf{1} \cdot \mathbf{d} - 1).$$

To get the form of the update to match that of Algorithm 4.3, let us do a change of variable and substitute $w_t = 1 - \alpha$.

$$L^t(\underbrace{\mathbf{d}}_{\text{primal}}, \underbrace{w_t, \beta}_{\text{dual}}) = \mathbf{u}^t \cdot \mathbf{d} + \frac{1}{\eta}\Delta_{2,\nu}(\mathbf{d}, \mathbf{d}^0) + (1-w_t)(\sum_{q=1}^{t-1} w_q^{t-1}(\mathbf{u}^q \cdot \mathbf{d}^t) - \mathbf{u}^t \cdot \mathbf{d}^t)$$
$$+ \sum_{n=1}^{N} \psi_n(d_n - 1/\nu) + \beta(\mathbf{1} \cdot \mathbf{d} - 1). \tag{4.15}$$

The dual is derived from the Lagrangian by plugging in the minimum value of the primal variables:

$$\Theta^t_{CB}(w_t, \beta) := \inf_{\mathbf{d}} L^t(\mathbf{d}, w_t, \beta).$$

Differentiating the Lagrangian w.r.t. $\mathbf{d}$ results in

$$\frac{\partial L^t}{\partial d_n} = \frac{1}{\eta}\left[\ln\frac{d_n}{d_n^0} - \ln\frac{\frac{1}{\nu} - d_n}{\frac{1}{\nu} - d_n^0}\right] + w_t u_n^t - (1 - w_t)\sum_{q=1}^{t-1} w_q^{t-1} u_n^q + \beta.$$

The update for d_n has the form

$$d_n^t := \frac{d_n^0 \exp(-\eta w_t u_n^t - \eta(1 - w_t)\sum_{q=1}^{t} w_q^{t-1} u_n^q - \eta\beta)}{1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta w_t u_n^t - \eta(1 - w_t)\sum_{q=1}^{t} w_q^{t-1} u_n^q - \eta\beta)}.$$

By plugging the optimal $\mathbf{d}$ into the Lagrangian (4.15), the dual function simplifies to

$$\Theta^t_{CB}(w_t, \beta) = -\frac{1}{\eta\nu}\sum_{n=1}^{N} \ln(1 - \nu d_n^0 + \nu d_n^0 \exp(-\eta(w_t u_n^t + (1 - w_t)\sum_{q=1}^{t} w_q^{t-1} u_n^q + \beta))) - \beta.$$

The primal objective is convex and the primal constraints are affine. Also, the uniform distribution is always a feasible solution. Therefore, Slater's condition tells us that since this problem has a non-empty feasible set, strong duality holds and the values of the primal and the dual problems are the same.

■

Chapter 5

Experimental Evaluation

In this chapter, we present an experimental evaluation of LPBoost, ERLPBoost, and Binary ERLPBoost. We address five different experimental questions. The first question is whether LPBoost is stable in practice. We found that it is unstable for most data sets on most hypothesis classes, so we asked whether the instability is corrected by either replacing the hard margin with the soft margin or by adding entropy regularization. The second question is whether the theoretical value of the regularization parameter for ERLPBoost and Binary ERLPBoost is optimal, and if not, what constitutes sufficient regularization? The third question is whether Binary ERLPBoost is faster than ERLPBoost and whether there is a difference in generalization performance between the two algorithms. The fourth question is how totally corrective algorithms compare to their corrective counterparts. The final question is how these algorithms compare to each other in terms of their best overall performance.

Table 5.1 lists the data sets used in our experiments. Every data set can be

Data Set	Train/Test Size	Dimensions
news20 [42]	19,996	1,355,191
real-sim [55]	72,309	20,958
astro-physics [40]	62,369/32,487	99,757
adult9 (*a9a*) [3]	32,561/16,281	123
german [3]	1000	24
diabetes [3]	768	8

Table 5.1: Data sets used in experimentation. The reported size of the training and test sets are *before* post-processing.

Data Set	Train	Valid	Test Size
news20	11,997	3,999	4,000
real-sim	43,385	14,462	14,462
astro-ph	56,913	18,971	18,972
a9a	29,305	9,768	9,769
german	600	200	200
diabetes	460	154	154

Table 5.2: Data sets after processing.

found in the LIBSVM[1] repository [16] except for the Astro-physics data set [40]. The data sets are post-processed according to the following procedure. First, the training and test sets are concatenated to make a single large data set. Then, we randomly assign examples such that 60% of the examples are in the training set, 20% of the examples are in the validation set, and 20% of the examples are in the test set. The script we used to process the data is freely available[2]. The sizes of the processed data sets are listed in Table 5.2.

The parameters used in our experiments are listed in Table 5.3. Recall that ϵ is the precision parameter, ν is the capping parameter, and η is the regularization parameter for algorithms with entropy regularization. Note that unless otherwise specified,

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`
[2] `http://www.soe.ucsc.edu/~kag/dissertation.shtml`

ERLPBoost, Binary ERLPBoost		
param	*description*	*value*
η	regularization	1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 3000
ν/N	capping	$1/N$, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
ϵ	precision	0.001
max-iter	max number of iterations	1,000
reflexive	consider negative features	True
opt	type of optimizer	TAO [5]
Corrective ERLPBoost, Corrective Binary ERLPBoost		
param	*description*	*value*
η	regularization	1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 3000
ν/N	capping	$1/N$, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
ϵ	precision	0.001
max-iter	max number of iterations	20,000
reflexive	consider negative features	True
LPBoost		
param	*description*	*value*
ν/N	capping	$1/N$, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
ϵ	precision	0.01
max-iter	max number of iterations	1,000
reflexive	consider negative features	True

Table 5.3: Parameters used in ERLPBoost, Binary ERLPBoost, and Corrective ERLPBoost. Note that we are fixing the ratio ν/N, where N is the number of examples. Although ν values will vary from data set to data set, the ratio remains fixed.

$\nu = 1$ and η is the theoretical value established in Chapter 3. Because it has not been mentioned before, the reflexive parameter requires some explanation. For a given set of hypotheses, if the reflexive parameter is set to *True*, then for each hypothesis in the set, its negative counterpart is also in the set. This provides a way to use hypotheses that are negatively correlated with the labels in the scenario where negative weights are not allowed. Some hypothesis classes, for example decision stumps, do this implicitly.

The parameters for ERLPBoost and Binary ERLPBoost are identical, but a few of the parameters for the corrective algorithms differ slightly. For consistency, all four of these algorithms are called with the same sets of values for ϵ, ν, and η. However, the corrective algorithms are allowed a much higher maximum number of iterations. This is because while each iteration of the corrective algorithm is faster than its totally corrective counterpart, it requires an order of magnitude more iterations to converge properly. The optimizer type is specified for the totally corrective algorithms. The corrective algorithms do not require an optimization solver.

The LPBoost parameters are also similar. While LPBoost has no regularization and therefore no regularization parameter η, we use the same values of ν. However, the LPBoost is extremely unstable on this data, and for $\epsilon = 0.001$, it would not converge. We therefore had to use $\epsilon = 0.05$ and 0.01 instead. LPBoost was implemented with the COIN-LP linear programming solver [50]. The only tunable parameter for AdaBoost is the number of iterations. We set the number of iterations to 20,000, the same number used for the corrective algorithms.

In our experiments, the oracle will return either decision stump, raw data, or

SVM hypotheses. We now describe the three hypothesis classes. Consider the original data sets to be matrices whose rows are the examples and whose columns are the original features. A raw data hypothesis is simply a column of this matrix, which is equivalent to a feature of the original data. A decision stump takes a column in the matrix we just described and thresholds it, predicting $+1$ for all values on one side of the threshold and -1 for all examples on the other side. Thus, a decision stump can be represented by a column number, a direction, and a threshold. Finally, an SVM hypothesis is equivalent to multiplying the original data matrix by one of its rows, where a row corresponds to an example. The resulting column vector is a linear combination of the columns of the data matrix. Note that boosting SVM hypotheses is not the same as training SVMs on the data and boosting the result. We will see that the results are heavily dependent on the class of hypotheses returned by the oracle, so when we discuss our results, we will specify the algorithm, the data set, and the hypothesis class.

Last but not least, we discuss the strength of the oracle we use in our experiments. Recall that at each iteration, boosting algorithms send the current distribution $\mathbf{d}$ to the oracle, which then returns a hypothesis. Our theoretical analysis only assumes that the oracle will return a hypothesis with edge at least g w.r.t. $\mathbf{d}$. In our experiments, on the other hand, the oracle returns the hypothesis of *maximum* edge w.r.t. $\mathbf{d}$. Thus, the oracle that we use in our experiments is stronger than the oracle that we use in our theoretical analysis. The reason for this discrepancy is that, for the hypothesis classes that we use in our experiments, finding the hypothesis of maximum edge is so easy that we considered it to be the right approach.

5.1 The Instability of LPBoost

In this section, we ask whether LPBoost is stable by looking at generalization error, the ultimate measure of any machine learning algorithm. LPBoost [36, 21] is a boosting algorithm that directly maximizes the margin over all of its current hypothesis at each iteration via linear programming. In Chapter 2, we showed that any linearly separable data set can be reduced to a data set on which LPBoost misclassifies all examples by adding a bad example and a bad hypothesis. Thus we have established that in at least one case, the generalization error can change dramatically in a single iteration. We conjecture that there will also be cases where LPBoost is similarly unstable in practice.

To test this conjecture, we plot the generalization error as a function of iteration for LPBoost in the uncapped case ($\nu = 1$). Each plot follows a single run of LPBoost through the training process. During each training iteration, a new hypothesis is added to the ensemble and LPBoost finds a new $\mathbf{w}$ that maximizes the margin. We then measure the generalization error achieved by this new ensemble on the test set. This tells us how well LPBoost would have generalized if it had terminated at that iteration. If LPBoost is stable, we would expect to see generalization error decrease steadily. Small increases in generalization error are reasonable, but generalization error should not increase dramatically in a single iteration. Recall that the only tunable parameter for LPBoost is ϵ, which controls the number of iterations. If tweaking ϵ to make LPBoost run for one more iteration also causes the generalization error to spike,

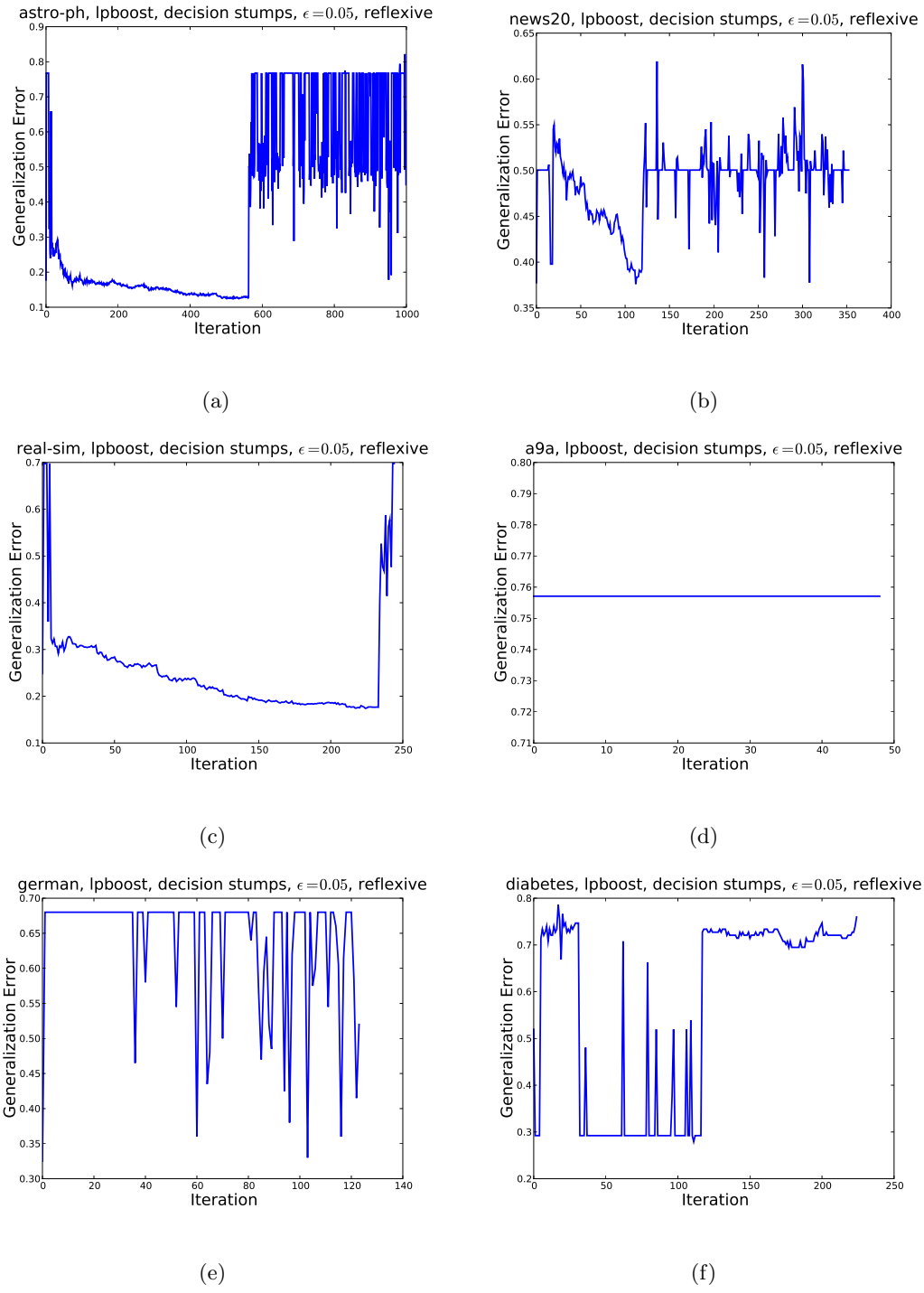
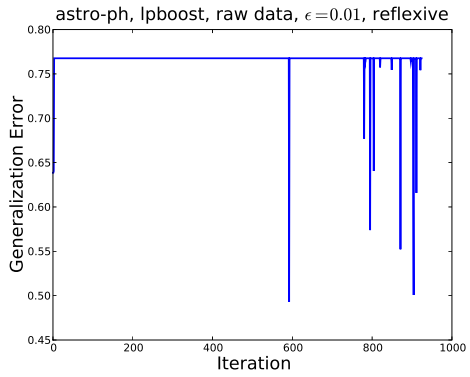
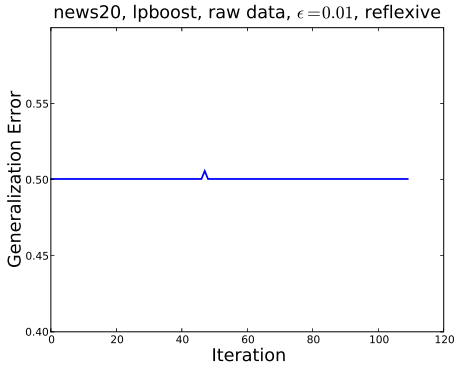


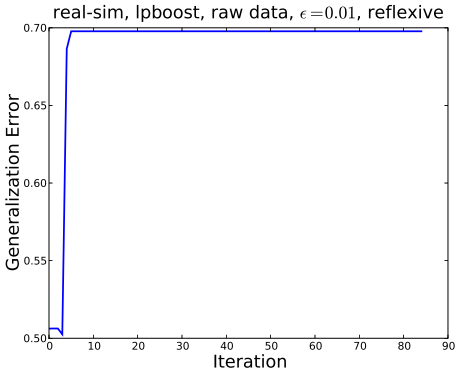
Figure 5.1: LPBoost *without* capping for **decision stump** hypotheses. This algorithm is extremely unstable.



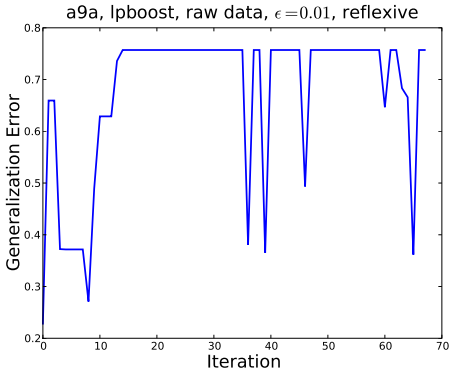
(a)



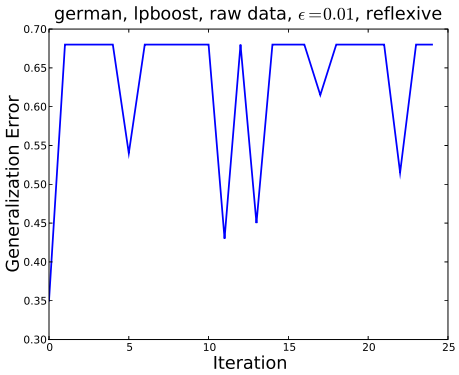
(b)



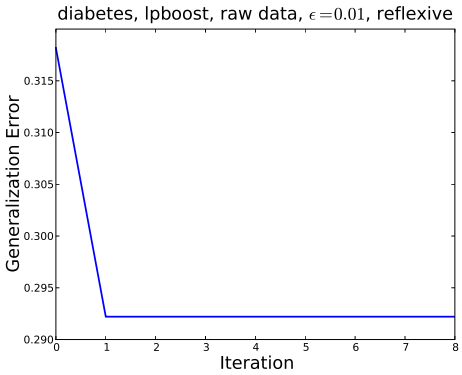
(c)



(d)

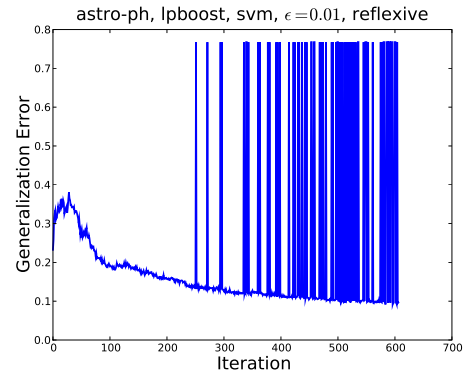


(e)

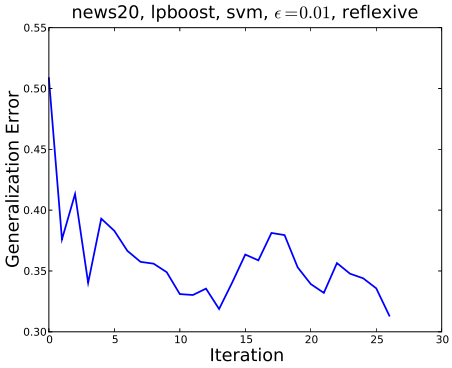


(f)

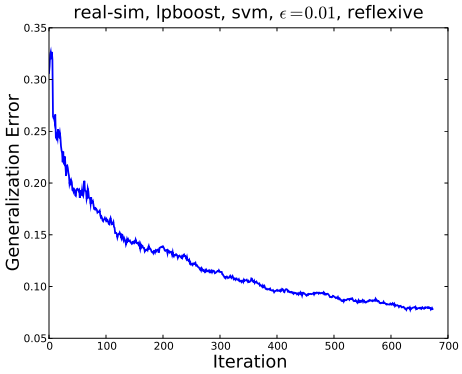
Figure 5.2: LPBoost *without* capping for **raw data** hypotheses. This algorithm is extremely unstable.



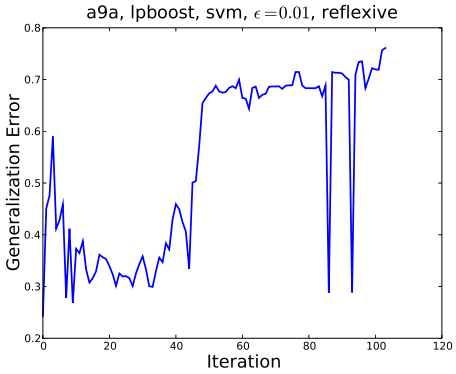
(a)



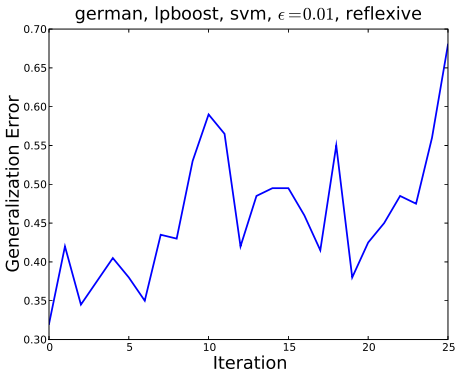
(b)



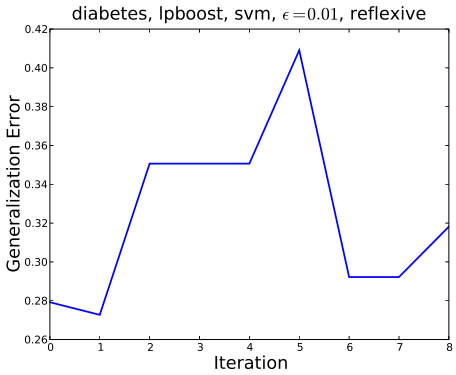
(c)



(d)



(e)



(f)

Figure 5.3: LPBoost *without* capping for **SVM** hypotheses. This algorithm is extremely unstable.

then the algorithm is not stable enough to be practical.

The main result in this section is that sudden increases in generalization error are extremely common for LPBoost. Figure 5.1 shows generalization error as a function of iteration for LPBoost with decision stump hypotheses. Note that the range of the y axis varies because it is chosen to match the range of the underlying data. Instead of decreasing steadily, the generalization error fluctuates wildly. This effect was observed on *every* data set except *a9a*, on which LPBoost has consistently poor performance. Moreover, the same effect can be observed for raw data hypotheses. Figure 5.2 shows generalization error as a function of iteration for LPBoost for raw data hypotheses, and these results are unstable on every data set except *news20* and *diabetes*. Finally, Figure 5.3 plots generalization error as a function of iteration for SVM hypotheses. In this case, LPBoost was unstable on every data set except *real-sim*. It is not clear why LPBoost is stable on *real-sim* and not on the other data sets. We have therefore demonstrated that in the uncapped case, LPBoost is unstable on nearly every data set for all three different classes of hypotheses. Our intuition was that LPBoost would be unstable on sparse data sets, especially if the hypotheses preserved the sparsity. Decision stumps and raw data hypotheses preserve sparsity, but LPBoost was also unstable on SVM hypotheses, which are dense because they are the result of a matrix-vector product.

