Empirical Bernstein Boosting

Pannagadatta K. Shivaswamy
Department of Computer Science
Columbia University, New York NY 10027
pks2103@cs.columbia.edu

Tony Jebara
Department of Computer Science
Columbia University, New York NY 10027
jebara@cs.columbia.edu

Abstract

Concentration inequalities that incorporate variance information (such as Bernstein's or Bennett's inequality) are often significantly tighter than counterparts (such as Hoeffding's inequality) that disregard variance. Nevertheless, many state of the art machine learning algorithms for classification problems like AdaBoost and support vector machines (SVMs) extensively use Hoeffding's inequalities to justify empirical risk minimization and its variants. This article proposes a novel boosting algorithm based on a recently introduced principle—sample variance penalization—which is motivated from an empirical version of Bernstein's inequality. This framework leads to an efficient algorithm that is as easy to implement as AdaBoost while producing a strict generalization. Experiments on a large number of datasets show significant performance gains over AdaBoost. This paper shows that sample variance penalization could be a viable alternative to empirical risk minimization.

1 INTRODUCTION

In classification problems, many machine learning algorithms minimize a convex upper bound on the misclassification rate. For example, AdaBoost (Freund & Schapire, 1997) minimizes the exponential loss and support vector machines (Vapnik, 1995) minimize the hinge loss. The convexity of such losses is helpful for computational as well as generalization reasons (Bartlett et al., 2006). In most problems, the aim is not to obtain a function that performs well on training data but rather to estimate a function (using training data) that performs well on future unseen test data. This is accomplished by minimizing empirical risk on the training set while choosing a function of small complexity. The rationale behind this approach is that the empirical risk converges (uniformly) to the true unknown risk. Various concentration inequalities show how fast the empirical risk converges to the true risk.

A key tool in obtaining such bounds is Hoeffding's inequality which relates the empirical mean of a bounded random variable to its true mean. Bernstein's and Bennett's inequalities relate the true mean of a random variable to the empirical mean but also incorporate the *true* variance of the random variable. If the true variance of a random variable is small, these bounds can be significantly tighter than Hoeffding's bound. Recently, there have been empirical counterparts of Bernstein's inequality (Audibert et al., 2007; Maurer & Pontil, 2009); these bounds incorporate the empirical variance of a random variable rather than its true variance. The advantage of these bounds is that the quantities they involve are actually observable. Previously, these bounds have been applied in sampling procedures (Mnih et al., 2008). In this paper, we motivate a new boosting algorithm that not only minimizes the empirical misclassification rate but also the empirical variance on the exponential loss. To do so, we make an appeal to the "sample variance penalization" principle and the empirical Bernstein inequality.

In recent years, there has been surge of interest in designing classification algorithms that incorporate higher order information beyond empirical risk to improve generalization. For instance, the classical perceptron algorithm was extended to include variance information in the second order perceptron approach (Cesa-Bianchi et al., 2002). Probabilistic versions of online algorithms (Crammer et al., 2009a) incorporating variance information have been successfully applied to text classification problems. A batch al-

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

gorithm, called the relative margin machine (Shivaswamy & Jebara, 2010), which trades off between the margin and the spread of the data has also been developed. Similarly, Gaussian margin machines (Crammer et al., 2009b) have been proposed and are motivated by minimizing a PAC Bayesian bound with a Gaussian prior on the hyperplane.

2 HOEFFDING AND EMPIRICAL BERNSTEIN BOUNDS

This section reviews some classical as well new concentration inequalities. The purpose of this section is to present sample variance penalization (Maurer & Pontil, 2009) as a motivation for the rest of the paper. However, for the sake of clarity and completeness, we review a few other classical results as well. First, recall Hoeffding's inequality.

Theorem 2.1 *Let $Z_1, Z_2, \ldots, Z_n$ be* iid *random variables with values in $[0,1]$. Then, for any $\delta > 0$, with probability at least $1-\delta$ (over the draw of $Z_1, Z_2, \ldots, Z_n$), the following inequality holds:*

$$\mathbf{E}[Z] \leq \frac{1}{n}\sum_{i=1}^{n} Z_i + \sqrt{\frac{\ln(1/\delta)}{2n}}.$$

In machine learning settings, samples $(X_i, y_i)_{i=1}^n$ are drawn *iid* from a distribution $\mathcal{D}$ where $(X_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. From this data, a learning algorithm outputs a function $f : \mathcal{X} \to \mathbb{R}$ (typically from a fixed set of functions) that should predict well on future samples. In other words, the function should minimize a fixed loss $l : \mathbb{R} \times \mathcal{Y} \to [0,1]$ on unseen test examples drawn from the same distribution.