We now ask whether maximizing the soft margin ($\nu > 1$) will stabilize LPBoost. In the previous experiment, an examination of the $\mathbf{d}$ and $\mathbf{w}$ distributions found by LPBoost shows that LPBoost is putting weight on very few examples and hypotheses. Because capping forces the algorithm to distribute the weight across more examples, it

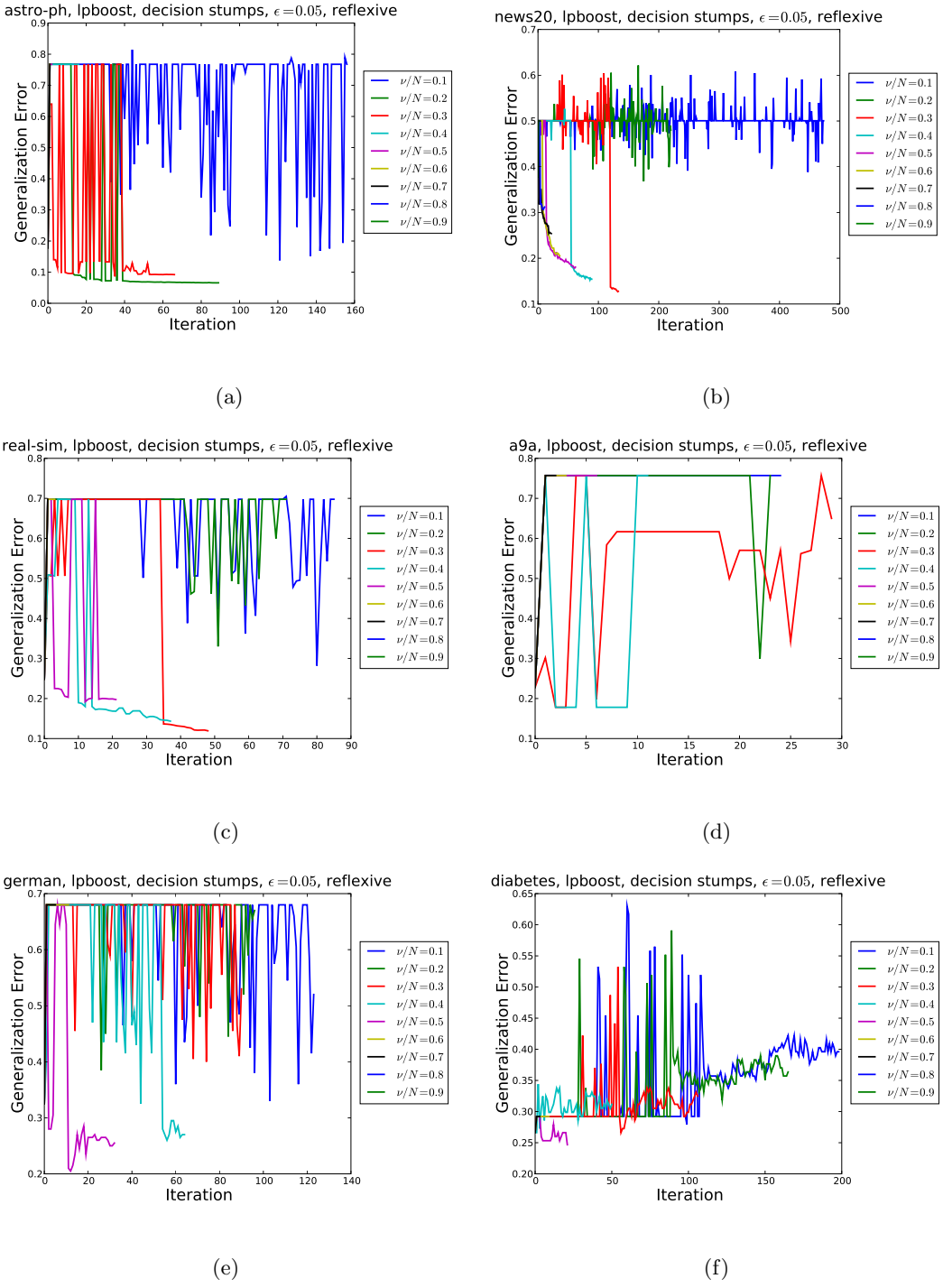


Figure 5.4: LPBoost **with** capping for **decision stump** hypotheses. Capping does not make LPBoost more stable.

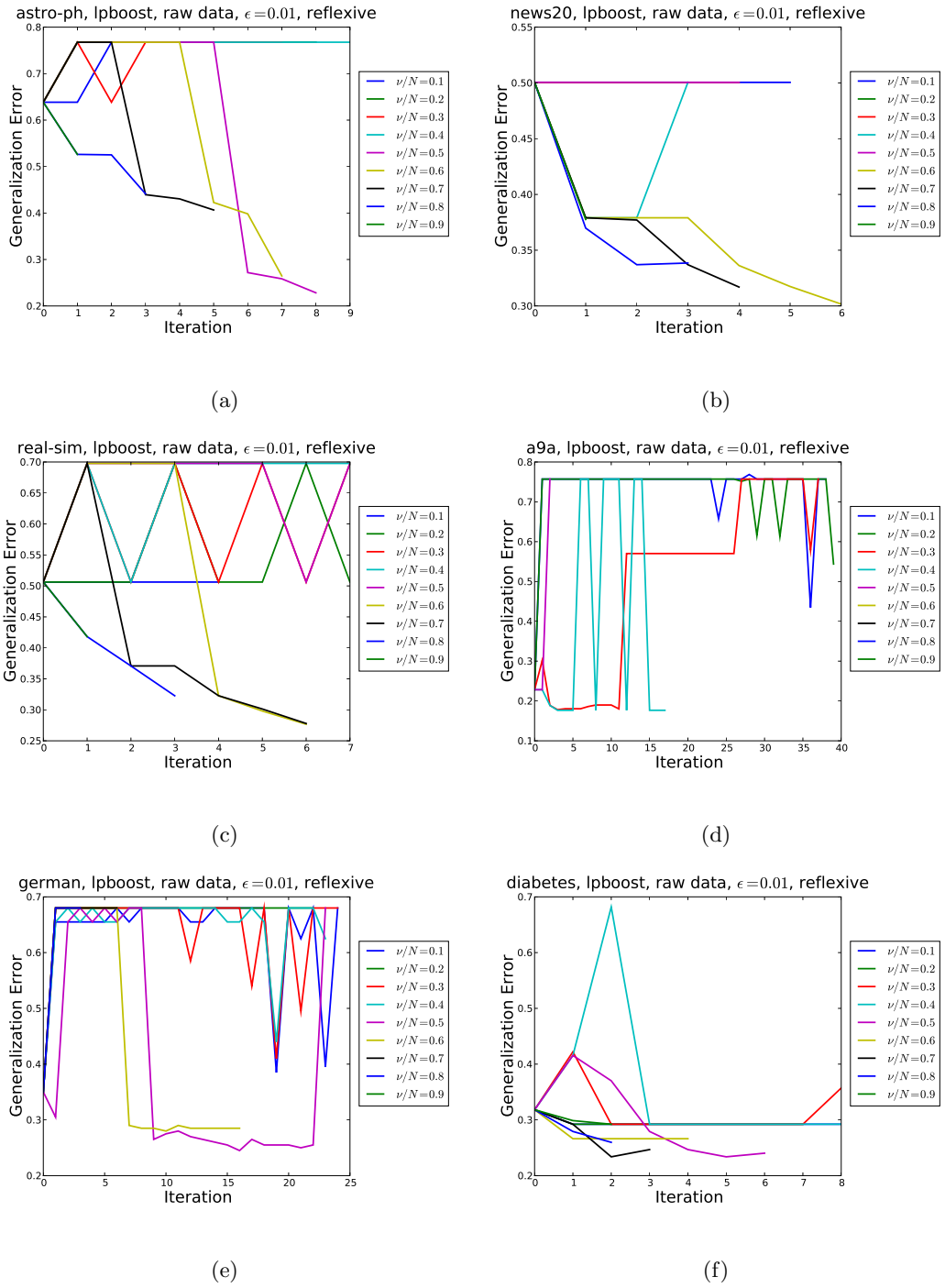
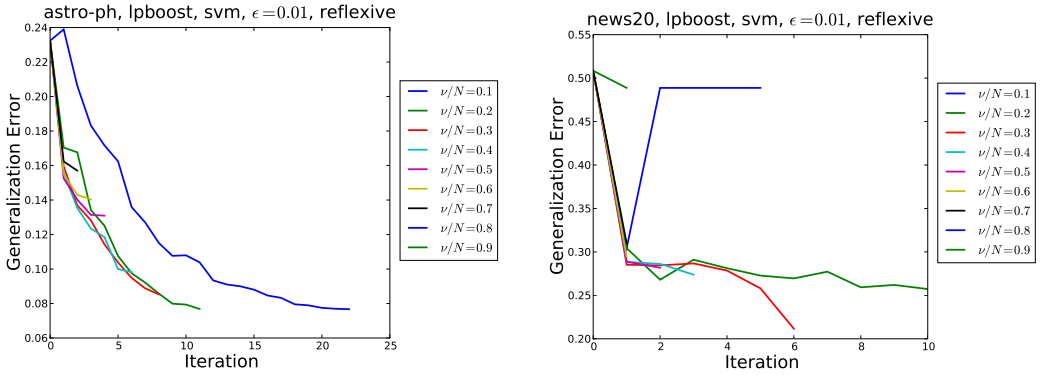
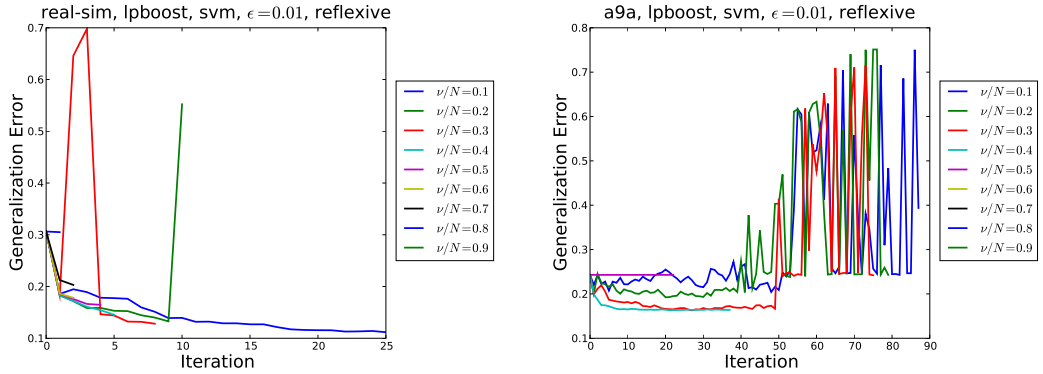


Figure 5.5: LPBoost **with** capping for **raw data** hypotheses. Capping does not make LPBoost more stable.



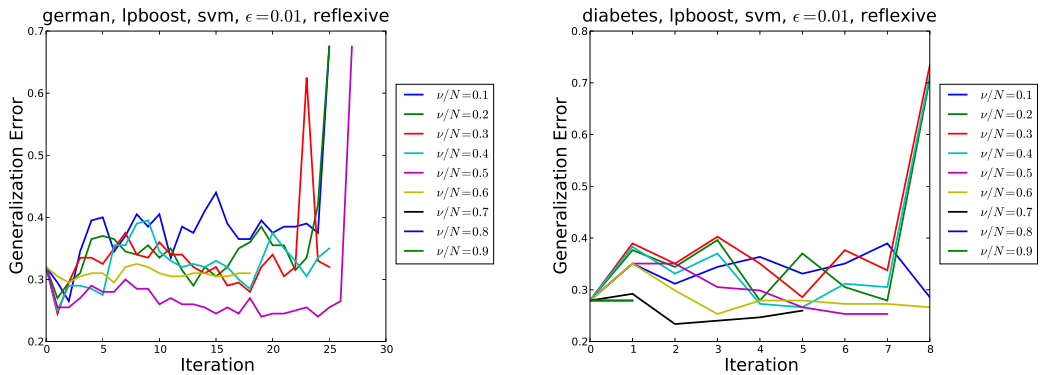
(a)

(b)



(c)

(d)



(e)

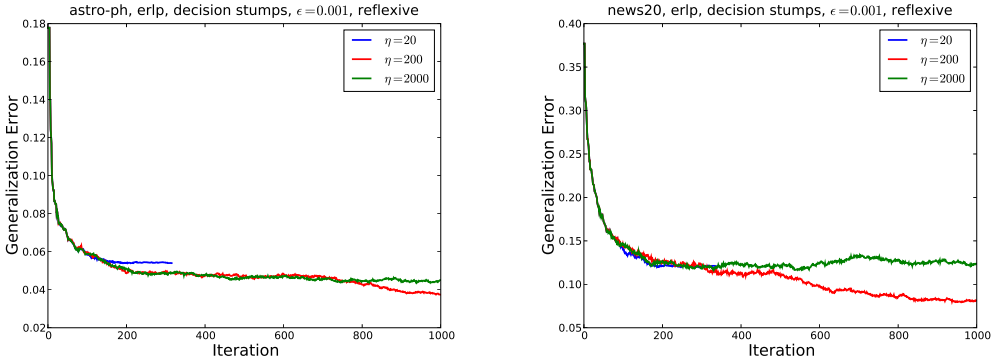
(f)

Figure 5.6: LPBoost **with** capping for **SVM** hypotheses. Capping does not make LPBoost more stable.

is plausible that introducing capping will eliminate the instability we have just observed. Figures 5.4, 5.5, and 5.6 show generalization error as a function of iteration for decision stumps, raw data, and SVM hypotheses respectively. The key result in these figures is that capping does not stabilize the algorithm. The only example where LPBoost is stable is the *real-sim* data set for SVM hypotheses, shown in Figure 5.6. Also note that for higher values of ν/N, the algorithms stop early. To understand why, recall that $\mathbf{d} \leq \frac{1}{\nu}\mathbf{1}$. Increasing ν constrains $\mathbf{d}$ more tightly. In particular, when $\nu = N$, d must remain uniform and the algorithm will terminate after a single iteration. In this way, capping works as an early stopping mechanism.

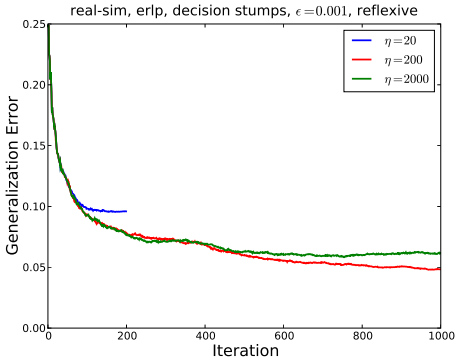
We now ask whether adding entropy regularization results in a more stable algorithm. Figures 5.7, 5.8 and 5.9 show generalization error as a function of iteration for ERLPBoost with decision stumps, raw data, and SVM hypotheses respectively. In these figures, we show results for ERLPBoost run with three different values of the regularization parameter: $\eta = 20$, 200, and 2000 (the theoretical value for $\epsilon = 0.001$ is approximately $20,000$). Recall that the regularization term is proportional to $1/\eta$, so $\eta = 2000$ represents a small amount of regularization. In almost every case, the instability that plagued LPBoost is now gone. Surprisingly, even a small amount of regularization is sufficient to ensure the stability of ERLPBoost. We discuss the tuning of the regularization parameter in Chapter 5.2.

It is worth noting that Figures 5.7(e) and 5.7(f) are not unstable. The upward drift of the generalization error in successive iterations suggests that ERLPBoost is overfitting. Nevertheless, the generalization error does not fluctuate dramatically at

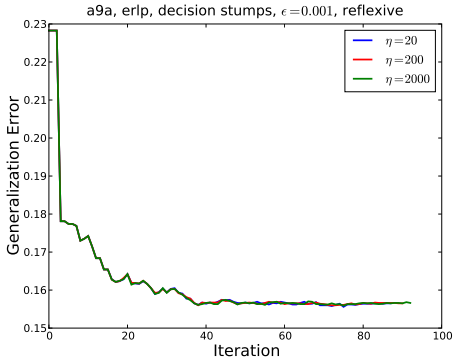


(a)

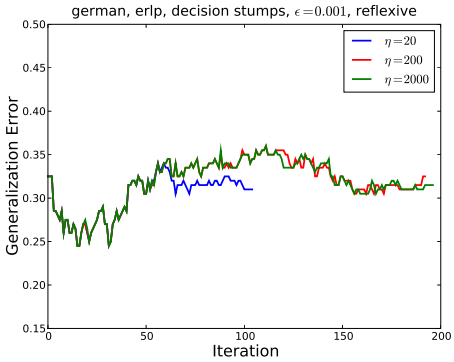
(b)



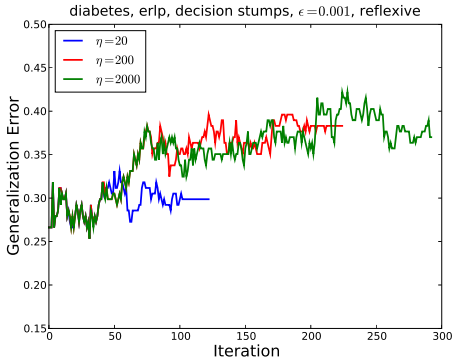
(c)



(d)

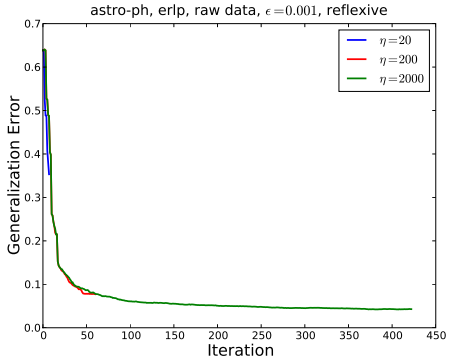


(e)

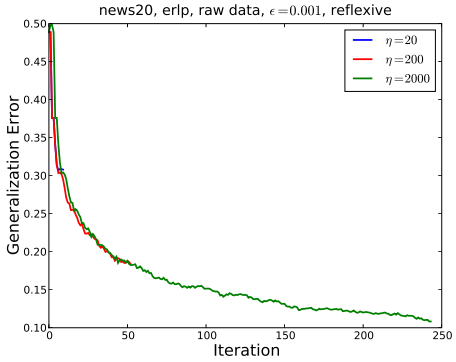


(f)

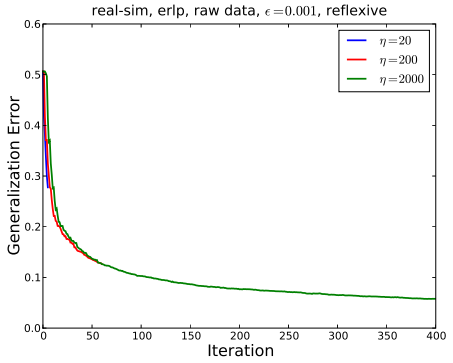
Figure 5.7: Entropy Regularized LPBoost for **decision stump** hypotheses. Entropy regularization **does** result in a stable algorithm.



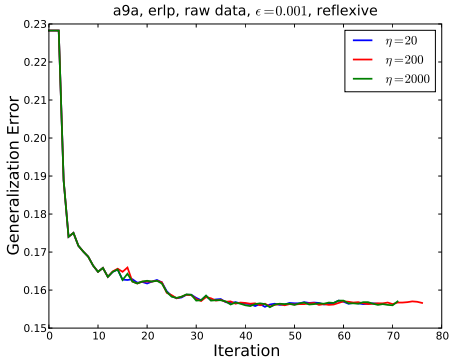
(a)



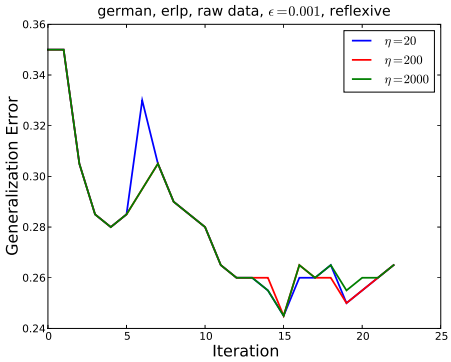
(b)



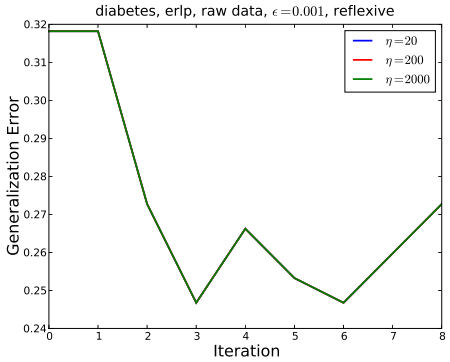
(c)



(d)



(e)



(f)

Figure 5.8: Entropy Regularized LPBoost for **raw data** hypotheses. Entropy regularization **does** result in a stable algorithm.

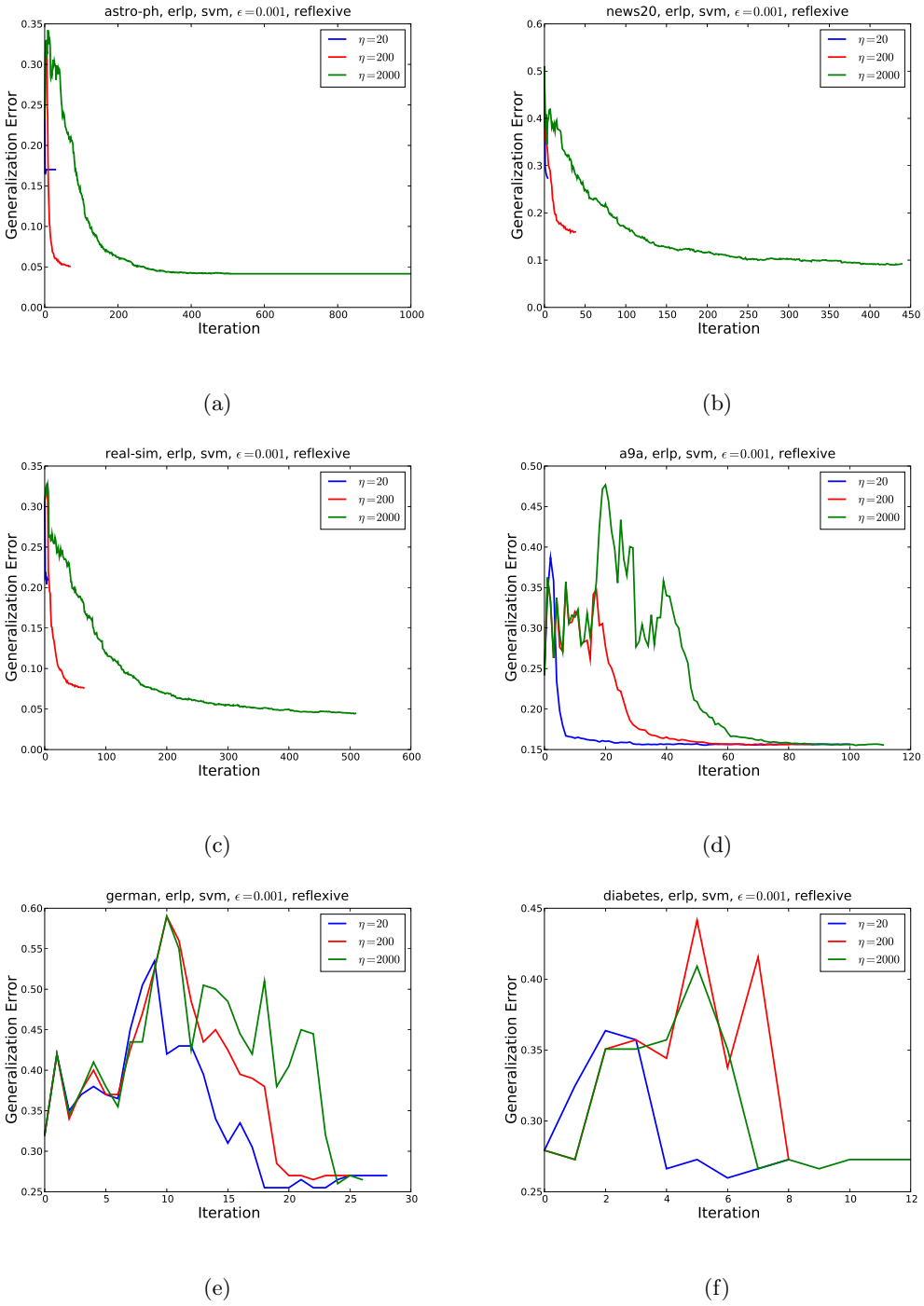


Figure 5.9: Entropy Regularized LPBoost for **SVM** hypotheses. Entropy regularization **does** result in a stable algorithm.

each iteration the way it did for LPBoost. The most unstable instance of ERLPBoost occurs in Figure 5.9(f), where the generalization error changes by as much as 0.07 in a single iteration. Nevertheless, this is not nearly as severe as the comparable fluctuations of LPBoost, shown in Figures 5.3(f) and 5.6(f).

It is also worth discussing the value of ϵ used in these experiments. For every algorithm in this thesis except LPBoost, we report results for $\epsilon = 0.001$. For many of the data sets we use, the algorithms in this thesis require ϵ as small as 0.001 to achieve a reasonable generalization error. Note that for LPBoost, we only report results for $\epsilon = 0.05$ and 0.01. The reason is that LPBoost will not converge in a timely manner (if at all) for smaller values of ϵ. However, for LPBoost, changing ϵ only changes the number of iterations. The generalization error at each iteration remains the same, and so does the brittleness shown in Figures 5.1 through 5.6. The point of these plots is to show the brittleness of LPBoost, and for this it suffices to use a higher value of ϵ.

In summary, we have shown that LPBoost is extremely unstable on real world data and that capping does not mitigate this problem. However, even a small amount of entropy regularization resolves the instability and causes generalization error to steadily decrease over time.

5.2 Sufficient Regularization

In this section, we investigate whether more regularization can result in reduced generalization error. Recall that in the theoretical analysis of Entropy Regularized

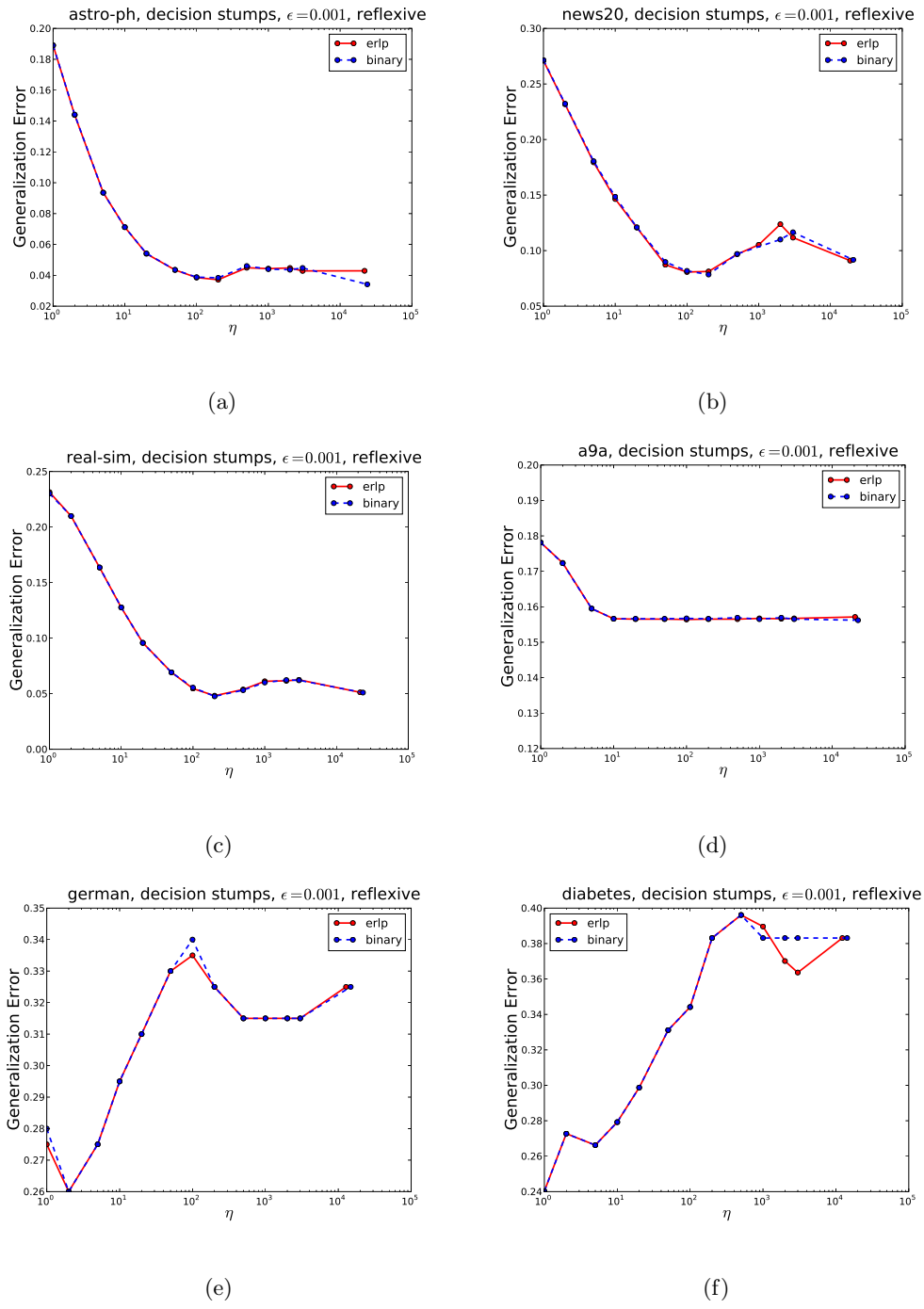


Figure 5.10: **Generalization error vs.** η (the regularization parameter) for ERLP-Boost and Binary ERLPBoost with **decision stump** hypotheses. Recall that larger η means less regularization.

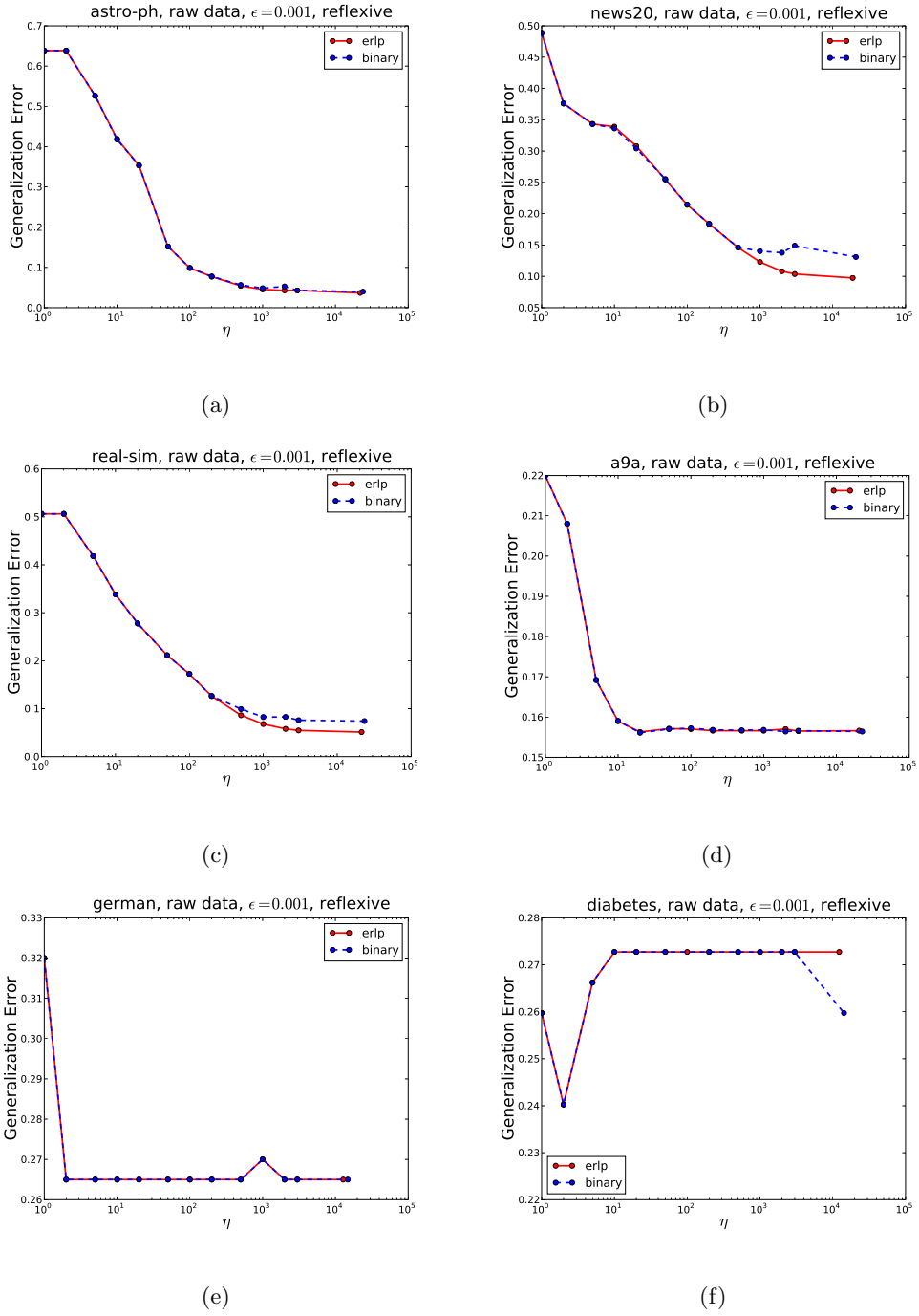


Figure 5.11: **Generalization error vs.** η (the regularization parameter) for ERLPBoost and Binary ERLPBoost with **raw data** hypotheses. Recall that larger η means less regularization.

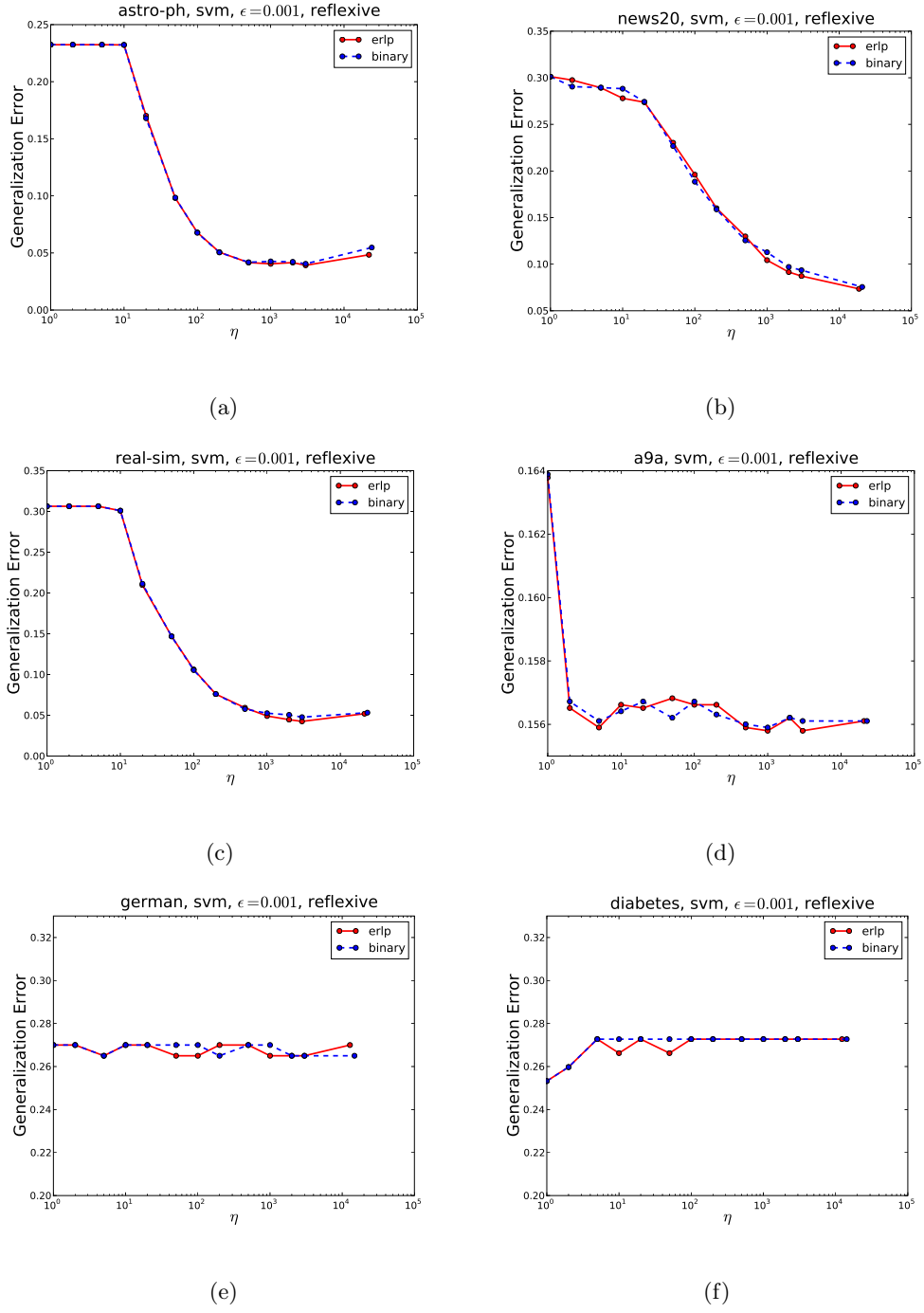


Figure 5.12: **Generalization error vs.** η (the regularization parameter) for ERLP-Boost and Binary ERLPBoost with **SVM** hypotheses. Recall that larger η means less regularization.

LPBoost in Chapter 3.2, we set $\eta = \frac{2}{\epsilon} \ln \frac{N}{\nu}$. Similarly, in the theoretical analysis of Binary Entropy Regularized LPBoost in Chapter 3.3, we set $\eta = \frac{2}{\epsilon} \left(\ln \frac{N}{\nu} + 1 \right)$. We chose these values to ensure that the value of the regularization term will never exceed ϵ. This allows us to assert that when the algorithm terminates, the value of the optimization problem will be ϵ-close to the linear programming solution P_{LP}. It would be interesting to know whether coming within ϵ of the optimal linear programming solution is truly important. For this reason we consider the possibility that more regularization might result in lower generalization error.

The figures in this section plot generalization error as a function of the regularization parameter η. Recall that the regularization coefficient is $\frac{1}{\eta}$, so as η increases, the amount of regularization decreases. In these plots, η ranges from 1 to the theoretical value. Observe that the theoretical value for Binary ERLPBoost is larger than the theoretical value for ERLPBoost. For decision stump hypotheses, shown in Figure 5.10, the theoretical value of η is a reasonable choice for all data sets except *german* and *diabetes*. However, with more regularization, the algorithms generalize just as well if not better. For raw data hypotheses, shown in Figure 5.11, the theoretical value is the best for *astro-ph*, *news20*, and *real-sim*. More regularization is equally good for *a9a* and more regularization significantly reduces the generalization error for the *german* and *diabetes* data sets. Finally, Figure 5.12 shows that for SVM hypotheses, the theoretical value of η is the best for the *news20* data set and it is a reasonable choice for all other data sets.

The naive interpretation of tuning η focuses only on how close ERLPBoost

gets to the maximum margin solution. However, it is also important to consider that insufficient regularization can also lead to overfitting. Increasing η also increases the number of iterations and recall that boosting algorithms add a new hypothesis at each iteration. Therefore, the longer the algorithm is allowed to run, the more complex the master hypothesis becomes, which can lead to overfitting. To illustrate this point, observe that *german* and *diabetes* are the only two data sets that seem to require extremely high regularization. The likely explanation is that these are the smallest data sets and they begin overfitting in very few iterations. Decreasing η forces the algorithm to terminate sooner, which prevents overfitting. A good illustration of this can be seen in Figures 5.7(e) and 5.7(f), where the generalization error initially decreases but then begins increasing again before iteration 50.

In summary, the theoretical value of η is a reasonable value except on the *german* and *diabetes* data sets. This holds true for all three hypothesis classes. Nevertheless we have shown that there are many cases where increasing the regularization will not hurt the generalization error and may even reduce it.

5.3 ERLPBoost vs. Binary ERLPBoost

In this section, we compare ERLPBoost and Binary ERLPBoost in terms of execution time and generalization error. We conjectured in Chapter 3.3 that Binary ERLPBoost would be faster than ERLPBoost because the optimization problem associated with Binary ERLPBoost has fewer variables. Recall from Chapter 3 that the

difference comes from the way the capping constraints are enforced. The variables in the ERLPBoost optimization problem are $\mathbf{w} \in \mathcal{S}^t$ and $\boldsymbol{\psi} \in \mathbb{R}^N_{\geq 0}$, where the $\boldsymbol{\psi}$ variables in the $\mathbf{w}$ domain are the dual variables associated with the capping constraints in the $\mathbf{d}$ domain. This is a problem of $t + N$ dimensions. In contrast, the optimization problem solved by Binary ERLPBoost enforces the capping constraints implicitly. The variables of Binary ERLPBoost in the $\mathbf{w}$ domain are $\mathbf{w} \in \mathcal{S}^t$ and $\beta \in \mathbb{R}$, so the problem has $t+1$ dimensions. Because its optimization problem has fewer variables, we conjectured that Binary ERLPBoost should converge more quickly than ERLPBoost.

To test this conjecture, we set η to the theoretical value and plotted the training time as a function of ν/N. We chose to vary ν/N because the capped case is where we expect the two algorithms to differ the most. Recall that N is the total number of examples and $\nu \in [1 \ldots N]$ is the capping parameter. The uncapped case corresponds to $\nu = 1$, and increasing the value of ν/N increases the stringency of the capping constraint. The purpose of these plots is to compare ERLPBoost and Binary ERLPBoost over a range of parameters, not to establish a relationship between ν/N and total time. Figure 5.13 shows total execution time as a function of ν/N for the two algorithms with decision stump hypotheses. In most cases Binary ERLPBoost is faster than ERLPBoost. Figure 5.14 shows this is true for raw data hypotheses as well. The results for SVM hypotheses, shown in Figure 5.15, are more mixed. For the *german* and *diabetes* data sets, Binary ERLPBoost is clearly faster than ERLPBoost for all ν, but the opposite is true of the other data sets.

We now ask whether one particular type of entropy regularization results in

lower generalization error than the other. This is an interesting question because when $\nu = 1$ and $\eta = 1$, ERLPBoost optimizes an exponential loss function and Binary ERLPBoost optimizes a logistic loss function. It has been argued that the logistic loss function results in an algorithm that is more robust to noise [34]. If this is true, then it may cause Binary ERLPBoost to generalize better than ERLPBoost. Figure 5.16 plots generalization as a function of ν/N for decision stump hypotheses. We found that the generalization error of the two algorithms is virtually identical. The same is true for SVM hypotheses, shown in Figure 5.18. The two algorithms are also very similar for raw data hypotheses, shown in Figure 5.17, but not quite as similar as they are for decision stumps and SVM hypotheses. Binary ERLPBoost performs slightly better on *german* and *diabetes*, the two noisiest data sets. ERLPBoost performs significantly better on *astro-ph* and *real-sim*, and the two algorithms are virtually the same on *a9a*.

These plots also show the effect of capping on each data set. There are three data sets with relatively little noise: *astro-ph*, *news20*, *real-sim*. On these data sets, capping (setting $\nu > 1$) increases the generalization error. There is another data set with a moderate amount of noise, *a9a*, where capping does not help much either. As we will show later, all algorithms are able to achieve a generalization error of approximately 0.15 on *a9a* without capping. Some values of ν/N do not seem to hurt the generalization, but none seem to be able to improve on this value. Finally, there are two data sets that are quite noisy: *german* and *diabetes*. On these data sets, capping appears to be quite effective. This discussion holds true for all three hypothesis classes.

In summary, Binary ERLPBoost is generally faster than ERLPBoost for de-

cision stumps and raw data hypotheses, but not for SVM hypotheses. In terms of generalization error, the algorithms are virtually identical for decision stump and SVM hypotheses, and they are still very similar for raw data hypotheses.

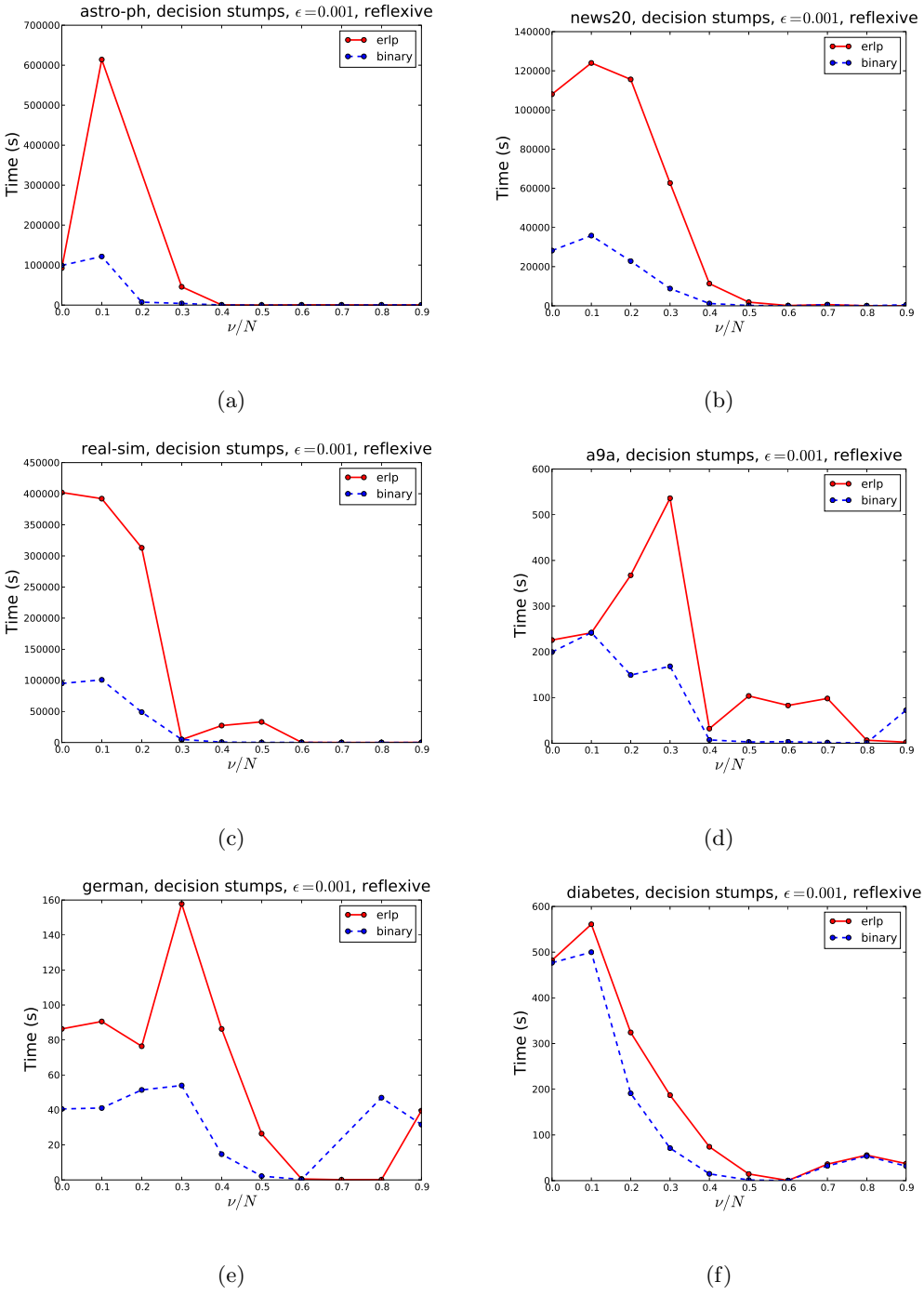
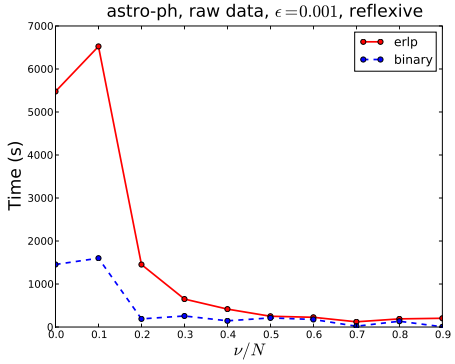
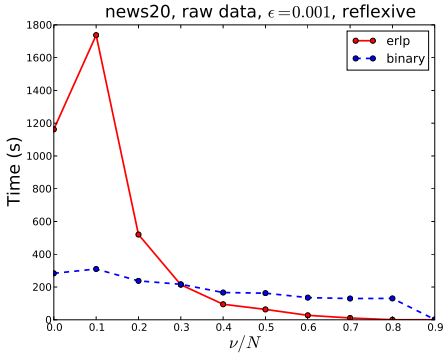


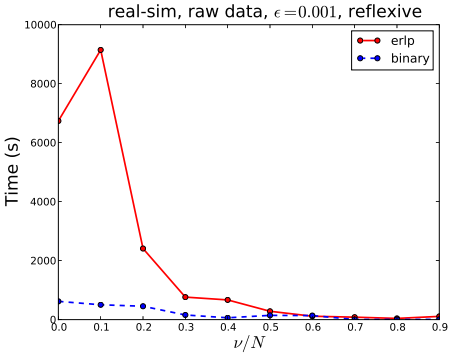
Figure 5.13: **Training time vs.** ν/N for ERLPBoost and Binary ERLPBoost with **decision stump** hypotheses. Note that $\nu = 1$ is the uncapped case, so the smaller values of ν/N have less stringent capping.