Empirical risk minimization (ERM) is a popular approach to minimizing loss on unseen test examples. Suppose the learning algorithm is to choose a function from a finite set of candidate functions $\mathcal{F}$. Consider the extension of Hoeffding's inequality such that it holds uniformly over all functions in $\mathcal{F}$.

Corollary 2.2 *Let $(X_i, y_i)_{i=1}^n$ be drawn* iid *from a distribution $\mathcal{D}$. Let $\mathcal{F}$ be a finite class of functions $f : \mathcal{X} \to \mathbb{R}$. Then, given a loss function $l : \mathbb{R} \times \mathcal{Y} \to [0,1]$, for any $\delta > 0$, with probability at least $1-\delta$, $\forall f \in \mathcal{F}$,*

$$\mathbf{E}[l(f(X), y)] \leq \frac{1}{n}\sum_{i=1}^{n} l(f(X_i), y_i) + \sqrt{\frac{\ln(|\mathcal{F}|)/\delta}{2n}} \quad (1)$$

where $|\mathcal{F}|$ is the cardinality of $\mathcal{F}$.

In fact, the above corollary can be extended to infinite function classes $\mathcal{F}$ by replacing the term $|\mathcal{F}|$ with a suitable complexity measure.

The empirical risk minimization principle selects a function from a class to minimize empirical loss on the training data, *i.e.,*

$$\arg\min_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} l(f(X_i), y_i).$$

Since (1) holds uniformly over all the functions in the class $\mathcal{F}$, minimizing the empirical risk minimizes an upper bound on the future loss.

As an alternative to Hoeffding's bound, consider Bernstein's inequality.

Theorem 2.3 *Under the same conditions as Theorem 2.1, for any $\delta > 0$, with probability at least $1-\delta$,*

$$\mathbf{E}[Z] \leq \frac{1}{n}\sum_{i=1}^{n} Z_i + \sqrt{\frac{2\mathbf{V}[Z]\ln(1/\delta)}{n}} + \frac{\ln(1/\delta)}{3n},$$

where $\mathbf{V}[Z] = \mathbf{E}\left(Z - \mathbf{E}[Z]\right)^2$.

When the variance $\mathbf{V}[Z]$ of the random variable Z is small, the above bound can be significantly tighter than Hoeffding's bound. To get an idea of why the bound in Theorem 2.3 can be better than the one in Theorem 2.1, consider a situation in which $\mathbf{V}[Z] = 0$. In this scenario, $\frac{1}{n}\sum_{i=1}^{n} Z_i$ converges to $\mathbf{E}[Z]$ at the rate $\mathcal{O}(1/n)$ according to Bernstein's inequality. However, in the case of Hoeffding's inequality, the convergence is at a much slower $\mathcal{O}(1/\sqrt{n})$ rate. However, one limitation of the above bound is that the true value $\mathbf{V}[Z]$ is often an unknown quantity and only an empirical estimate is available. To address this limitation, recall the following result from (Maurer & Pontil, 2009) which is similar to Theorem 2.3 but holds for an empirical estimate of the variance as opposed to the true value $\mathbf{V}[Z]$.

Theorem 2.4 *Under the same conditions as Theorem 2.1, for any $\delta > 0$, with probability at least $1-\delta$,*

$$\mathbf{E}[Z] \leq \frac{1}{n}\sum_{i=1}^{n} Z_i + \sqrt{\frac{2\hat{\mathbf{V}}[Z]\ln(2/\delta)}{n}} + \frac{7\ln(2/\delta)}{3(n-1)},$$

where $\hat{\mathbf{V}}[Z]$ is the empirical variance given by:[1]

$$\hat{\mathbf{V}}[Z] = \frac{1}{n(n-1)}\sum_{n \geq i > j \geq 1} (Z_i - Z_j)^2.$$

The above theorem has the advantage that all the quantities involved are empirical quantities that can be obtained from data. We finally state the following uniform convergence result that can be used to motivate sample variance penalization.