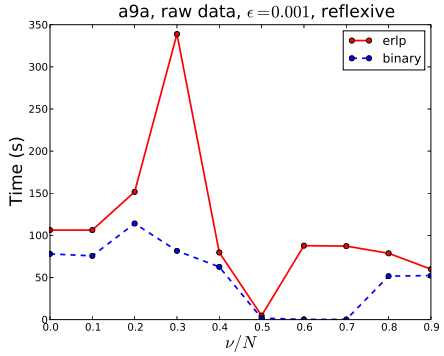
(a)



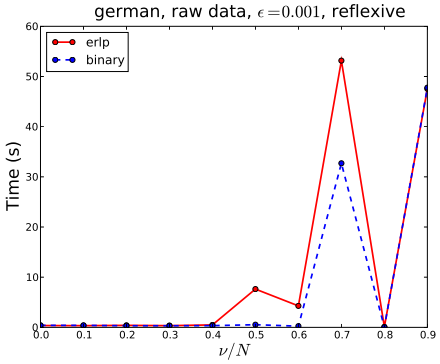
(b)



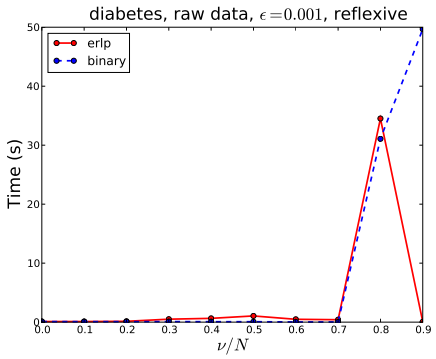
(c)



(d)



(e)



(f)

Figure 5.14: **Training time vs.** ν/N for ERLPBoost and Binary ERLPBoost with **raw data** hypotheses. Note that $\nu = 1$ is the uncapped case, so the smaller values of ν/N have less stringent capping.

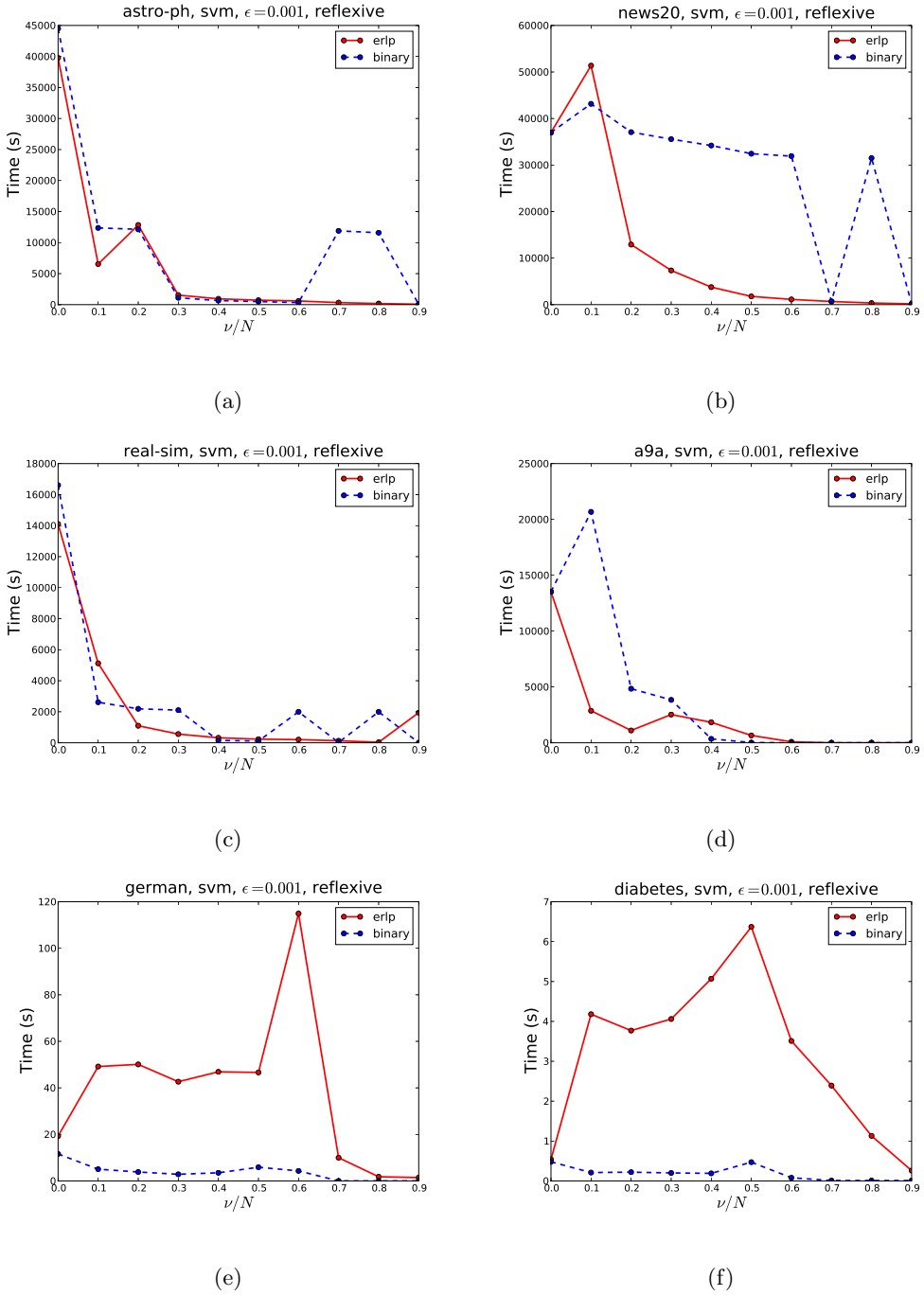


Figure 5.15: **Training time vs.** ν/N for ERLPBoost and Binary ERLPBoost with **SVM** hypotheses. Note that $\nu = 1$ is the uncapped case, so the smaller values of ν/N have less stringent capping.

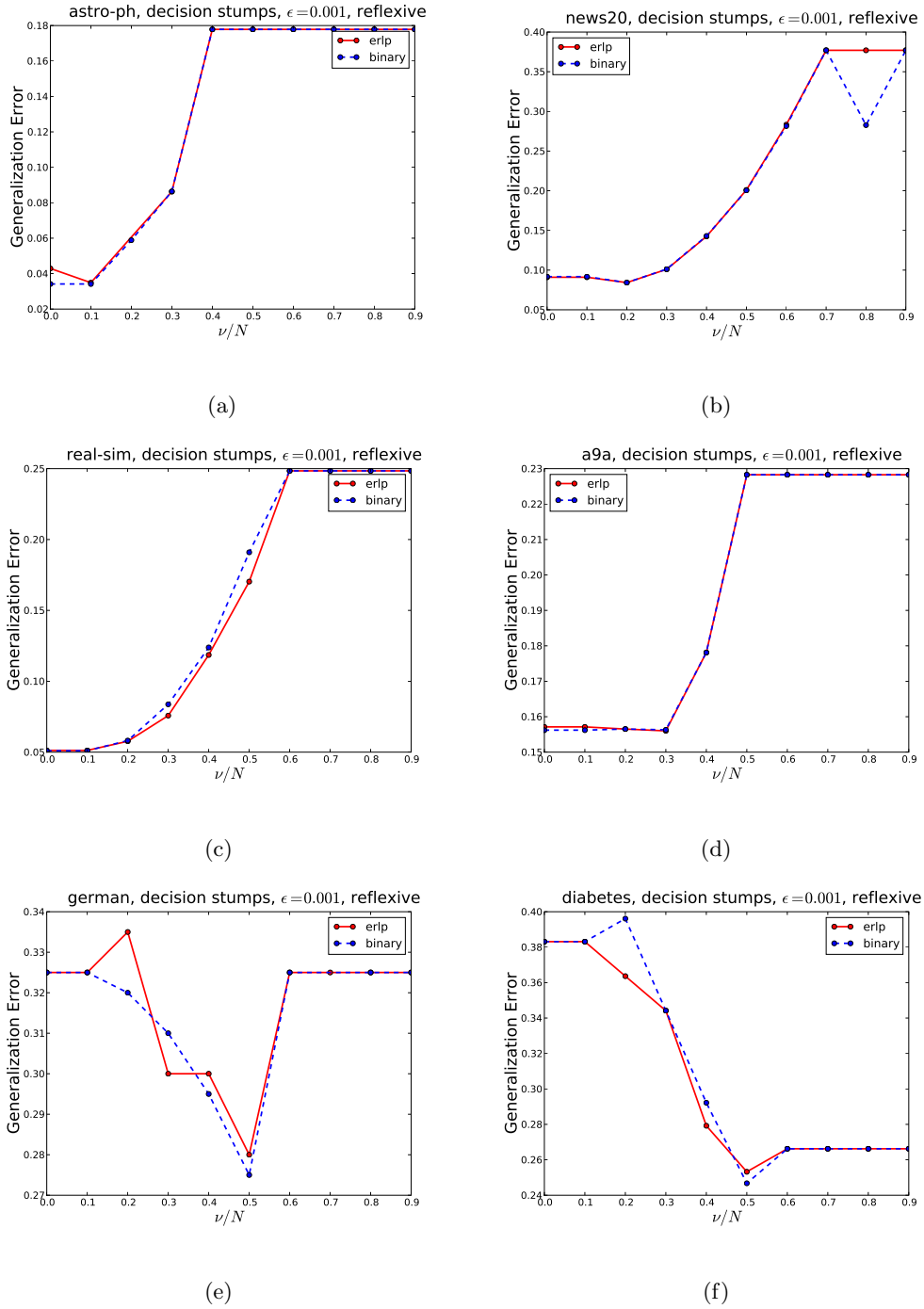


Figure 5.16: **Generalization error vs.** ν/N for ERLPBoost and Binary ERLPBoost with **decision stump** hypotheses. Note that $\nu = 1$ is the uncapped case, so the smaller values of ν/N have less stringent capping.

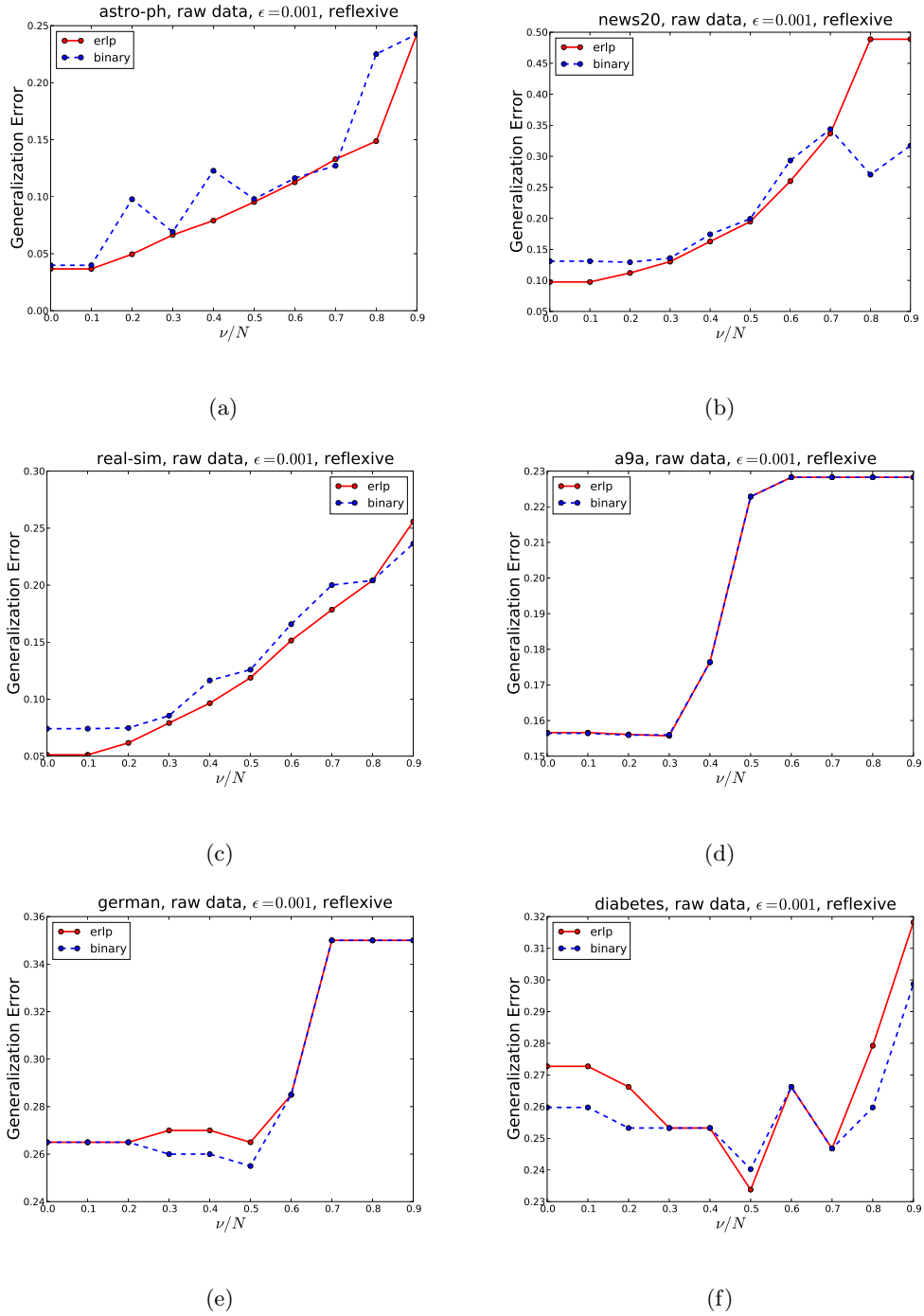


Figure 5.17: **Generalization error vs.** ν/N for ERLPBoost and Binary ERLPBoost with **raw data** hypotheses. Note that $\nu = 1$ is the uncapped case, so the smaller values of ν/N have less stringent capping.

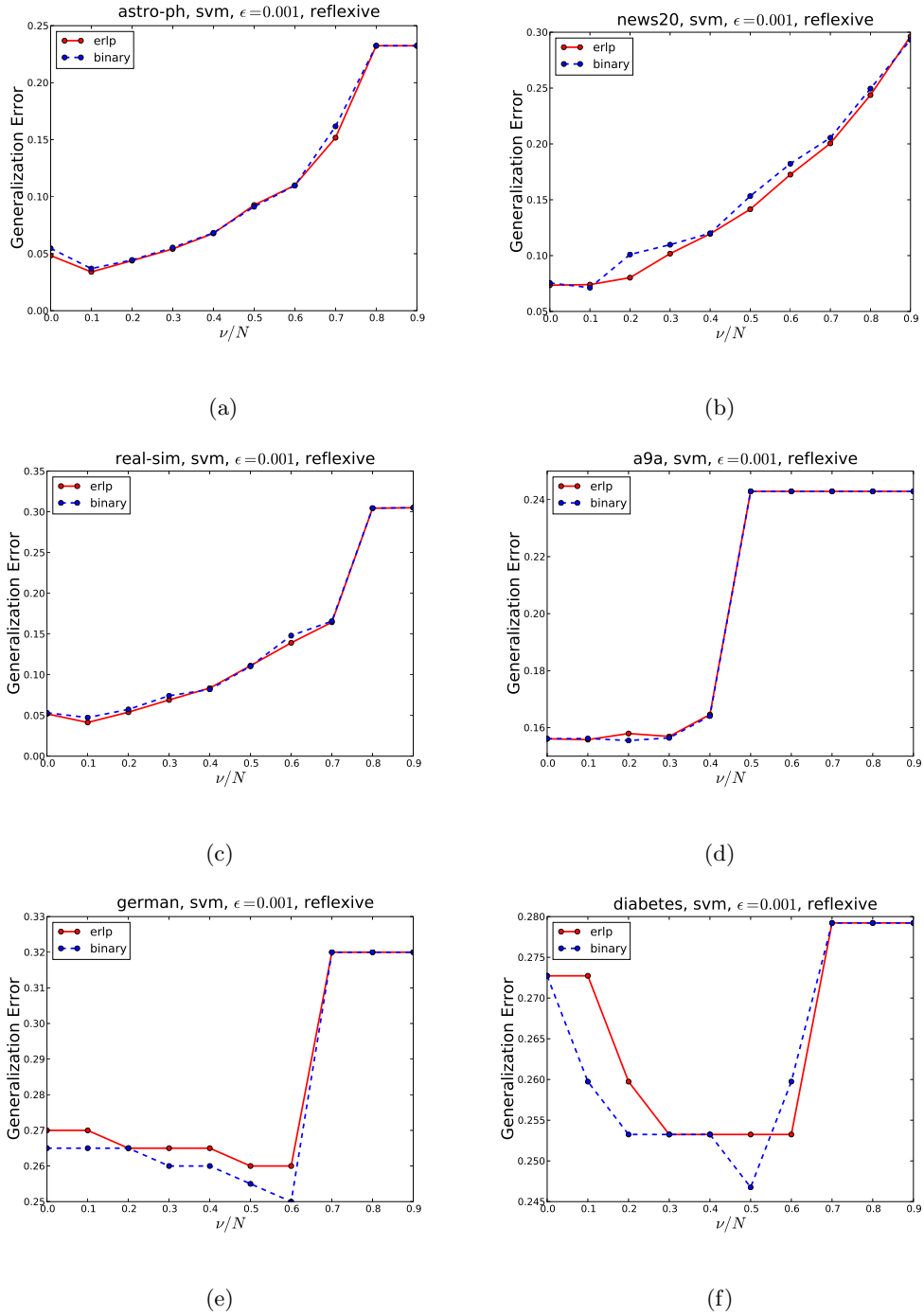


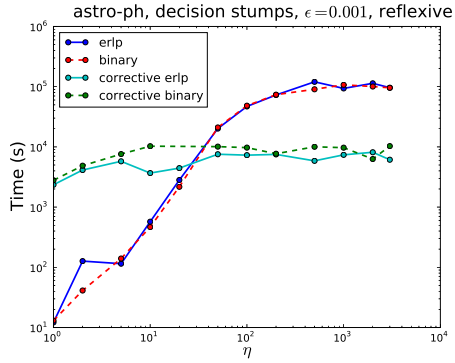
Figure 5.18: **Generalization error vs.** ν/N for ERLPBoost and Binary ERLPBoost with **SVM** hypotheses. Note that $\nu = 1$ is the uncapped case, so the smaller values of ν/N have less stringent capping.

5.4 Corrective vs. Totally Corrective Algorithms

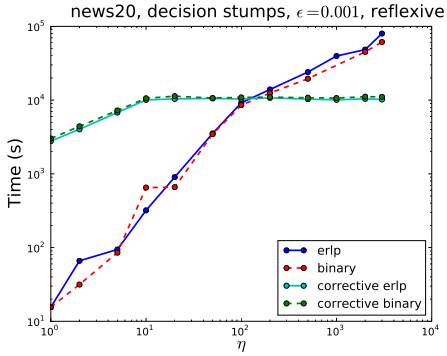
In this section, we compare the totally corrective algorithms with their corrective counterparts. Recall that the corrective family of algorithms only update the weights on the examples based on the *last* hypothesis while *totally corrective* algorithms optimize the weights on the examples based on *all* past hypotheses. The totally corrective algorithms are *ERLPBoost* and *Binary ERLPBoost*. The corrective algorithms are *Corrective ERLPBoost* and *Corrective Binary ERLPBoost*. Corrective ERLPBoost is described in detail in Chapter 4.2. It is directly comparable to ERLPBoost. Similarly, the Corrective Binary ERLPBoost algorithm, described in Chapter 4.3, is directly comparable to Binary ERLPBoost.

In comparing the corrective and totally corrective algorithms, there are four primary questions we want to ask. First, is the corrective algorithm faster than the corresponding totally corrective algorithm overall? Second, is Corrective ERLPBoost faster than Corrective Binary ERLPBoost? Third, do the corrective algorithms generalize better than the totally corrective algorithms? Fourth, are the master hypotheses returned by the totally corrective algorithms smaller than those of the corrective algorithms? Smaller master hypotheses are sometimes preferred because they are easier to interpret and because they are more efficient to apply to large amounts of test data.