[1] For brevity, unless we specify extra constraints on i and j, $i > j$ in future summations must be read as $n \geq i > j \geq 1$.

Theorem 2.5 *Let $(X_i, y_i)_{i=1}^n$ be drawn* iid *from a distribution $\mathcal{D}$. Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$. Then, given a loss function $l : \mathbb{R} \times \mathcal{Y} \to [0, 1]$, for any $\delta > 0$, with probability at least $1 - \delta$, $\forall f \in \mathcal{F}$,*

$$\mathbf{E}[l(f(X), y)] \leq \frac{1}{n}\sum_{i=1}^{n} l(f(X_i), y_i) + \frac{15\ln(\mathcal{M}(n)/\delta)}{(n-1)} + \sqrt{\frac{18\hat{\mathbf{V}}[l(f(X), y)]\ln(\mathcal{M}(n)/\delta)}{n}}, \quad (2)$$

where $\mathcal{M}(n)$ is a complexity measure (Maurer & Pontil, 2009). Unlike the empirical risk in (2), the empirical variance $\hat{\mathbf{V}}[l(f(X), y)]$ has a multiplicative factor involving $\mathcal{M}(n)$ (which is typically difficult to estimate), δ (the required confidence at which we want the bound to hold) and n. Thus, for a given problem, it is difficult to specify the trade-off between empirical risk and empirical variance a priori. Hence, the algorithm we propose in this paper will involve a scalar parameter which trades off between these two criteria. As is often the case in practice, this trade-off parameter has to be tuned using a validation set. Minimizing this uniform convergence bound leads to the so-called sample variance penalization principle:

$$\arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} l(f(X_i), y_i) + \lambda\sqrt{\frac{\hat{\mathbf{V}}[l(f(X), y)]}{n}},$$

where $\lambda \geq 0$ explores the trade-off between the empirical risk and the empirical variance. The aim of the rest of this paper is to derive an efficient, AdaBoost style algorithm for sample variance penalization.

3 LOSS FUNCTIONS

In this section, we first explore the possibility of applying the results from the previous section on the $0-1$ loss function. We show that sample variance penalization with $0-1$ loss merely corresponds to empirical risk minimization. Subsequently, we apply the sample variance penalization principle to the exponential loss where it produces a qualitatively different criterion.

In the previous section, we dealt with a generic loss function without making any assumptions on the problem type. However, from now on, we restrict ourselves to binary classification problems. In classification problems, we have, $\mathcal{Y} = \{\pm 1\}$. Define the classification loss function $l_1(f(X), y)$ as:

$$l_1(f(X), y) := \mathbf{I}_{yf(X)\leq 0} = \begin{cases} 0 & \text{if } yf(X) > 0, \\ 1 & \text{if } yf(X) \leq 0. \end{cases} \quad (3)$$

Given an *iid* set of examples $(X_i, y_i)_{i=1}^n$, our aim is to minimize the future probability of error, *i.e.,* to minimize $\mathbf{P}_{\mathcal{D}}[yf(X) \leq 0]$.

Lemma 3.1 *Let $(X_i, y_i)_{i=1}^n$ be drawn* iid *from a distribution $\mathcal{D}$. Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathcal{Y}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, $\forall f \in \mathcal{F}$,*

$$\mathbf{P}_{\mathcal{D}}[yf(X) \leq 0] \leq \hat{p} + \sqrt{\frac{18\hat{p}(1-\hat{p})\ln(\mathcal{M}(n)/\delta)}{n-1}} + \frac{15\ln(\mathcal{M}(n)/\delta)}{(n-1)}, \quad (4)$$

where $\hat{p} = \hat{\mathbf{P}}[l_1(f(X), y)] = \frac{1}{n}\sum_{i=1}^{n} \mathbf{I}_{y_i f(X_i)\leq 0}$ is the empirical error.

Proof The proof is straightforward; it is a direct application of Theorem (2.5) on the 0-1 loss (3). First observe that,

$$\mathbf{E}_{\mathcal{D}}[\mathbf{I}_{yf(X)\leq 0}] = \mathbf{P}_{\mathcal{D}}[yf(X) \leq 0].$$

Moreover, denoting $\mathbf{I}_{y_i f(X_i)\leq 0}$ by s_i for brevity,

$$\begin{aligned}
\hat{\mathbf{V}}[l_1(f(X), y)] &= \frac{1}{n(n-1)}\sum_{i>j}(s_i - s_j)^2 \\
&= \frac{1}{n(n-1)}\left((n-1)\sum_{i=1}^{n} s_i^2 - 2\sum_{i>j} s_i s_j\right) \\
&= \frac{n}{n-1}\left(\frac{1}{n}\sum_{i=1}^{n} s_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n} s_i\right)^2\right) \\
&= \frac{n}{n-1}\hat{p}(1-\hat{p}).
\end{aligned}$$

Going from line three to four we used the fact that, for an indicator random variable, $s_i^2 = s_i$. ∎

Thus, to minimize future classification error, sample variance penalization suggests minimizing $\hat{p} + \lambda\sqrt{\hat{p}(1-\hat{p})}$. Consider $\hat{p} \in [0, 1/2)$ which is clearly the regime of interest in classification problems. It is easy to see that, for $\hat{p} \in [0, 1/2)$, the quantity $\hat{p} + \lambda\sqrt{\hat{p}(1-\hat{p})}$ is a monotonically increasing function of the empirical error $\hat{p}$. Therefore, sample variance penalization is equivalent to minimizing the empirical error $\hat{p}$. Thus, for any finite non-negative value of λ, sample variance penalization merely reduces to empirical risk minimization with the $0-1$ loss.

3.1 MINIMIZING A CONVEX UPPER BOUND ON THE $0-1$ LOSS

It is important to note that empirical risk minimization with the $0-1$ loss is a hard problem; this problem is often circumvented by minimizing convex upper bounds on this empirical risk. In this paper, we consider the following exponential loss:

$$l_2(f(X), y) := e^{-yf(X)}. \quad (5)$$

Algorithm 1 AdaBoost

Require: $(X_i, y_i)_{i=1}^n$, weak learners $\mathcal{H}$
  Initialize the weights: $w_i \leftarrow \frac{1}{n}$; initialize f to predict zero on all inputs.
  for $s \leftarrow 1$ to S **do**
    Get a weak learner $G^s(\cdot)$ that minimizes $\sum_{i:y_i \neq G^s(X_i)} w_i$
    $\alpha_s = \frac{1}{2} \log \left(\frac{\sum_{y_i = G^s(X_i)} w_i}{\sum_{y_i \neq G^s(X_i)} w_i} \right)$
    if $\alpha_s < 0$ **then** break **end if**
    $f(\cdot) \leftarrow f(\cdot) + \alpha_s G^s(\cdot)$
    $w_i \leftarrow w_i \exp(-y_i G^s(X_i) \alpha_s)/Z_s$ where Z_s is such that $\sum_{i=1}^n w_i = 1$.
  end for

Here, we assume $f : \mathcal{X} \to [-1, 1]$. Our aim is to still minimize the future $0-1$ loss yet only through the surrogate exponential loss function. A computational advantage of this convex loss function is that it is easy to minimize. First, we relate the future probability of error to the exponential loss:

$$\mathbf{P}_{\mathcal{D}}[yf(X) \leq 0] = \mathbf{E}_{\mathcal{D}}[\mathbf{I}_{yf(X) \leq 0}] \leq \mathbf{E}_{\mathcal{D}}[e^{-yf(X)}].$$

It is now possible to relate the future probability of error to the empirical mean and the empirical variance of the exponential loss. By applying the result from Theorem 2.5, we have:

$$\mathbf{E}_{\mathcal{D}}[e^{-yf(X)}] \leq \frac{1}{n} \sum_{i=1}^n e^{-y_i f(X_i)} + \frac{15e \ln(\mathcal{M}(n)/\delta)}{(n-1)} + \sqrt{\frac{\sum_{i>j} (e^{-y_i f(X_i)} - e^{-y_j f(X_j)})^2}{n(n-1)}} \sqrt{\frac{18 \ln(\mathcal{M}(n)/\delta)}{n}},$$

where an extra e appears in the second term to normalize the exponential loss so that it has the range $[0, 1]$ (such that Theorem 2.5 applies directly).