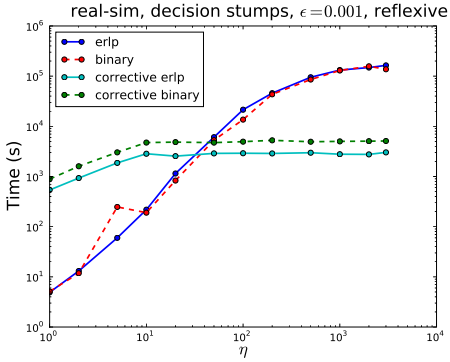
First we compare the total training time for the corrective and totally corrective algorithms. One of the primary motivations for the corrective algorithms is that at each iteration they perform a simple exponential update instead of solving an optimization



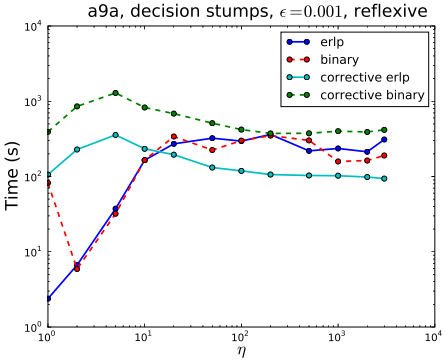
(a)



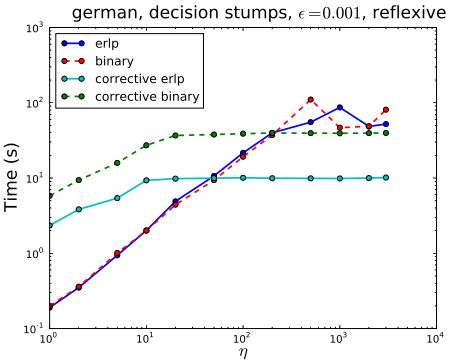
(b)



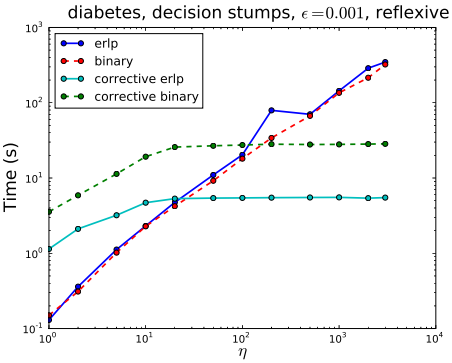
(c)



(d)

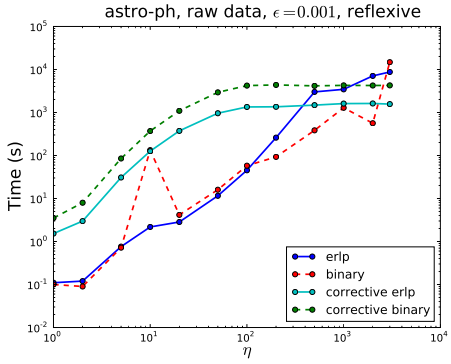


(e)

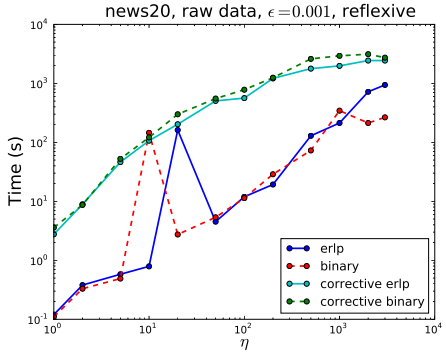


(f)

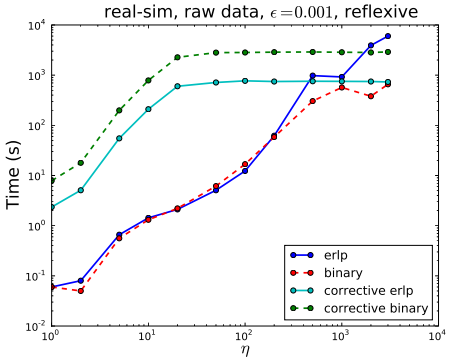
Figure 5.19: **Time vs. η** with **decision stump** hypotheses.



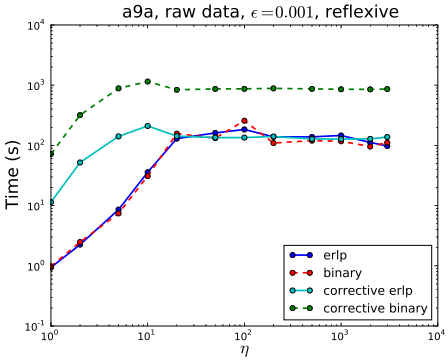
(a)



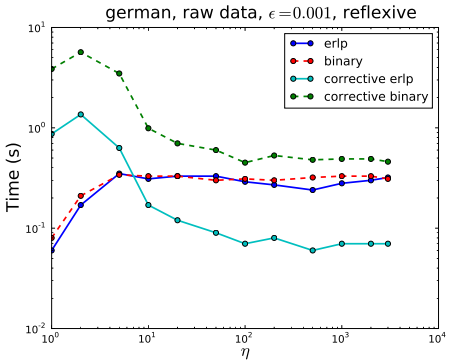
(b)



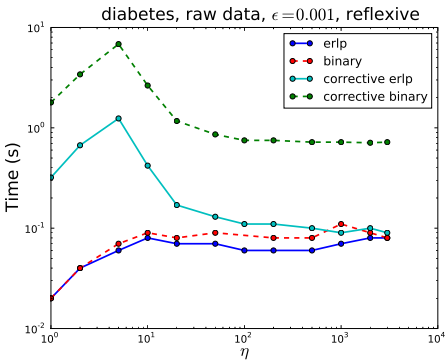
(c)



(d)



(e)



(f)

Figure 5.20: **Time vs. η** with **raw data** hypotheses.

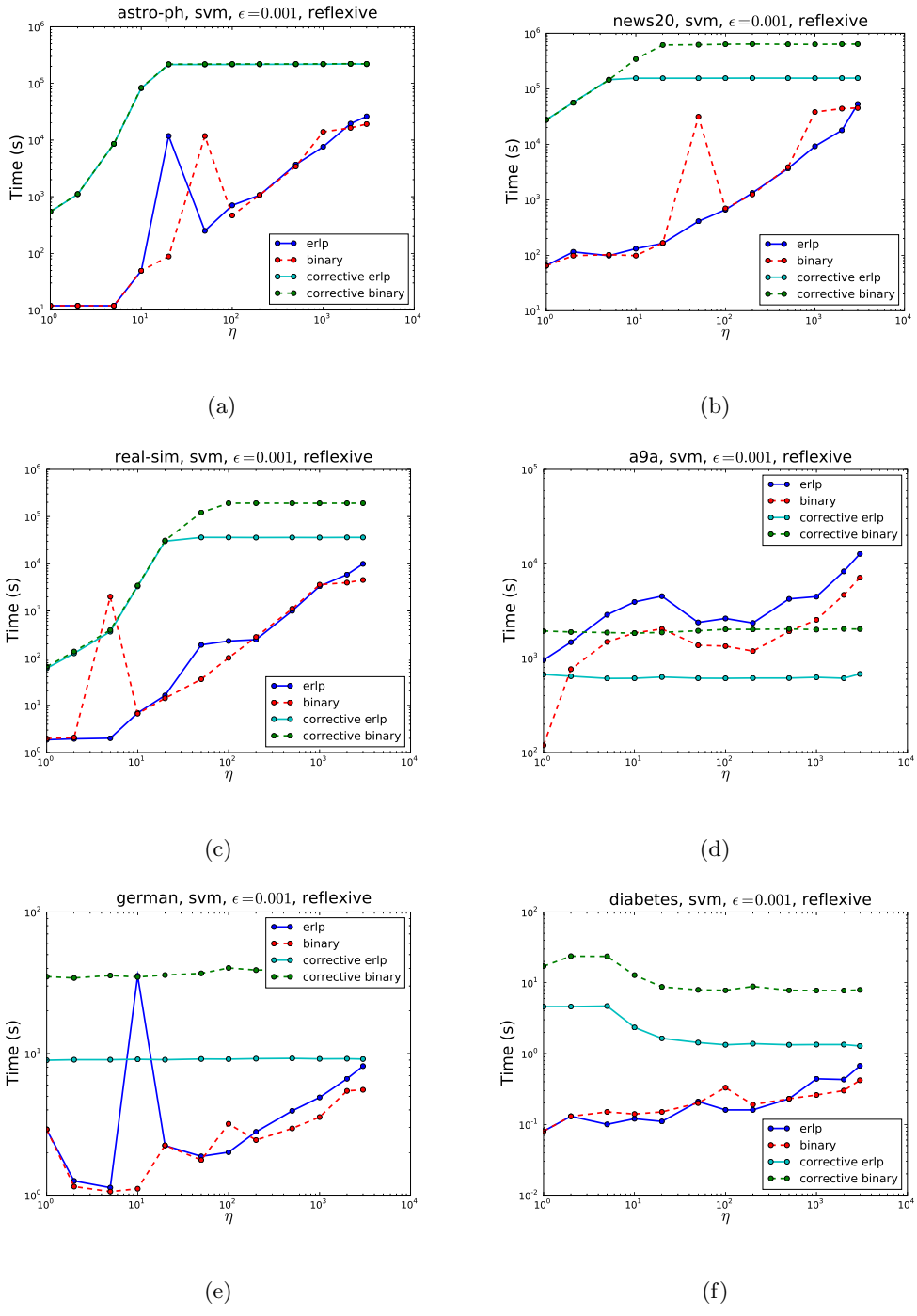


Figure 5.21: **Time vs. η** with **SVM** hypotheses.

problem [77]. As a result, a single iteration of the corrective algorithms should be faster than a single iteration of the totally corrective algorithms. However, the corrective algorithms should require far more iterations than the totally corrective algorithms. Clearly the overall difference in training time between corrective and totally corrective algorithms will depend on both the relative speed of a single iteration and the relative number of iterations.

In this experiment, we plot the training time of each of the four algorithms as a function of η. The results depend heavily on the hypothesis class. Figure 5.19 plots total execution time as a function of η for decision stump hypotheses. The totally corrective algorithms are always faster for low η, which corresponds to high regularization. However, as η increases, the number of iterations also increases, and the totally corrective algorithms become slower than the corrective algorithms on every data set except *a9a*. The reason is that the optimizer used by the totally corrective algorithms struggles with decision stumps, particularly in later iterations. Figure 5.20 plots training time as a function of η for raw data hypotheses. In this case the totally corrective algorithms are generally faster than the corrective algorithms except for the *german* data set, where the results are mixed. For SVM hypotheses, shown in Figure 5.21, the results are even more dramatic. For every data set except *a9a*, the totally corrective algorithms are significantly faster than the corrective algorithms. The optimizer does not struggle with these hypotheses the way it did with decision stumps. However, the primary reason that the totally corrective algorithms have a competitive advantage with SVM hypotheses is that the SVM oracle takes longer. The more time-consuming it is

to find the next hypothesis, the more it pays to invest in a slower but more effective update. This result is especially important because, as we will establish shortly, the SVM hypotheses have the lowest overall generalization error.

In addition, Figures 5.19, 5.20, and 5.21 allow us to compare the training time of Corrective ERLPBoost and Corrective Binary ERLPBoost. Recall from Chapter 4.3 that the primary difference between the two algorithms is the normalization mechanism. Corrective ERLPBoost is able to use the projection method described in Chapter 4.2, and it must enforce the normalization constraint via a binary search. We therefore expect Corrective Binary ERLPBoost to be slower, and this conjecture is confirmed by our experiments.

Overall, Corrective ERLPBoost is generally faster than Corrective Binary ERLPBoost. For decision stumps, shown in Figure 5.19, we can see that the only time Corrective Binary ERLPBoost is faster is on the *astro-ph* data set when $\eta = 2000$. For SVM hypotheses, shown in Figure 5.21, Corrective Binary ERLPBoost ties Corrective ERLPBoost on *astro-ph*, but it is slower on the other data sets. Finally, Figure 5.20 shows that for raw data hypotheses, Corrective Binary ERLPBoost is always significantly slower.

The next question we will address is whether the corrective algorithms generalize better than the totally corrective algorithms. This is significant because generalization error is the single most important criterion by which we judge machine learning algorithms. In this experiment, we plot generalization error as a function of η for each of the four algorithms. This serves to establish whether the trend holds across a broad

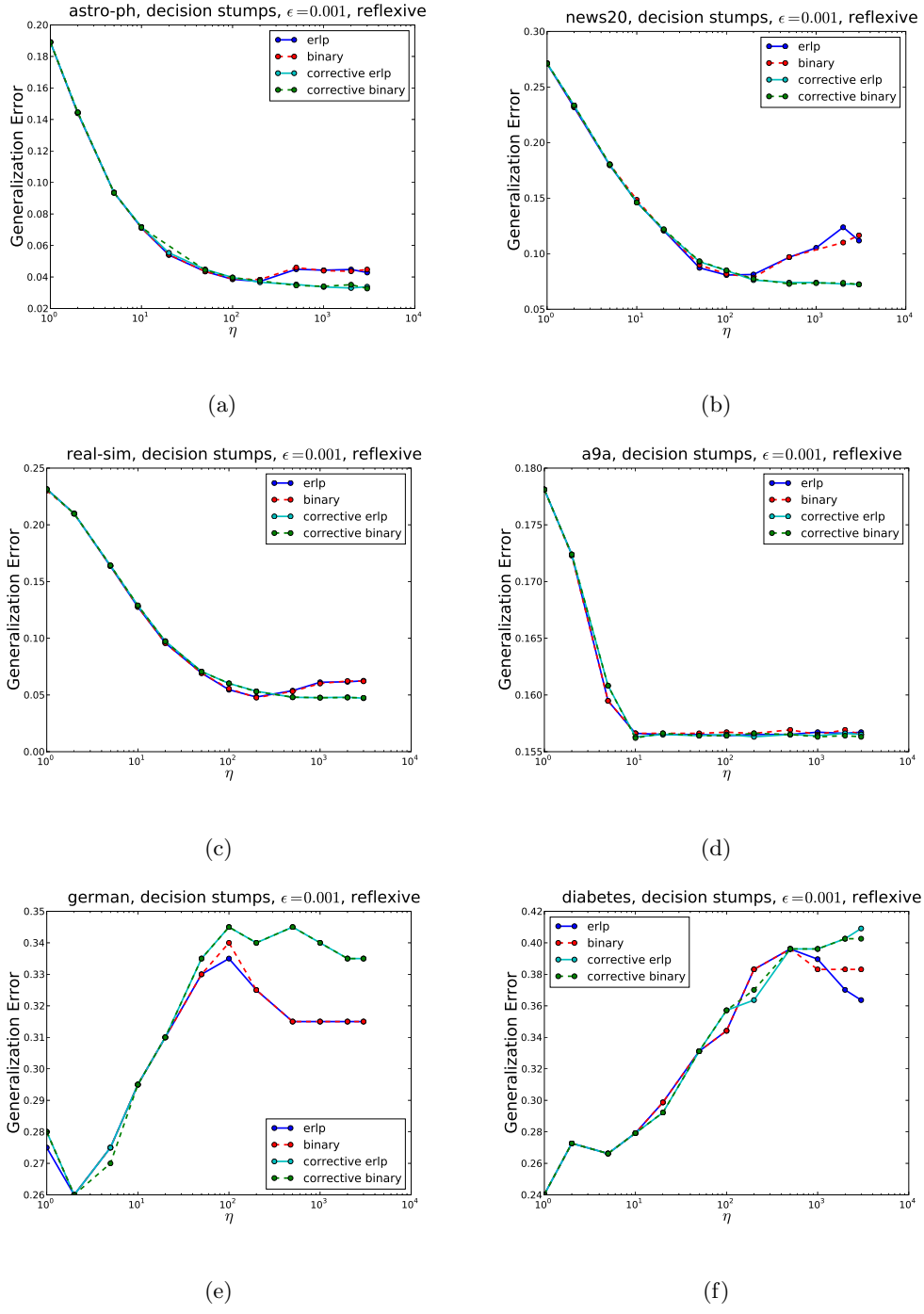


Figure 5.22: **Generalization error vs. η** with **decision stump** hypotheses.

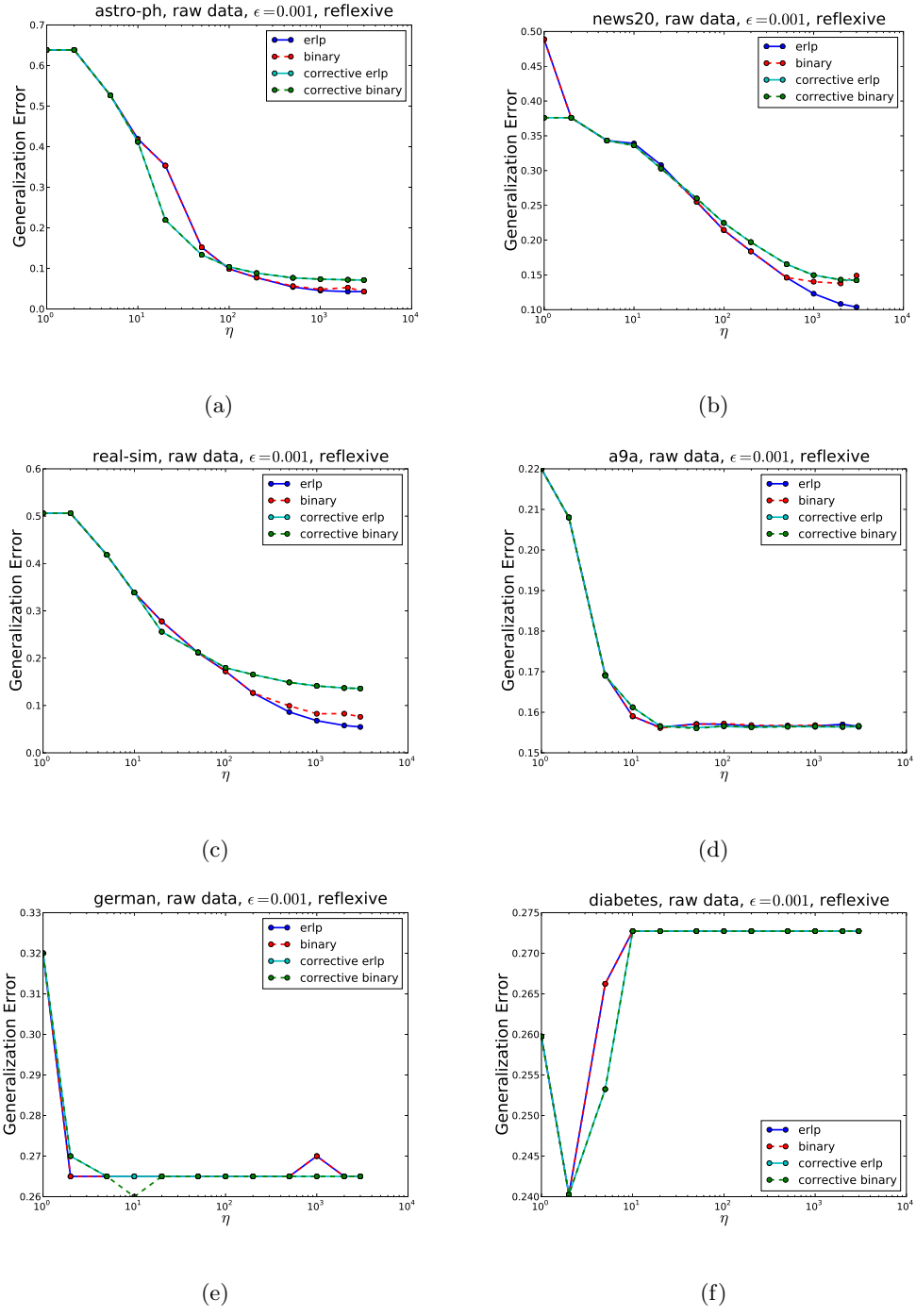


Figure 5.23: **Generalization error vs. η** with **raw data** hypotheses.

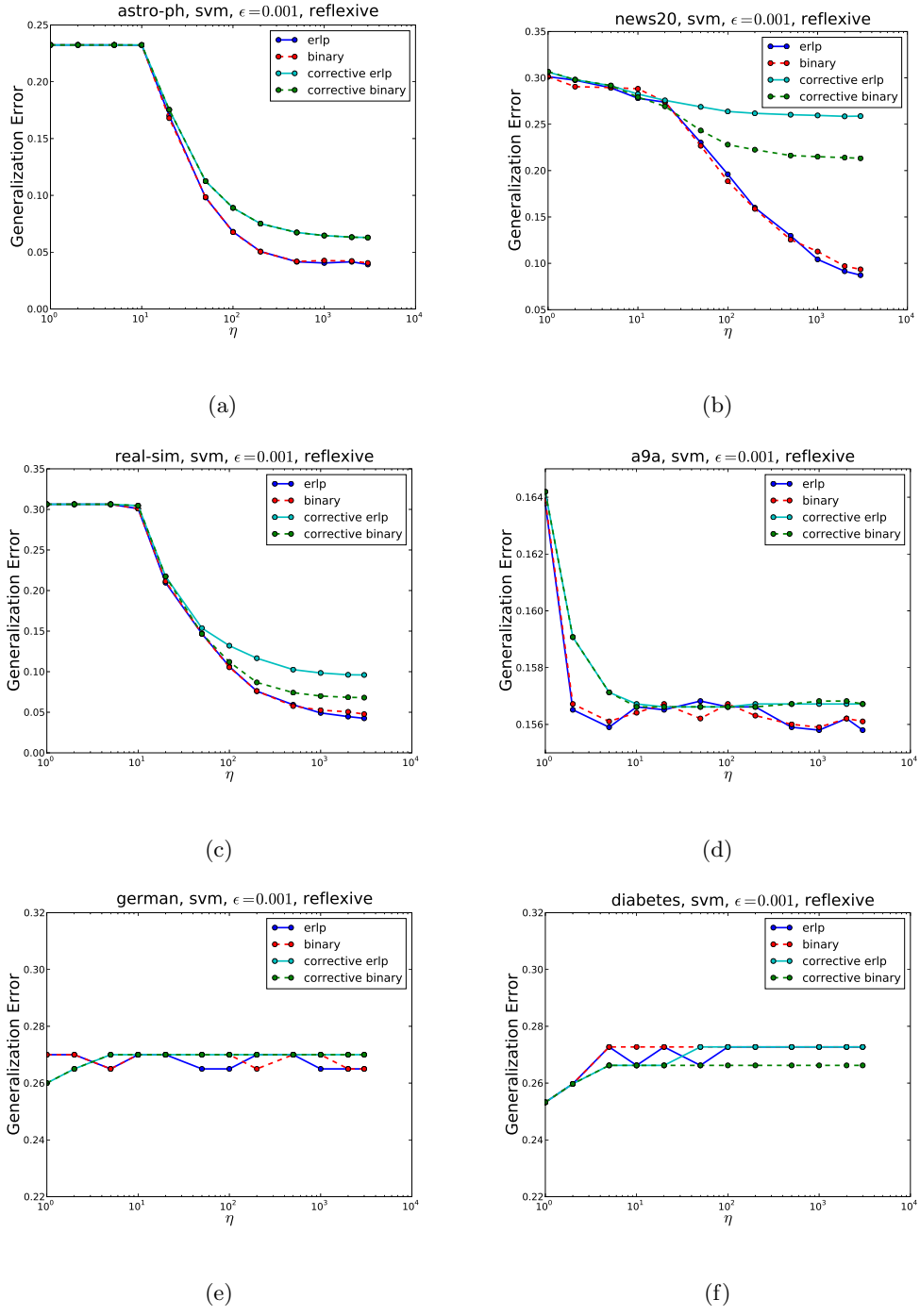


Figure 5.24: **Generalization error vs. η** with **SVM** hypotheses.

range of regularization parameters.

Again, the results seem to depend heavily on the hypothesis class. Figure 5.22 shows generalization error as a function of η for decision stumps. The corrective and totally corrective algorithms are tied when η is small, but as η increases, the corrective algorithms are at least as good as the totally corrective algorithms on all but the *german* and *diabetes* data sets. For raw data hypotheses, shown in Figure 5.23, the corrective and totally corrective algorithms are also tied when η is small. However, as η increases, the totally corrective algorithms beat the corrective algorithms on three data sets and tie on the other three. It is with SVM hypotheses, shown in Figure 5.24, that the totally corrective algorithms are truly dominant. They outperform their corrective counterparts on *astro-ph*, *news20*, *real-sim*, and *a9a*. There is no significant difference between the algorithms on the remaining two data sets. Observe that the SVM hypotheses generalize better overall than the other hypotheses, so the fact that the totally corrective algorithms beat the corrective algorithms with SVM hypotheses is particularly important.