Thus, the sample variance penalization principle, applied to the exponential loss suggests minimizing the following quantity for some scalar $\tau > 0$:

$$\sum_{i=1}^n e^{-y_i f(X_i)} + \tau \sqrt{\sum_{i>j} \left(e^{-y_i f(X_i)} - e^{-y_j f(X_j)} \right)^2}. \quad (6)$$

4 A BOOSTING ALGORITHM

In this section, a boosting style algorithm to minimize (6) is derived. Assume the function class $\mathcal{H}$ of interest is composed of so-called weak learners $G^s : \mathcal{X} \to \{\pm 1\}$ for $s = 1, \ldots, S$. The function to be output consists of the following additive model over weak learners:

$$f(X) = \sum_{s=1}^S \alpha_s G^s(X). \quad (7)$$

where $\alpha_s \in \mathbb{R}^+$ for $s = 1, \ldots, S$.

One minor technicality is that the analysis in the previous section assumed that the function f had a range $[-1, 1]$. However, while building an additive model (7) this does not hold. Thus the range of the function obtained both by AdaBoost and the proposed algorithm will be $[e^{-\sum_{s=1}^S \alpha_s}, e^{\sum_{s=1}^S \alpha_s}]$.

AdaBoost AdaBoost is described in **Algorithm** 1. Since it is a well studied algorithm, we do not provide further details. We merely point out that it minimizes an exponential loss, *i.e.,* it minimizes $\sum_{i=1}^n e^{-y_i f(X_i)}$ in a stage-wise manner to build an additive model (7).

4.1 DERIVING AN UPDATE RULE

The update of our boosting algorithm is based on the stage-wise greedy interpretation of AdaBoost (Hastie et al., 2001). Our aim is to minimize the sample variance cost (6) while building an additive model (7). As in most boosting methods, we minimize a convex cost in each stage of the boosting algorithm. Effectively, we consider the same class of weak learners as AdaBoost (*i.e.*, a conic combination of weak learners) while performing sample variance penalization instead of empirical risk minimization.

Since there is an unknown trade-off between the two terms in (6), one way to minimize that cost is by the following minimization:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^n e^{-y_i f(X_i)} \quad \text{s.t.} \sqrt{\sum_{i>j} \left(e^{-y_i f(X_i)} - e^{-y_j f(X_j)} \right)^2} \leq B,$$

where the trade-off parameter is now parametrized by B. For every value of τ there is a B that obtains the same optimal function; in particular, $B = \infty$ is equivalent to $\tau = 0$. The above problem can be equivalently

Algorithm 2 EBBoost

Require: $(X_i, y_i)_{i=1}^n$, scalar parameter $\lambda \geq 0$, weak learners $\mathcal{H}$
Initialize the weights: $w_i \leftarrow \frac{1}{n}$; initialize f to predict 0 on all inputs.
for $s \leftarrow 1$ to S **do**
Get a weak learner $G^s(\cdot)$ that minimizes (9) with the following choice of α_s:
$\alpha_s = \frac{1}{4}\log\left(\frac{(1-\lambda)(\sum_{i\in I} w_i)^2 + \lambda n \sum_{i\in I} w_i^2}{(1-\lambda)(\sum_{i\in J} w_i)^2 + \lambda n \sum_{i\in J} w_i^2}\right)$
if $\alpha_s < 0$ **then** break **end if**
$f(\cdot) \leftarrow f(\cdot) + \alpha_s G^s(\cdot)$
$w_i \leftarrow w_i \exp(-y_i G^s(X_i)\alpha_s)/Z_s$ where Z_s is such that $\sum_{i=1}^n w_i = 1$.
end for

posed as:

$$\min_{f\in\mathcal{F}} \left(\sum_{i=1}^n e^{-y_i f(X_i)}\right)^2 \quad \text{s.t.} \sum_{i>j}\left(e^{-y_i f(X_i)} - e^{-y_j f(X_j)}\right)^2 \leq B^2.$$

We now write the above optimization problem introducing a non-negative Lagrange multiplier λ:

$$\left(\sum_{i=1}^n e^{-y_i f(X_i)}\right)^2 + \lambda \sum_{i>j}\left(e^{-y_i f(X_i)} - e^{-y_j f(X_j)}\right)^2 - \lambda B^2.$$