We now compare the number of unique hypotheses and the number of iterations of each algorithm. For each algorithm, we plot the number of unique hypotheses as a function of η. On the same figure, we also plot the number of iterations as a function of η, but only for the corrective algorithms. The reason is that the totally corrective algorithms never select the same hypothesis twice. Therefore, the number of hypotheses is equal to the number of iterations, and it suffices to show only one of those quantities. In contrast, the corrective algorithms are perfectly free to choose the same hypothesis

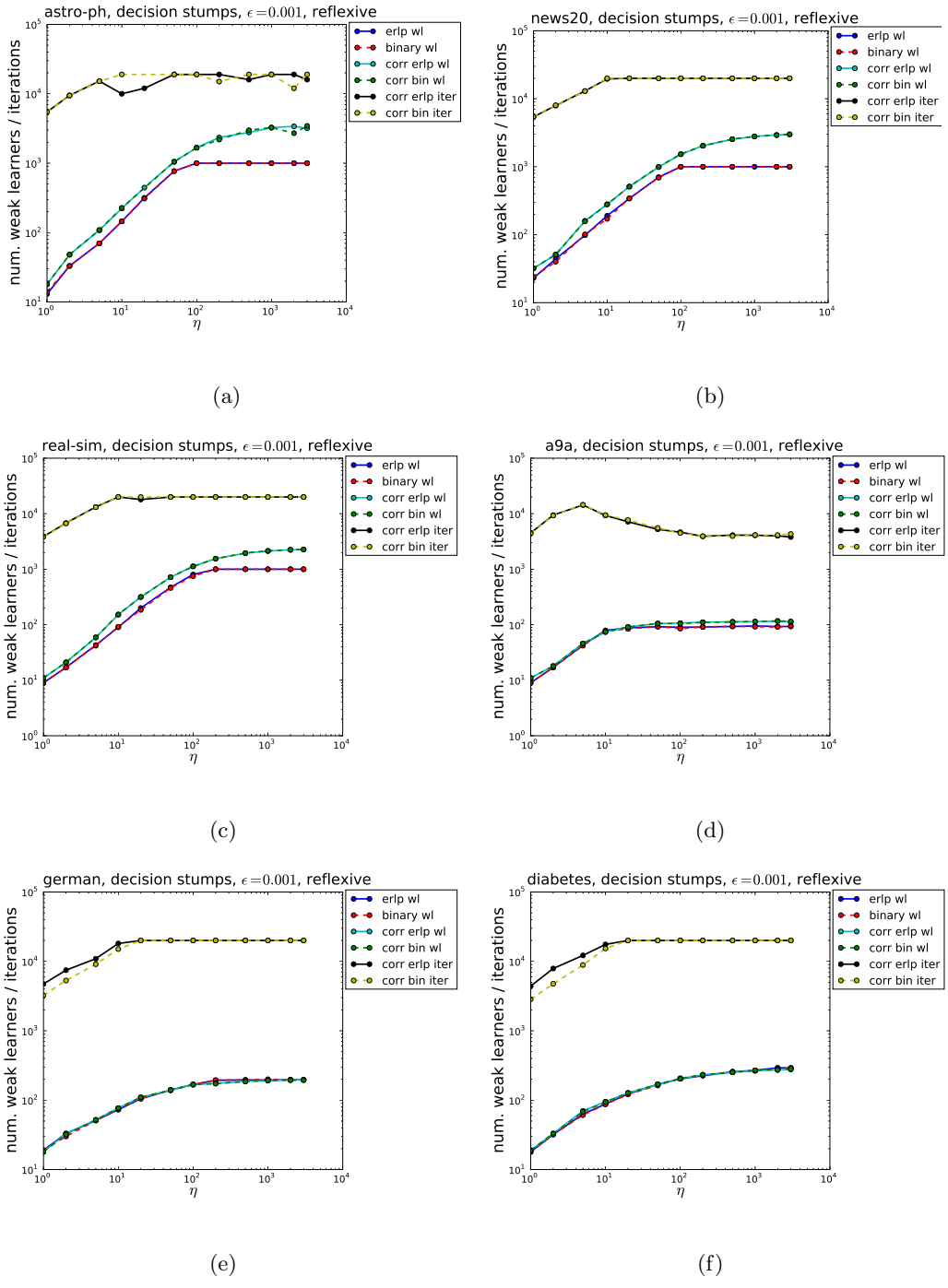


Figure 5.25: **Number of hypotheses vs. η** with **decision stump** hypotheses.

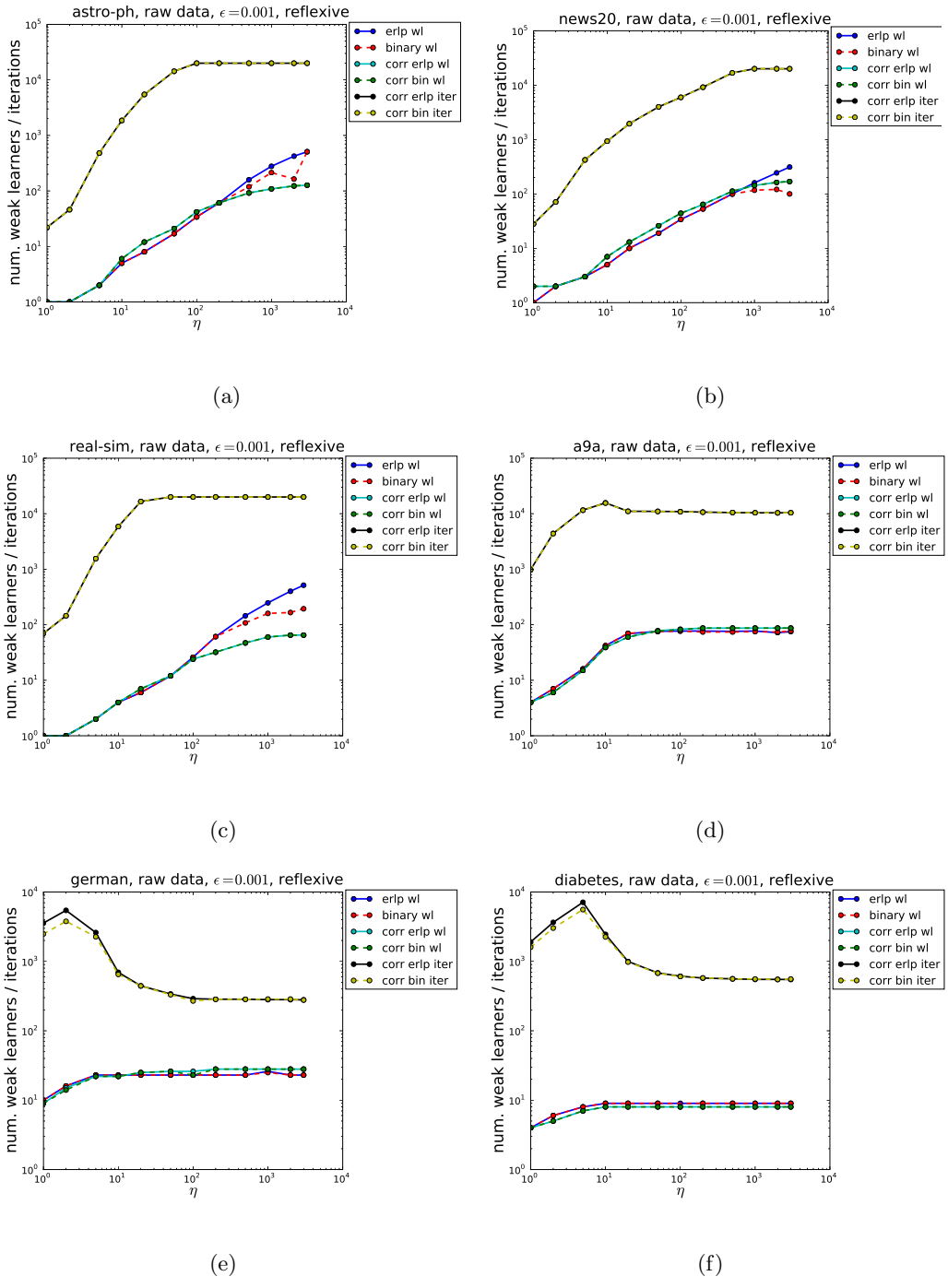


Figure 5.26: **Number of hypotheses vs. η** with **raw data** hypotheses.

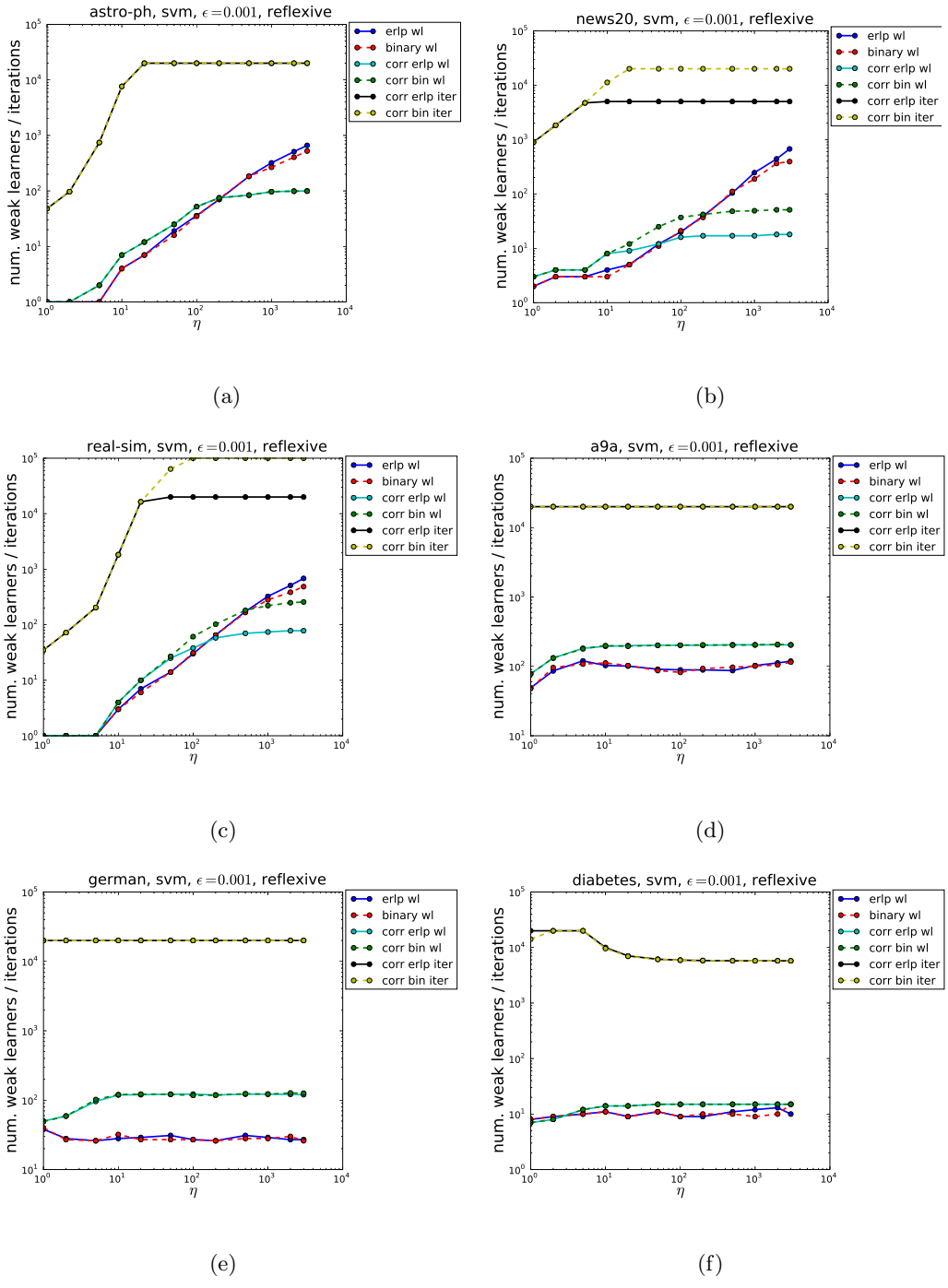


Figure 5.27: **Number of hypotheses vs. η** with **SVM** hypotheses.

multiple times. In fact, this is a common behavior.

The results are shown in Figures 5.25, 5.26, and 5.27. Two interesting results can be inferred from these plots. First, the corrective algorithms require an order of magnitude more iterations than the totally corrective algorithms, sometimes two. Second, the number of unique hypotheses selected by the corrective algorithms is much lower than the number of iterations. It is always the same order of magnitude as the number selected by the totally corrective algorithms. This makes intuitive sense because the corrective algorithms can choose the same hypotheses multiple times, so the number of hypotheses should be much smaller than the number of iterations.

In addition, we can observe a relationship between the number of unique hypotheses and generalization error. Let us compare Figure 5.22 with Figure 5.25 for *astro-ph*, *news20*, and *real-sim*. These figures show the number of hypotheses and the generalization error for decision stump hypotheses. By comparing these figures, we can see that the algorithms that find the most hypotheses also have the best generalization performance. The same is true when we compare Figures 5.23 and 5.26 for raw data and Figures 5.24 and 5.27 for SVM hypotheses. It is harder to make this comparison with the *a9a*, *diabetes*, and *german* data sets because the difference in generalization error between the algorithms is often insignificant. While it is probably incorrect to draw the sweeping conclusion that more hypotheses result in better generalization, the correlation nonetheless merits a remark.

In summary, the most significant contributor to training time for the corrective algorithms is the time it takes the oracle to return a hypothesis. This is because

the corrective algorithms require far more iterations than their totally corrective counterparts. For the totally corrective algorithms, the most significant contributor to the training time is the optimization problem solved at each iteration. These optimization problems have not been studied extensively before, and improving them would make the totally corrective algorithms even more competitive. Most importantly, the best overall generalization is achieved by the SVM hypotheses and in this case, the totally corrective algorithms are faster and have better generalization error.

5.5 Corrective vs. Totally Corrective Algorithms at Higher Precision

In the previous section, we found many instances where the corrective algorithms converge more quickly than the totally corrective algorithms. Recall that all of these experiments were done with precision parameter $\epsilon = 0.001$. In this section, we seek to determine whether the totally corrective algorithms converge more quickly than the corrective algorithms for smaller values of ϵ. We conjecture that for sufficiently small ϵ, the totally corrective algorithms will converge in both fewer iterations *and* less time than the corrective algorithms. To test this conjecture, we set $\epsilon = 10^{-5}$ - two orders of magnitude smaller than it was before. We then compared ERLPBoost with Corrective ERLPBoost on our two smallest data sets: *diabetes* and *german*. It is not currently possible to get ERLPBoost to converge to sufficiently high precision on the larger data sets.

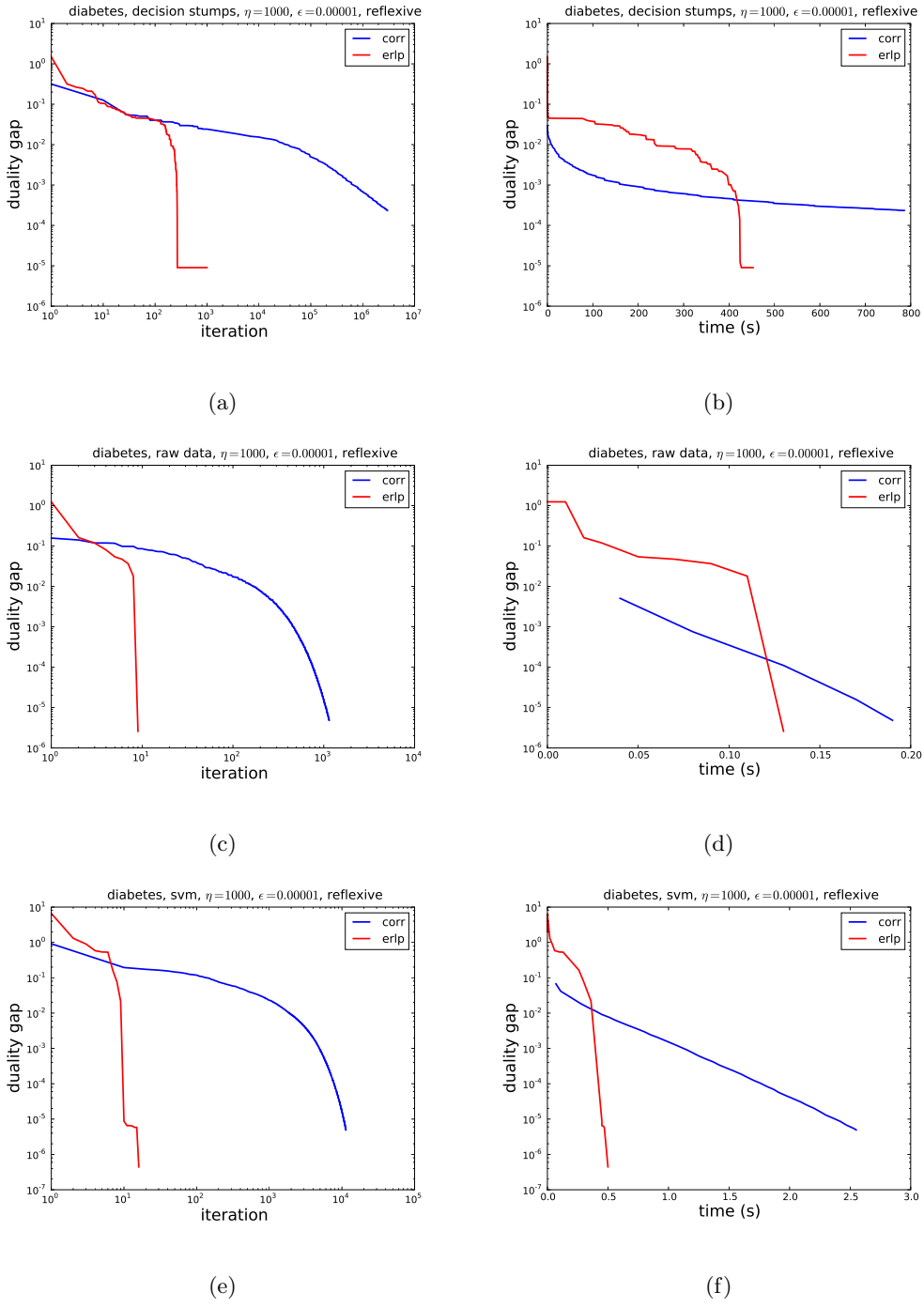
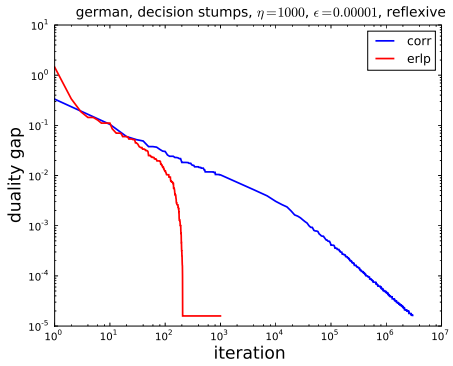
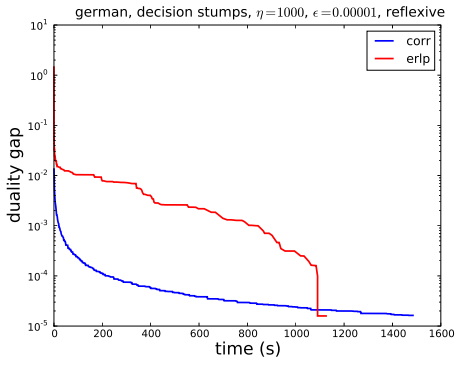


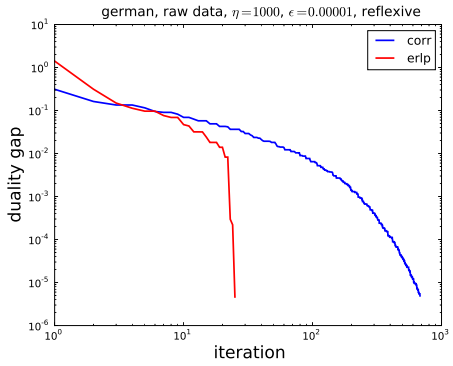
Figure 5.28: Duality gap vs. time and iteration for corrective vs. totally corrective algorithms for *diabetes* data set with $\epsilon = 10^{-5}$



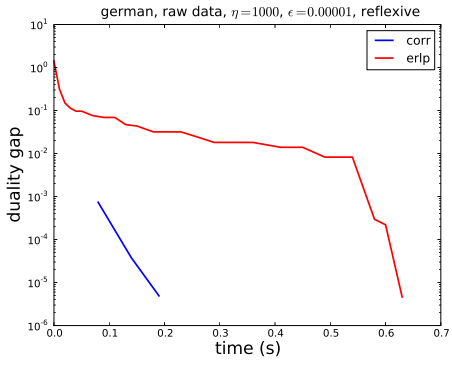
(a)



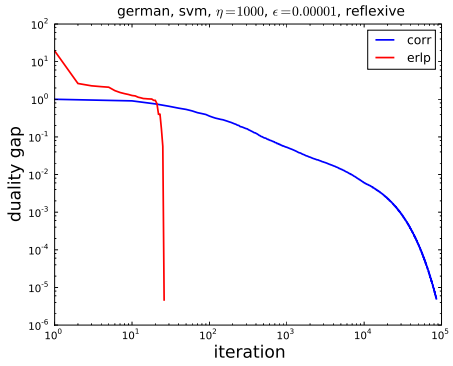
(b)



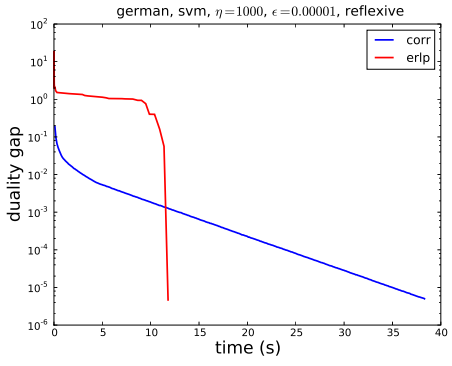
(c)



(d)



(e)



(f)

Figure 5.29: Duality gap vs. time and iteration for corrective vs. totally corrective algorithms for *german* data set with $\epsilon = 10^{-5}$

Recall that both ERLPBoost and Corrective ERLPBoost converge when the duality gap is less than $\epsilon/2$. Comparing the duality gaps of the two algorithms is a good way to visualize their relative convergence behavior. Figure 5.28 plots the duality gap as a function of iteration and time side by side for each hypothesis class on the *diabetes* data set. When $\epsilon = 10^{-5}$, ERLPBoost converges in fewer iterations than Corrective ERLPBoost. Moreover, ERLPBoost also requires less time to converge. The results are similar for the *german* data set. Figure 5.29 plots the duality gap as a function iteration and the duality gap as a function of time for the *german* data set for all three hypothesis classes. In this case, ERLPBoost converges in less time than Corrective ERLPBoost for decision stump and SVM hypotheses, but not for raw data hypotheses. Interestingly, even decreasing ϵ to 10^{-8} will not cause ERLPBoost to converge faster than Corrective ERLPBoost for raw data hypotheses on the *german* data set.

5.6 Overall Comparison of Algorithms

In this section, we analyze the best achievable performance of each algorithm. This is the experimental question that is of the most interest to practitioners. For each hypothesis class, we report the best generalization error of each algorithm on each data set over all parameter settings that we explored. In addition, we report the training time associated with the best overall results.

The algorithms included in this comparison are ERLPBoost, Binary ERLPBoost, Corrective ERLPBoost, Corrective Binary ERLPBoost, and AdaBoost. Ad-

aBoost was chosen as a comparator because it is by far the most well known boosting algorithm. Also, we included linear SVMs in the comparison for SVM hypotheses. In fact, the reason we introduced SVM hypotheses is because the master hypothesis returned by a linear SVM is a linear combination of SVM hypotheses. Thus, SVM hypotheses enable us to make a direct comparison between boosting and linear SVMs. We did not include LPBoost in these comparisons because we were not able to run it on comparable parameters. Since LPBoost is extremely unstable on these data sets it is unlikely to be a strong competitor anyway.

The experiments in this section use the train-validation-test methodology. For each algorithm and each data set, we used the training set to find a master hypothesis. To determine the best parameter values, we applied the master hypothesis to the validation data. The parameter settings we used are described in Table 5.3. We obtained our final result by applying the master hypothesis trained with the best parameter values to the test data. The best overall parameters values are reported in Tables 5.11, 5.10 and 5.12.

Generalization error is the most important criterion by which machine learning algorithms are judged. We now report the best overall generalization error for our experiments. For each hypothesis class, we present a table with the best result achieved for each data set by each algorithm. For each data set, the best result is in bold font.

First we show the best achievable generalization error using decision stump hypotheses in Table 5.4. In the case of decision stumps, the best result is always achieved by either Corrective ERLPBoost or Corrective Binary ERLPBoost. Nevertheless, there

alg / data set	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	3.85	8.43	4.82	15.60	28.50	**24.03**
bin	3.42	8.18	4.76	15.65	27.50	**24.03**
corr	3.30	7.25	**4.72**	15.65	**25.00**	**24.03**
corr bin	**3.27**	**7.22**	4.75	**15.54**	26.00	**24.03**
ada	3.44	7.25	**4.72**	15.65	30.50	34.42

Table 5.4: Lowest overall **generalization error** of each algorithm for **decision stump** hypotheses. The best result for each data set is in bold.

alg / data set	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	**3.67**	**9.75**	**5.12**	15.61	27.00	26.62
bin	3.98	12.93	7.41	**15.59**	26.50	26.62
corr	7.12	14.22	13.53	15.65	27.50	**25.97**
corr bin	7.12	14.22	13.59	15.69	26.50	26.62
ada	10.23	24.10	16.28	15.64	**26.00**	**25.97**

Table 5.5: Lowest overall **generalization error** of each algorithm for **raw data** hypotheses. The best result for each data set is in bold.

alg/data set	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	3.41	7.28	4.25	**15.62**	26.50	**24.03**
bin	**3.20**	7.35	3.58	15.64	26.50	25.97
corr	6.26	21.35	6.73	15.65	26.50	25.97
corr bin	6.26	21.35	6.73	15.65	26.50	24.68
SVM	3.21	**3.26**	**3.29**	**15.62**	**26.00**	25.96

Table 5.6: Lowest overall **generalization error** of each algorithm for **SVM** hypotheses. The best result for each data set is in bold.

are some notable ties. For example, AdaBoost is tied for the lowest generalization error on the *real-sim* data set and all algorithms except AdaBoost are tied on the *diabetes* data set. The latter result is not surprising because the results for the *diabetes* data set have been similar across a broad range of algorithms and parameter settings.

We now show the best achievable generalization error for raw data hypotheses in Table 5.5. For raw data hypotheses, the ERLPBoost algorithm has the best overall performance. It beats every other algorithm on *astro-ph*, *news20*, and *real-sim* datasets and it is a close second on *a9a*. The second best algorithm in terms of overall performance is Binary ERLPBoost. AdaBoost is significantly worse than the other algorithms on the *astro-ph*, *news20*, and *real-sim* datasets, and comparable on *a9a*, *german*, and *diabetes*. At first this was surprising because *german* and *diabetes* are the noisiest datasets and AdaBoost is believed to perform badly on noisy data. However, *german* and *diabetes* have yielded very similar results across a broad range of algorithms and parameter settings so perhaps none of the algorithms we have tried are particularly well suited to these data sets.

Finally, we show the best achievable generalization error for SVM hypotheses in Table 5.6. This hypothesis class was introduced to enable a direct comparison between boosting algorithms and linear SVMs. Overall, the SVM has the lowest generalization error on four data sets. The only dataset where it does significantly worse than the boosting algorithms is *diabetes*. Still, this may be the result of using only a small number of data sets. Perhaps if we try a broader range of datasets with different characteristics we will find a class of problems that are better suited to boosting than

SVMs. Also observe that for each dataset except *german*, the best result for SVM hypotheses beats the best result for decision stumps and raw data. Also note that the totally corrective algorithms are at least as good as the corrective algorithms on every data set. Most importantly, the totally corrective algorithms are dramatically better than the corrective algorithms on the *astro-ph*, *news20*, and *real-sim* datasets.

Given two algorithms with similar generalization error, we would prefer the faster one. Also, many end users would be willing to sacrifice a small amount of generalization performance for a marked reduction in training time. For this reason, it is also useful to compare the time required to achieve the best overall generalization result of each algorithm.

First we show the training time of each algorithm for decision stump hypotheses in Table 5.7. One every data set except *real-sim* and *german*, the totally corrective algorithms have the fastest overall time. Still, it is worth noting that the corrective algorithms are faster by nearly a factor of ten on the *real-sim* data set. In general, the training time for the totally corrective algorithms is dominated by the optimizer, which struggles with decision stumps. Finally, it is worth noting that Corrective Binary ERLPBoost is slower than Corrective ERLPBoost on all data sets. This is true for the other two hypothesis classes as well.

We now show the training time of each algorithm for raw data hypotheses in Table 5.8. Although its generalization error was poor, AdaBoost was the fastest on three out of six datasets. The AdaBoost update is both the fastest but also the least sophisticated, so this is not surprising. The comparison between corrective and totally

alg/dataset	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	**4693.00**	11563.00	45821.00	536.10	1.80	**0.10**
bin	12167.00	**8535.80**	43449.00	**149.10**	2.10	**0.10**
corr	8143.30	10194.00	3025.40	194.90	**0.00**	1.10
corr bin	10302.00	11063.00	5100.90	456.90	13.00	3.50
ada	5507.71	15846.90	**2561.14**	350.30	6.99	3.32

Table 5.7: **Training time** in seconds for best result of each algorithm for **decision stump** hypotheses. The fastest time for each dataset is in bold.

alg/dataset	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	6524.40	1737.30	9145.10	151.60	0.10	0.40
bin	1600.30	**237.50**	503.20	114.00	0.30	**0.00**
corr	1566.90	2436.20	828.00	134.70	**0.00**	**0.00**
corr bin	4268.00	2724.00	2902.60	1458.50	0.90	19.20
ada	**1101.09**	1827.55	**478.00**	**75.72**	22.03	5.77

Table 5.8: **Training time** in seconds for best result of each algorithm for **raw data** hypotheses. The fastest time for each dataset is in bold.

alg/dataset	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	6540.00	22806.00	9965.60	8312.20	11.00	41.30
bin	29803.00	95984.00	18128.00	3844.60	81.60	**0.10**
corr	21870.00	64028.00	19154.00	16117.00	38.30	**0.10**
corr bin	22393.00	64028.00	19154.00	16117.00	38.30	24.10
svm	**18**	**60**	**30**	**3.6**	**2.05**	1.11

Table 5.9: **Training time** in seconds for best result of each algorithm for **SVM** hypotheses. The fastest time for each dataset is in bold.

corrective algorithms is mixed. Corrective ERLPBoost was faster than ERLPBoost on every dataset except *news20*. At the same time, Binary ERLPBoost was faster than Corrective Binary ERLPBoost on every data set.

Finally, we show the training time of each algorithm for SVM hypotheses in Table 5.9. Observe that SVMs are dramatically faster than any of our boosting algorithms. However, the SVM optimization problem has been studied extensively for over a decade while the ERLPBoost optimization problem has not. It is likely that part of the disparity comes from comparing a mature technology with a nascent one. Also, boosting algorithms must solve an optimization problem at each iteration while the SVM problem is only solved once. Thus, even if the two optimization problems could be solved in the same amount of time, ERLPBoost would be slower overall unless the optimal solution includes very few hypotheses. This table also shows that the totally corrective algorithms are faster than the corrective algorithms for SVM hypotheses. Recall that SVM hypotheses had the best overall generalization error and that the totally corrective algorithms generalized better than the corrective algorithms. Theses two results combine to form a powerful argument in favor of totally corrective algorithms.

alg/dataset	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	0, 21899	.2, 3219	0, 21356	.3, 2408	.5, 139 $*$	0, 12262
bin	.1, 6605	0, 20785	0, 23356	.2, 5219	.5, 3386	0, 14262
corr	0, 21899	0, 18785	0, 21356	0, 20571	.4, 1833	0, 12262
corr bin	0, 23899	0, 20785	0, 23356	.1, 6605	.5, 339 $*$	0, 14262

Table 5.10: **Parameters** for best result of each algorithm for **decision stump** hypotheses. For each algorithm and dataset, we report ν/N, η. By default, $\epsilon = 0.001$ except for the $*$ entries, which have $\epsilon = 0.01$. The value of η depends on N/ν, which is set independently of N, so many η values are the same. Here, $\nu/N = 0$ indicates that $\nu = 1$.

alg/dataset	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	.1, 4605	.1, 4605	.1, 4605	.2, 3219	0, 1279 $*$	.6, 1022
bin	.1, 6605	.2, 5219	.1, 6605	.2, 5219	.2, 5219	.6, 3022
corr	0, 21899	0, 18785	.1, 4605	0, 20571	.3, 241 $*$	.3, 241 $*$
corr bin	0, 23899	0, 20785	0, 23356	.3, 4408	.2, 5219	.6, 3022

Table 5.11: **Parameters** for best result of each algorithm for **raw data** hypotheses. For each algorithm and dataset, we report ν, η. By default, $\epsilon = 0.001$ except for the $*$ entries, which have $\epsilon = 0.01$. The value of η depends on N/ν, which is set independently of N, so many η values are the same. Here, $\nu/N = 0$ indicates that $\nu = 1$.

alg/dataset	*astro-ph*	*news20*	*real-sim*	*a9a*	*german*	*diabetes*
erlp	.1, 4605	.1, 4605	0, 21356	0, 20571	0, 1.27e6	.4, 1833
bin	.1, 6.6e5	.2, 5.2e5	.1, 6.6e5	.3, 4408	0, 1.48e6	0, 1.4e6
corr	0, 21899	0, 18785	0, 21356	0, 2.06e6	0, 1.28e6	.4, 1833
corr bin	0, 23899	0, 20785	0, 23356	0, 2.26e6	0, 1.48e6	.5, 3386

Table 5.12: **Parameters** for best result of each algorithm for **SVM** hypotheses. For each algorithm and dataset, we report ν/N, η. By default, $\epsilon = 0.001$ except for the $*$ entries, which have $\epsilon = 0.01$. The value of η depends on N/ν, which is set independently of N, so many η values are the same. Here, $\nu/N = 0$ indicates that $\nu = 1$.

Chapter 6

Related Work

Boosting is a a class of algorithms that has been extensively studied, and in placing our work within the context of the literature, we discuss five different areas of boosting research. We begin with a review of the fundamentals of boosting, including a basic discussion of AdaBoost, the most popular boosting algorithm. Next, we review the research on margin maximization and the convergence properties of AdaBoost. We follow this with a review the previous work on boosting algorithms that are robust to noise. We then review corrective and totally corrective boosting algorithms. Finally, we review significant experimental results for boosting.

6.1 The Basics of Boosting

One of the questions posed within the PAC framework [81] is whether a weak learning algorithm (or oracle) that returns a hypothesis that is only marginally better than random guessing can be used to create a strong learning algorithm, which is highly

accurate [41]. The first algorithm to show the equivalence between weak learnability and strong learnability was discovered by Schapire [72]. The was the first *boosting* algorithm because it boosted a weak learning algorithm into a strong learning algorithm. This is a relatively complicated algorithm using a recursive construction, and so the question was whether a linear combination of weak hypotheses could constitute a boosting algorithm. The first such algorithm was Boost by Majority [31]. Although simple and elegant, Boost by Majority was not practical because it required the weak guarantee g to be known ahead of time (see the definition of guarantee g in Chapter 2). Algorithms that do not require g to be known ahead of time are called *adaptive*. The first adaptive boosting algorithm was AdaBoost [30]. Some very good introductory discussion of boosting can be found in [57, 73].

AdaBoost is a practical, efficient algorithm that works by maintaining a distribution $\mathbf{d}$ on the examples. At each iteration, this distribution is used to find a hypothesis that has high weighted accuracy with respect to distribution $\mathbf{d}$. When hypothesis $h^t \in \pm 1$, it is given weight

$$w_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

where ϵ_t is the weighted error of hypothesis t. When $h^t \in [-1.1]$, then $\mathbf{w}_t$ is chosen via line search. The distribution $\mathbf{d}^t$ is updated so that it puts more weight on the examples that are harder to classify:

$$d_n^t = d_n^{t-1} \exp(-w_{t-1} y_n h^{t-1}(x_n))/Z_t,$$

where Z_t is the normalization factor that ensures that $\mathbf{d}^t$ is a probability distribution.

This distribution is a key component of all modern boosting algorithms. What differentiates them is how they compute $\mathbf{d}$ and how they compute the weights on the hypotheses $\mathbf{w}$. After T iterations, AdaBoost returns a master hypothesis that is a linear combination of weak hypotheses. The master hypothesis takes the form $f_{\mathbf{w}}(\mathbf{x}) = \sum_{q=1}^{T} w_q h^q(\mathbf{x})$ and the sign of the master hypothesis is the prediction.

It was shown in [34] that AdaBoost is an additive logistic regression algorithm that uses Newton's method to optimize the loss function

$$\frac{1}{N}\sum_{n=1}^{N}\exp(-y_n f_{\mathbf{w}}(\mathbf{x}_n)).$$

Note that $y_n f_{\mathbf{w}}(\mathbf{x}_n)$ is the margin of example $\mathbf{x}_n$ achieved by the master hypothesis. Boosting algorithms can also be generalized to optimize other loss functions, which may actually result in more effective algorithms [54]. However, not all potential functions result in boosting algorithms. Duffy and Helmbold [25] were able to establish necessary and sufficient conditions for potential functions to result in boosting algorithms.

A different interpretation of AdaBoost is that at each iteration, it minimizes a relative entropy to the current distribution subject to the linear constraint that the edge of the last hypothesis is non-negative [43, 45]. All boosting algorithms whose iteration bounds are logarithmic in the number of examples are motivated by either optimizing a sum of exponentials in the hypothesis domain or a cost function that involves a relative entropy in the example domain. As we discussed in Chapter 3, both motivations lead to the same algorithms and are dual views of the same underlying optimization problems.

Boosting algorithms can also be extended to cover richer hypothesis classes and

more complex cost structures. The original AdaBoost algorithm handles only simple predictions of ± 1. Schapire and Singer proposed an improvement to this algorithm that can handle confidence-rated predictions [74]. There are also boosting algorithms that use asymmetric loss functions [52, 28] to handle the case where it is more costly to mispredict one class than another.

Boosting has been successfully applied to many practical problems, which fuels its popularity. The first experimental result for AdaBoost was on an optical character recognition problem (OCR) [32]. Boosting has also been applied to text classification [75]. In the imagery domain, popular applications include object detection [27, 82, 59, 80, 46], face detection [83], and image retrieval [79].

6.2 Margin Maximization and Boosting

Multiple studies have observed that allowing AdaBoost to continue running even after it achieves consistency on the training examples results in improved generalization [24, 61, 11]. However, each iteration of AdaBoost increases the complexity of the resulting master hypothesis. Because overly complex models are prone to overfitting, this improvement in generalization error is surprising. Schapire et al. [71] attribute this to the observation that allowing AdaBoost to continue for additional iterations improves the margin. This observation is reinforced by the existence of bounds on the generalization error of the master hypothesis that improve as the margin of the master hypothesis increases [44, 53].

There are two problems with this line of reasoning. First, Rudin et al. [68] proved that there are cases where AdaBoost does not maximize the margin. Second, the generalization error bounds of [71] depend on the *entire* empirical margin distribution, not the *minimum* margin. This point is emphasized in [67]. There are other generalization results for boosted classifiers in terms of the margin distribution. Mason et al. extend the upper bound on the generalization error to other cost functions of the margin [53]. This bound is independent of the dimension of the master hypothesis, though it does depend on the VC dimension of the hypothesis class. Koltchinskii et al. [44] developed a similar bound that depends on the empirical margin distribution and the Rademacher complexity of the hypothesis class.

Although AdaBoost does not maximize the margin, there are many boosting algorithms that do. Breiman proposed Arc-gv [10], which maximizes the margin, but only asymptotically. The first boosting algorithm to provably maximize the margin was AdaBoost* [66]. Since that time, there have been many boosting algorithms proposed that provably maximize either the margin or the soft margin. Examples include TotalBoost [91], SoftBoost [87] ERLPBoost [90], Binary ERLPBoost, and several others [69, 70, 63, 65, 77].

In Chapter 2 we explored the connection between margins, edges, and linear programming. Freund and Schapire [33] were the first to suggest the connection between boosting, game theory and von Neumann's minimax theorem [86]. In a similar vein, there is also a connection between margins and weak learnability. We say that a set of examples is *weak learnable* when $g > 0$. This is equivalent to saying that when any

distribution on the examples $\mathbf{d}$ is given to our oracle, it will return a hypothesis h that has edge at least $g > 0$. Shalev Schwartz and Singer pointed out that this is true if and only if the examples are linearly separable [77].

The LPBoost algorithm is motivated by direct margin maximization [21, 36]. For the current set of hypotheses, it computes the weights $\mathbf{w}$ to maximize the margin. This can be solved as a linear programming problem. The earliest variant of this algorithm was DualLPBoost [36], developed by Grove and Schuurmans. DualLPBoost only solves the hard margin problem. This algorithm was motivated by the margin theory, which was used to explain the success of AdaBoost. Grove and Schuurmans reasoned that if margins were fundamentally important, then DualLPBoost would be even more successful. Experimentally, they found that on real-world data, focusing exclusively on the minimum margin was actually detrimental to performance. Demiriz et al.[21] extended DualLPBoost to the soft margin case, and it is their algorithm that we employ extensively in this thesis. One of the more interesting results for LPBoost is that it directly minimizes the bound on the generalization error given in [17] in terms of the soft margin and the covering number.

The key point is that if margin maximization were the single most important contributor to good generalization performance, then LPBoost would be ideal. However, one of the primary results of this thesis is that directly maximizing the margin via the LPBoost optimization problem can result in a very brittle algorithm.

6.3 Boosting Algorithms for Noisy Data

Dietterich [22] was the first to observe that the classification performance of AdaBoost deteriorates markedly in the presence of noise. The reason is that AdaBoost's exponential update concentrates too much weight on examples that are the hardest to classify. When some examples are noisy, AdaBoost rapidly concentrates the weight on these examples. This causes the noisy examples dominate the hypothesis selection process.

A popular approach for dealing with noisy examples is to cap the amount of weight that can be put on any given example. In this thesis we call this *capping*. This approach originated in the SVM community [20]. In the boosting literature this is also referred to *smoothing* the distribution on the examples, and boosting algorithms that employ this technique are often called *smooth boosters*. DualLPBoost [36] is the first example of a smooth booster. Another early example comes from the work of Rätsch et al. [63], which presents two adaptations of AdaBoost that employ capping, LP-AdaBoost and QP-AdaBoost. All of the algorithms discussed in this thesis (LPBoost [21], SoftBoost [87], ERLPBoost [90], Binary ERLPBoost, Corrective ERLPBoost [77], and Corrective Binary ERLPBoost) use capping as well. Other examples of smooth boosting algorithms are SmoothBoost [76] and AdaFlat [35], though the iteration bounds on these algorithms are significantly weaker than those of the ERLPBoost family. MadaBoost[23] also fits loosely into this category because it caps the weights on the examples by not allowing them to exceed their initial probabilities. However, in

this case the weights on the examples are no longer a probability distribution. Another drawback of MadaBoost is that it requires edges of the hypotheses to be monotonically decreasing, something that is difficult to ensure in practice.

A different approach to making boosting robust to noise replaces the exponential loss function of AdaBoost with a loss function that penalizes misclassified examples more gently. One example is the truncated exponential loss function of MadaBoost[23]. A more popular example is the negative log likelihood function, also known as the logistic loss function. The most well-known algorithm that minimizes the logistic loss is LogitBoost. Note that LogitBoost requires the oracle to choose h^t by solving a weighted least squares regression problem rather than via the edge, and each hypothesis is given weight $w_t = 1/2$. The LogLossBoost [18] and FilterBoost [8] algorithms also provably minimize the logistic loss function. While it can be plausibly argued that the logistic loss is better than the exponential loss for noisy data, there is no way to adapt any of these algorithms to different noise rates, something that can easily be done with capping. Also, the two approaches are not mutually exclusive. Binary ERLPBoost is an algorithm that optimizes the logistic loss and employs capping as well.

All of the algorithms that we have seen so far optimize some convex function of the margin, but some boosting algorithms optimize loss functions that are not convex. BrownBoost [29] was the first algorithm of this type. Perhaps the most useful characteristic of BrownBoost is that it is able to give up on examples that are frequently misclassified by the weak hypotheses to focus on examples that have a better chance of being correctly classified. In contrast, algorithms that employ capping would still put

weight on those examples. In BrownBoost, the weights on the examples take the form of a Gaussian where the mean is related to the average margin and the variance is C, BrownBoost's only tunable parameter, which corresponds to the time the algorithm is allowed to run. Note that the mean changes with each iteration. In our experiments, SoftBoost and ERLPBoost outperform BrownBoost [90]. The same result was found in a recent experimental study by Arvey [2]. There is also a modification of BrownBoost known as *margin conscious BrownBoost* [2]. In its final iterations, the loss function of BrownBoost is dominated by examples of very small positive margin. Margin conscious BrownBoost [2] introduces an additional parameter, δ, which adds additional loss to examples with margin less than δ. This modification appears to outperform BrownBoost empirically.

The other significant boosting algorithm that does not optimize a convex loss function is MartiBoost [48]. Unlike the other boosting algorithms discussed so far, MartiBoost does not return a linear combination of hypotheses. The MartiBoost hypothesis is a directed graph. Each node of this graph corresponds to a hypothesis and the starting point of this graph is the first hypothesis. MartiBoost requires the hypotheses to have error rates that are balanced between false positives and false negatives and to be slightly better than random guessing, but in [48] it was shown that the balanced error rate requirement is achievable. Classifying an example corresponds to a random walk along this graph with constant step size, where each step is slightly biased toward correct classification. A drawback of MartiBoost is its requirement that its hypotheses predict ± 1, but this is remedied by Adaptive MartiBoost [47]. Adaptive MartiBoost in

an extension of MartiBoost that can take advantage of the varying quality of hypotheses by varying the step size in the random walk. Adaptive MartiBoost can also handle confidence-rated hypotheses.

6.4 Corrective and Totally Corrective Algorithms

In Chapter 4, we discussed the corrective and totally corrective algorithms used in this thesis. In this section, we provide a broader overview of the subject. Recall that corrective algorithms, the updated distribution $\mathbf{d}^t$ is based on the last hypothesis, while totally corrective algorithms base their updates on all previous hypotheses.