One way to implement the above optimization is to minimize it under different settings of λ. Since the last term does not involve the function f, we merely optimize the following objective:

$$\min_{f\in\mathcal{F}} \left(\sum_{i=1}^n e^{-y_i f(X_i)}\right)^2 + \lambda \sum_{i>j}\left(e^{-y_i f(X_i)} - e^{-y_j f(X_j)}\right)^2. \tag{8}$$

Since we are interested in building an additive model, we assume that we already have a function $h(\cdot)$. We will then derive a greedy algorithm to obtain a weak learner $G(\cdot)$ and a positive scalar α such that $f(\cdot) = h(\cdot) + \alpha G(\cdot)$ minimizes the above objective the most.

Denoting $e^{-y_i h(X_i)}$ by w_i, (8) can be written as:[2]

$$\begin{aligned}
&\left(\sum_{i=1}^n w_i e^{-y_i \alpha G(X_i)}\right)^2 \\
&+ \lambda \sum_{i>j}\left(w_i e^{-y_i \alpha G(X_i)} - w_j e^{-y_j \alpha G(X_j)}\right)^2 \\
=&(1 + (n-1)\lambda)\sum_{i=1}^n w_i^2 e^{-2y_i \alpha G(X_i)} \\
&+ (2-2\lambda)\sum_{i>j} w_i w_j e^{-y_i \alpha G(X_i) - y_j \alpha G(X_j)}.
\end{aligned} \tag{9}$$

[2] In the final algorithm there will be a normalization factor dividing w_i.

For brevity, we define the following sets of indices:

$$I = \{i : y_i G(X_i) = +1\}, \quad J = \{i : y_i G(X_i) = -1\}.$$

Here, I denotes the set of examples that are correctly classified by $G(\cdot)$ and J is the set of misclassified examples. Equation (9) can now be rewritten as:

$$\begin{aligned}
&\lambda_1\left(\sum_{i\in I} w_i^2 e^{-2\alpha} + \sum_{i\in J} w_i^2 e^{2\alpha}\right) + \lambda_2 \sum_{i>j:i,j\in I} w_i w_j e^{-2\alpha} \\
&+ \lambda_2 \sum_{i>j:i,j\in J} w_i w_j e^{+2\alpha} + \lambda_2 \sum_{i>j:i\in I, j\in J \text{ or } i\in J, j\in I} w_i w_j,
\end{aligned}$$

where we have defined $\lambda_1 = (1 + (n-1)\lambda)$ and $\lambda_2 = 2 - 2\lambda$. The above expression is convex in α; it is easy to see this by taking the second derivative with respect to α. We can now minimize the above expression in α; differentiating with respect to α and equating to zero, we get:

$$\alpha = \frac{1}{4}\log\left(\frac{\lambda_1 \sum_{i\in I} w_i^2 + \lambda_2 \sum_{i>j:i,j\in I} w_i w_j}{\lambda_1 \sum_{i\in J} w_i^2 + \lambda_2 \sum_{i>j:i,j\in J} w_i w_j}\right).$$

At this point, it appears that all pairwise interactions between weights are needed to find α which would make the computation of α (given a weak learner G) cumbersome because of the $\mathcal{O}(n^2)$ pairwise terms. Consider the numerator in the update rule for α and substitute the values of λ_1 and λ_2, to get

$$\begin{aligned}
&(1 + (n-1)\lambda)\sum_{i\in I} w_i^2 + (2-2\lambda)\sum_{i>j:i,j\in I} w_i w_j \\
&= \sum_{i\in I} w_i^2 + 2\sum_{i>j:i,j\in I} w_i w_j + \lambda n \sum_{i\in I} w_i^2 \\
&- \lambda\left(\sum_{i\in I} w_i^2 + 2\sum_{i>j:i,j\in I} w_i w_j\right) \\
&= (1-\lambda)(\sum_{i\in I} w_i)^2 + \lambda n \sum_{i\in I} w_i^2.
\end{aligned}$$

Applying a similar simplification to the denominator yields the following $\mathcal{O}(n)$ rule

$$\alpha = \frac{1}{4}\log\left(\frac{(1-\lambda)(\sum_{i\in I} w_i)^2 + \lambda n \sum_{i\in I} w_i^2}{(1-\lambda)(\sum_{i\in J} w_i)^2 + \lambda n \sum_{i\in J} w_i^2}\right).$$

Table 1: For each dataset, the algorithm with the best percentage test error is represented by a shaded cell. All the bold entries in a row denote results that are not significantly different from the minimum error (by a paired t-test at 5% significance level). EBBoost outperforms AdaBoost on all datasets.