We begin with the origins of the corrective update. Schapire and Singer [74] were able to bound the classification error of the training sample in terms of the product of the normalization factors Z_t found at each iteration of the AdaBoost algorithm. Recall that at iteration t, AdaBoost chooses weight w_t based only on hypothesis h^t, and w_t is not changed in subsequent iterations. Schapire and Singer also showed that when $u_n^t \in \pm 1$, AdaBoost chooses w_t to minimize Z_t. In the general case where $u_n^t \in [-1, 1]$, then the optimal choice of w_t minimizes Z_t via line search. Schapire and Singer showed that in this case, Z_t is minimized when $\mathbf{d}^t \cdot \mathbf{u}^t = 0$. In other words, the updated distribution $\mathbf{d}^t$ is chosen so that its edge with respect to the most recent hypothesis is zero. This choice of w_t makes the bound on the training error as tight as possible, and the resulting update is called a *corrective update*.

Kivinen and Warmuth [43] showed that AdaBoost with the corrective update

is equivalent to solving the following entropy minimization problem for $\mathbf{d}^t$:

$$\mathbf{d}^t = \underset{d \in \mathcal{S}^N, \mathbf{d} \cdot \mathbf{u}^t = 0}{\operatorname{argmin}} \Delta(\mathbf{d}, \mathbf{d}^{t-1}).$$

Furthermore, they showed that that the value of w_t that minimizes Schapire and Singer's bound is the Lagrange multiplier corresponding to the $\mathbf{d} \cdot \mathbf{u}^t = 0$ constraint. This was an early connection between boosting and optimization theory. Other early connections between boosting and optimization theory include [62, 33, 36, 45].

Since the corrective update depends only on the last hypothesis, corrective algorithms can cycle, picking similar hypotheses multiple times [91]. This motivates the *totally corrective* update. At iteration t, totally corrective algorithms constrain the edges $\mathbf{d}^t \cdot \mathbf{u}^q$, for $q = 1 \ldots t$. The weight w_q on hypothesis q is the dual variable to the $\mathbf{d}^t \cdot \mathbf{u}^q$ constraint, so the entire weight vector $\mathbf{w}$ changes at each iteration, instead of just w_t. This also means that the updated $\mathbf{d}^t$ depends on *all* previous hypotheses, not just the last one one.

LogLossBoost [18], which changes all weights $\mathbf{w}$ on the hypotheses at each iteration, is a precursor to the totally corrective algorithm The update is called a *parallel update*, but it is not equivalent to a totally corrective update because it is not based on requiring all previous hypotheses to have small edge.

The first totally corrective algorithm was TotalBoost[91]. At each iteration, TotalBoost finds $\mathbf{d}$ by solving the optimization problem

$$\begin{aligned} \mathbf{d}^t = \quad & \underset{d \in \mathcal{S}^N}{\operatorname{argmin}} \quad \Delta(\mathbf{d}, \mathbf{d}^{t-1}) \\ & \text{s.t.} \qquad \mathbf{d} \cdot \mathbf{u}^q \leq \widehat{\gamma}_t - \epsilon \text{ for } q = 1 \ldots t \end{aligned}$$

where $\widehat{\gamma}_t$ is a specially chosen parameter and ϵ is the precision parameter. LPBoost [36, 21], SoftBoost [87], ERLPBoost [90], and Binary ERLPBoost are also totally corrective algorithms.

A major criticism of totally corrective algorithms is that they must solve complex optimization problems at each iteration. Shalev Schwartz and Singer [77] propose a corrective version of ERLPBoost, and this algorithm can easily be extended to the binary entropy as well. These algorithms are discussed at length in Chapters 4 and 5.

6.5 Significant Experimental Results for Boosting

Many of the theoretical developments in the study of boosting were inspired by experimental results. In an early experimental study of boosting, Quinlan [61] observed that even after the master hypothesis classifies all training examples correctly, the generalization error of AdaBoost continued to decrease with additional iterations. This was attributed to the fact that the minimum margin of the examples for the master hypothesis continued to increase even after the training error stabilizes [71]. In the same study, Quinlan observed that AdaBoost sometimes generalized very poorly, and in those cases, he observed that the distribution on the examples was very concentrated. Similar results were found in [51] and [4]. In addition, Bauer and Kohavi [4] observed that AdaBoost reduces both bias and variance.

Dietterich [22] made the famous connection between poor generalization and noisy data. In particular, when noise was deliberately introduced, AdaBoost puts a great

deal of weight on a few noisy examples, and the performance of AdaBoost deteriorated markedly. This is the motivation for capping the weights on the examples, which was discussed in Chapter 6.3.

Mease and Wyner [56] attempt to refute the statistical view of boosting first set out in [34] with a series of simple experiments on data where the Bayes error is controlled. The Bayes error of a dataset is the lowest error that any algorithm can achieve. One of their most interesting findings is that when the Bayes error is greater than zero, LogitBoost often overfits while AdaBoost does not. This is surprising because LogitBoost optimizes a loss function that is more *gentle* than the AdaBoost loss function, and this fact is used to argue that LogitBoost is robust to noise.

A rigorous comparison of a very large number of disparate supervised learning algorithms was performed by Caruana and Niculesco-Mizil [14]. The only boosting algorithm they used was AdaBoost, though it was tried with a variety of decision trees. In these experiments, AdaBoost with decision trees had the best overall performance. Although experiments were performed on a large number of datasets, the experiments were limited to 5000 examples, so the scalability of the algorithms was not explored.

Finally, Arvey [2] performed a recent experimental comparison of margin-conscious BrownBoost, BrownBoost, AdaBoost, LogitBoost, and ERLPBoost. In this comparison, ERLPBoost outperformed AdaBoost, LogitBoost and BrownBoost and is competitive with margin conscious BrownBoost on noisy data. Furthermore, in these experiments, the performance of ERLPBoost might be underestimated because of a poor choice of η.

Chapter 7

Conclusion

The boosting community commonly uses margin maximization as a proxy for good generalization. The first contribution of this thesis is a strong argument that direct margin maximization does not necessarily result in low generalization error. LPBoost is the boosting algorithm that maximizes the margin directly. Although LPBoost is often used in practice, the analysis of this algorithm has proved to be elusive. In particular, there are no known iteration bounds, and we show in this thesis why this is the case. This result, originally shown in [87], is a lower bound on the number of iterations required by LPBoost that is linear in the number of examples. This is significant because a good iteration bound in the boosting context is one that is logarithmic in the number of examples. In this thesis we also show that any linearly separable dataset can be reduced to a dataset on which LPBoost misclassifies all examples by adding a bad example and a bad hypothesis. Both of the results we have discussed rely on forcing LPBoost to concentrate its weight on a single example and a single hypothesis. Boosting algorithms

use the weights on the examples to choose subsequent hypotheses. If the weight becomes concentrated on a few examples, this can cause the boosting algorithm to choose bad hypotheses. This is particularly true when some of the examples are noisy, which means that the data is not linearly separable.

In this thesis we present three algorithms: SoftBoost, ERLPBoost, and Binary ERLPBoost, each of which employs either the relative entropy or the binary relative entropy to distribute the weights on the examples more uniformly. Thus, we avoid the problem caused by too much weight concentrating on a single example as can happen with LPBoost. SoftBoost minimizes the entropy subject to an increasing number of constraints, while the entropies are used as regularizers in ERLPBoost and Binary ERLPBoost. Although no iteration bound is known for LPBoost, incorporating an entropy into the algorithm makes it possible to prove an iteration bound that is $O\left(\frac{\ln(N/\nu)}{\epsilon^2}\right)$. Furthermore, all three algorithms come ϵ-close to the optimal linear programming solution.

Shalev-Schwartz and Singer [77] proposed a similar algorithm to ERLPBoost except that instead of updating the weights on *all* of the hypotheses at every iteration, it updates only the weight on the *last* hypothesis. We call this algorithm Corrective ERLPBoost. The algorithm defined in [77] uses a relative entropy regularizer, but it is easy to derive a similar algorithm that uses a binary entropy regularizer. We call this algorithm Corrective Binary ERLPBoost. We first describe the relationship between these corrective algorithms and their totally corrective counterparts. Next, we note that the algorithm described in [77] does not have a stopping criterion. We next introduce a

stopping criterion for this algorithm to enable a fair comparison between corrective and totally corrective algorithms in terms of training time and generalization error.

This thesis presents the first experimental analysis of optimization-based boosting algorithms on large-scale data. Our main experimental result demonstrates that LPBoost is extremely unstable on real world data. For LPBoost, generalization error fluctuated wildly from one iteration to the next. Capping is a common approach to making boosting algorithms robust to noise by limiting the weight on a single example, but it did not ameliorate the instability. However, even a small amount of entropy regularization resolved the instability and caused generalization error to steadily decrease over time.

The second important experimental result compares corrective and totally corrective algorithms. Relative performance depends heavily on the hypothesis class. SVM hypotheses generalize better overall than the other hypothesis classes. For SVM hypotheses, the totally corrective algorithms are faster and have lower generalization error. An iteration of the corrective algorithms is dominated by the time it takes the oracle to return a hypothesis. In contrast, the totally corrective algorithms are dominated by the optimization problem that defines the update on the weights. The slowness of the SVM oracle explains why totally corrective algorithms are faster than the corrective algorithms. Ultimately, the more time-consuming it is to find the next hypothesis, the more it pays to invest in a slower but more effective update. The totally corrective optimization problems have not been studied extensively before, and improving them would make the totally corrective algorithms even more competitive.

Our motivation for including SVM hypotheses is to enable a direct comparison between SVMs and boosting using the same hypothesis class. Unfortunately, none of our boosting algorithms compare favorably with SVMs in either training time or generalization error. Part of the disparity probably comes from comparing a mature technology with a nascent one. Also, boosting algorithms have to solve an optimization problem at every iteration while the SVM optimization problem is only solved once. Thus, even if the two optimization problems could be solved in the same amount of time, ERLPBoost would be slower overall unless the optimal solution includes very few hypotheses.

Ultimately, we demonstrated theoretically and experimentally that ERLPBoost and Binary ERLPBoost are more robust than direct margin maximization. Furthermore, we demonstrated that totally corrective algorithms clearly outperform corrective algorithms when invoking the oracle is computationally expensive. We hope this work will inspire future theoretical exploration so that the full potential of this algorithmic paradigm may be realized. It is also our hope that these algorithms will be a practical and effective tool to solve classification problems on real-world data.

Appendix A

The Active Set Method

In this section, we discuss how the primal and dual variables can be related via the Karush-Kuhn-Tucker (KKT) conditions. This is discussed in great length in both [58] and [7], and the following discussion is a summary of the relevant points.

Relating the primal and dual variables is relatively straightforward in the case where there are only equality constraints. When the optimization problem also has inequality constraints, the KKT conditions are considerably more difficult. The active set method[58] can be employed to reduce a problem with inequality constraints to one that only has equality constraints.

First we derive the KKT system for an equality constrained quadratic program. Suppose we are trying to solve an optimization problem of the form

$$\begin{aligned} \min_{\mathbf{x}} \ & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{x}^T\mathbf{g} \\ \text{s.t.} \ & A\mathbf{x} = \mathbf{b} \end{aligned}$$

where we there are m constraints and A is an $m \times n$ matrix. Let us assume that $m \leq n$. Also, let us denote the Lagrange multipliers for the equality constraints by $\boldsymbol{\lambda}$. Then the Lagrangian for this optimization problem is

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{x}^T\mathbf{g} + \boldsymbol{\lambda}^T(A\mathbf{x} - b).$$

Let us denote the optimal values of $\mathbf{x}$ and $\boldsymbol{\lambda}$ by $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$ respectively. Then Karush-Kuhn-Tucker (KKT) conditions for this problem are

$$A\mathbf{x}^* = b \text{ and } H\mathbf{x} + g + A^T\boldsymbol{\lambda}^* = 0.$$

These can be expressed equivalently in matrix form as

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -g \\ b \end{bmatrix}. \tag{A.1}$$

If H is positive definite, A is of full row rank, and the feasible set is nonempty, then there is a unique solution for $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$. Moreover, if we already know $\mathbf{x}^*$ and we want to solve for $\boldsymbol{\lambda}^*$, the problem reduces to

$$A^T\boldsymbol{\lambda}^* = -g - Hx^*. \tag{A.2}$$

We have assume that $m \leq n$, so this is potentially an overconstrained problem, but we can solve this system using least squares because we know that there is a consistent solution for $\boldsymbol{\lambda}^*$.

Suppose instead that the optimization has inequality constraints:

$$\begin{aligned} \min_{\mathbf{x}} \ & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{x}^T\mathbf{g} \\ \text{s.t. } & \mathbf{a}_i^T\mathbf{x} \leq b_i \text{ for } \ i = 0 \ldots m, \end{aligned} \tag{A.3}$$

where the $\mathbf{a}_i$ are the rows of the A matrix and $\boldsymbol{\lambda}$ are the dual variables for the constraints. As before, $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$ are the optimal solutions for $\mathbf{x}$ and $\boldsymbol{\lambda}$ respectively. The KKT conditions for this problem are

$$A\mathbf{x}^* = b, \qquad H\mathbf{x} + g + A^T\boldsymbol{\lambda}^* = 0,$$

$$\lambda_i(\mathbf{a}_i^T\mathbf{x} - b) = 0, \quad \boldsymbol{\lambda}_i \geq 0 \text{ for } i = 1 \dots m$$

The key to the active set method is the complementary slackness condition:

$$\lambda_i(\mathbf{a}_i^T\mathbf{x} - b) = 0.$$

From this we can infer that if $\mathbf{a}_i^T\mathbf{x} < b$ (the inequality constraint is inactive), then $\lambda_i = 0$. Therefore the only nonzero dual variables correspond to the constraints where $\mathbf{a}_i^T\mathbf{x} = b$, also known as the active constraints. Let us define the active set $\mathcal{A}(\mathbf{x}) = \{i : \mathbf{a}_i^T\mathbf{x} = b\}$. If the active set is already known, then the optimal solution to (A.3) is equivalent to the solution of

$$\begin{aligned} \min_{\mathbf{x}} \ & \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{x}^T\mathbf{g} \\ \text{s.t.} \ & \mathbf{a}_i^T\mathbf{x} = b_i \text{ for } i = \mathcal{A}(\mathbf{x}). \end{aligned}$$

This can be solved by a linear system similar to (A.1), except that in this case only the active constraints are included. The active set is rarely known ahead of time. However, there is one particular scenario that will come up frequently in this dissertation where this is true: the case where $\mathbf{x}^*$ is known and we want to solve for $\boldsymbol{\lambda}^*$. In this case, the problem reduces to (A.2).

It should be noted that the KKT conditions can be reduced to solving a matrix equation because the problem we are considering has a quadratic objective function and

affine constraints. However, the algorithms proposed in this dissertation will not have quadratic objective functions. This can be resolved in two different ways. First, we generally solve these optimization problems via sequential quadratic programming [58], so it makes sense to simply take the quadratic approximation of the objective function.

In this thesis, we make use of active sets when we know the value of the primal variables and seek to use this information to find the value of the dual variables. In this case, reducing the problem to a linear system is also straightforward. It is important to consider is that numerical issues may make this approach unreliable. In practice, optimizers find the value of the primal variables to a finite precision. Often this is not very high precision, which results in a certain amount of ambiguity when it comes to determining the optimal active set.

Bibliography

[1] N. Abe, J. Takeuchi, and M. K. Warmuth. Polynomial learnability of stochastic rules with respect to the KL-divergence and quadratic distance. *IEICE Transactions on Information and Systems*, E84-D(3):299–316, 2001.

[2] Aaron Arvey. Experiments with a margin conscious brownboost. submitted to JMLR, 2009.

[3] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[4] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, pages 1–33, 1998.

[5] Steve Benson, Lois Curfman McInnes, Jorge Moré, Todd Munson, and Jason Sarich. TAO user manual (revision 1.9). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2007. http://www.mcs.anl.gov/tao.

[6] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training al-

gorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[8] Joseph K. Bradley and Robert E. Schapire. Filterboost: Regression and classification on large datasets. In *Advances in Neural Information Processing Systems*, 2008.

[9] L.M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

[10] L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1518, 1999. Also Technical Report 504, Statistics Department, University of California Berkeley.

[11] Leo Breiman. Arcing classifiers. *Annals of Statistics*, 26(2):841–849, 1998.

[12] T. Bylander. The binary exponentiated gradient algorithm for learning linear functions. In *Proc. 10th Annu. Workshop on Comput. Learning Theory*. ACM Press, New York, NY, 1997.

[13] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, 1995.

[14] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, 2006.

[15] Y. Censor and S. A. Zenios. *Parallel Optimization*. Oxford, New York, 1997.

[16] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] Nello Christianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[18] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285, 2002.

[19] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[20] C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[21] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Mach. Learn.*, 46(1-3):225–254, 2002.

[22] T. G. Dietterich. An experimental comparison of three methods for construct-

ing ensembles of decision trees: Bagging, Boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[23] C. Domingo and O. Watanabe. Madaboost: A modification of Adaboost. In *Proc. COLT '00*, pages 180–189, 2000.

[24] Harry Drucker and Corinna Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems*, pages 479–485, 1995.

[25] Nigel Duffy and David Helmbold. Potential boosters? In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *In Advances in Neural Information Processing Systems 12*, pages 258–264. MIT Press, 2000.

[26] Nigel Duffy and David P. Helmbold. A geometric approach to leveraging weak learners. In Paul Fischer and Hans Ulrich Simon, editors, *Computational Learning Theory: 4th European Conference (EuroCOLT '99)*, pages 18–33, Berlin, March 1999. Springer.

[27] Damian Eads, Edward Rosten, and David Helmbold. Learning object location predictors with boosting and grammar-guided feature extraction. In *British Machine Vision Conference*, 2009.

[28] Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *International Conference on Machine Learning*, pages 97–105, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[29] Y. Freund. An adaptive version of the boost by majority algorithm. *Mach. Learn.*, 43(3):293–318, 2001.

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

[31] Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, 1995.

[32] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[33] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *COLT '96: Proceedings of the ninth annual conference on Computational learning theory*, pages 325–332, New York, NY, USA, 1996. ACM.

[34] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2), 2000.

[35] Dmitry Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *J. Mach. Learn. Res.*, 4:101–117, 2003.

[36] Adam J. Grove and Dale Schuurmans. Boosting in the limit: maximizing the margin of learned ensembles. In *AAAI '98/IAAI '98*, pages 692–699, Menlo Park, CA, USA, 1998.

[37] D. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning*, 27(1):97–119, 1997.

[38] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[39] R. Hettich and K.O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, September 1993.

[40] Thorsten Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, New York, NY, USA, 2006. ACM.

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

[42] S. Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, (6):341–361, 2005.

[43] J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proc. 12th Annu. Conference on Comput. Learning Theory*, pages 134–144. ACM Press, New York, NY, 1999.

[44] Vladimir Koltchinskii, Dmitriy Panchenko, and Fernando Lozano. Further explanation of the effectiveness of voting methods: The game between margins and weights.

In *COLT '01/EuroCOLT '01: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 241–255, London, UK, 2001. Springer-Verlag.

[45] J. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 125–133. ACM, 1999.

[46] I. Laptev. Improvements of object detection using boosted histograms. *Biologically Motivated Computer Vision*, 3:949–958, 2006.

[47] P. Long and R. Servedio. Adaptive martingale boosting. In *22st Annual Conference in Neural Information Processing Systems (NIPS)*, pages 977–984, 2008.

[48] Philip M. Long and Rocco A. Servedio. Martingale boosting. In *COLT*, pages 79–94, 2005.

[49] Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 608–615, New York, NY, USA, 2008. ACM.

[50] Robin Lougee-Heimer. The common optimization interface for operations research. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[51] Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. In *In Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551. AAAI Press, 1997.

[52] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Asymmetric boosting. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 609–619, New York, NY, USA, 2007. ACM.

[53] Llew Mason, Peter Bartlett, and Jonathan Baxter. Direct optimization of margins improves generalization in combined classifiers. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 288–294, Cambridge, MA, USA, 1999. MIT Press.

[54] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 512–518, 1999.

[55] Andrew McCallum. Andrew mccallum's code and data, 2009.

[56] D. Mease and A. Wyner. Evidence contrary to the statistical biew of boosting. *Journal of Machine Learning Research*, 9:131–156, 2008.

[57] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. pages 118–183, 2003.

[58] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer, 1999.

[59] A. Opelt, A. Pinz, and A. Zisserman. A Boundary-Fragment-Model for Object Detection. *Lecture Notes in Computer Science*, 3952:575, 2006.

[60] M.S. Pinsker. *Information and Information Stability of Random Variables and Processes*. Holden-Day, 1964.

[61] J. R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[62] G. Rätsch. *Robust Boosting via Convex Optimization: Theory and Applications*. PhD thesis, University of Potsdam, 2001.

[63] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Mach. Learn.*, 42(3):287–320, 2001.

[64] G. Rätsch, B. Schölkopf, A.J. Smola, S. Mika, T. Onoda, and K.-R. Müller. Robust ensemble learning. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 207–219. MIT Press, Cambridge, MA, 2000.

[65] G. Rätsch and M. K. Warmuth. Maximizing the margin with boosting. In *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 334–350. Springer, July 2002.

[66] G. Rätsch and M.K. Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

[67] Lev Reyzin and Robert E. Schapire. How boosting the margin can also boost classifier complexity. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 753–760, New York, NY, USA, 2006. ACM Press.

[68] C. Rudin, I. Daubechies, and R.E. Schapire. The dynamics of adaboost: Cyclic be-

havior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

[69] C. Rudin, R.E. Schapire, and I. Daubechies. Boosting based on a smooth margin. In *17th Annual Conference on Computational Learning Theory*, 2004.

[70] C. Rudin, R.E. Schapire, and I. Daubechies. Analysis of boosting algorithms using the smooth margin function. *The Annals of Statistics*, 6(35):2723–2768, 2007.

[71] R.E. Schapire, Y. Freund, P.L. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[72] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, 1990.

[73] Robert E. Schapire. A brief introduction to boosting. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[74] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proc. 11th Annu. Conf. on Comput. learning Theory*, 1998.

[75] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based systemfor text categorization. *Mach. Learn.*, 39(2-3):135–168, 2000.

[76] Rocco A. Servedio. Smooth boosting and learning with malicious noise. In *14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 2001, Proceedings*, volume 2111 of *Lecture Notes in Artificial Intelligence*, pages 473–489. Springer, 2001.

[77] Shai Shalev-Shwartz and Yoram Singer. On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. In *Proceedings of COLT 2008*, 2008.

[78] A. Smola, S. V. N. Vishwanathan, and Q. Le. Bundle methods for machine learning. In Daphne Koller and Yoram Singer, editors, *Advances in Neural Information Processing Systems 20*, Cambridge MA, 2007. MIT Press.

[79] Kinh Tieu and Paul Viola. Boosting image retrieval. *Int. J. Comput. Vision*, 56(1-2):17–36, 2004.

[80] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:762–769, 2004.

[81] L. G. Valiant. A theory of the learnable. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445, New York, NY, USA, 1984. ACM.

[82] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[83] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[84] S.V.N. Vishwanathan. personal communication, 2009.

[85] S.V.N. Vishwanathan. personal communication, 2009.

[86] J. von Neumann. Zur theorie der gesellschaftsspiele. *Math. Annalen*, 100:295–320, 1928.

[87] M. K. Warmuth, K. Glocer, and G. Rätsch. Boosting algorithms for maximizing the soft margin. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1585–1592. MIT Press, Cambridge, MA, 2008.

[88] Manfred Warmuth. personal communication, 2009.

[89] Manfred Warmuth and S.V.N. Vishwanathan. Survey of boosting from an optimization perspective, 2009.

[90] Manfred K. Warmuth, Karen A. Glocer, and S. V. N. Vishwanathan. Entropy regularized lpboost. In *ALT*, pages 256–271, 2008.

[91] M.K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *Proc. ICML '06*, pages 1001–1008. ACM Press, 2006.