Dataset	AdaBoost	EBBoost	RLP-Boost	RQP-Boost	ABR
a5a	**18.07 ± 0.60**	**17.82 ± 0.65**	**17.90 ± 0.84**	**18.06 ± 0.86**	**17.80 ± 0.52**
abalone	**22.53 ± 0.77**	**22.38 ± 0.94**	23.68 ± 1.34	23.01 ± 1.28	**22.40 ± 0.68**
image	4.28 ± 0.78	**4.04 ± 0.78**	4.19 ± 0.80	**3.79 ± 0.68**	4.27 ± 0.82
nist09	1.28 ± 0.22	**1.17 ± 0.13**	1.43 ± 0.24	**1.25 ± 0.25**	**1.18 ± 0.18**
nist14	0.80 ± 0.18	**0.70 ± 0.11**	0.89 ± 0.16	0.78 ± 0.16	**0.74 ± 0.13**
nist27	2.56 ± 0.31	**2.41 ± 0.34**	2.72 ± 0.30	2.49 ± 0.30	**2.32 ± 0.35**
nist38	5.68 ± 0.56	**5.34 ± 0.44**	6.04 ± 0.42	5.48 ± 0.48	**5.24 ± 0.47**
nist56	3.64 ± 0.45	**3.38 ± 0.37**	3.97 ± 0.47	3.61 ± 0.43	**3.42 ± 0.35**
mushrooms	0.35 ± 0.35	**0.28 ± 0.30**	**0.30 ± 0.34**	**0.30 ± 0.34**	**0.29 ± 0.37**
musklarge	7.80 ± 0.99	**6.89 ± 0.58**	7.83 ± 1.00	7.29 ± 0.96	7.22 ± 0.71
ringnorm	15.05 ± 3.14	**13.45 ± 2.37**	**15.25 ± 4.15**	**14.55 ± 2.98**	**14.35 ± 3.13**
spambase	7.74 ± 0.74	**7.18 ± 0.79**	7.45 ± 0.60	7.25 ± 0.69	**6.99 ± 0.62**
splice	10.57 ± 1.13	**10.27 ± 0.85**	**10.28 ± 0.85**	**10.18 ± 0.99**	**10.02 ± 0.86**
twonorm	4.30 ± 0.40	**4.00 ± 0.21**	4.87 ± 0.47	4.19 ± 0.39	4.16 ± 0.45
w4a	**2.80 ± 0.23**	**2.75 ± 0.23**	**2.76 ± 0.14**	**2.77 ± 0.23**	**2.75 ± 0.19**
waveform	12.96 ± 0.75	12.90 ± 0.81	12.75 ± 0.86	**12.22 ± 0.90**	**12.47 ± 0.71**
wine	26.03 ± 1.18	25.66 ± 1.00	**25.00 ± 1.16**	**25.20 ± 0.96**	**25.09 ± 1.17**
wisconsin	5.00 ± 1.50	**4.00 ± 1.34**	**4.14 ± 1.50**	4.71 ± 1.54	**4.46 ± 1.58**

We now state the algorithm based on sample variance penalization (6) in **Algorithm** 2. It merely requires the sum of weights on examples and the sum of squared weights on appropriate partitions defined by the weak learner. Further, given a weak learner, we note that (9) only requires $\mathcal{O}(n)$ time to evaluate.

It is easy to see that AdaBoost is a specific instance of EBBoost algorithm. If we substitute $\lambda = 0$, (6) becomes $\left(\sum_{i=1}^{n} e^{-y_i f(X_i)}\right)^2$. Even though this cost function is the AdaBoost cost squared, the optimal choice of α remains the same since the AdaBoost cost is non-negative. Substituting, $\lambda = 0$ in the expression for α above, we have,

$$\alpha = \frac{1}{4}\log\left(\frac{(\sum_{i\in I} w_i)^2}{(\sum_{i\in J} w_i)^2}\right) = \frac{1}{2}\log\left(\frac{\sum_{i\in I} w_i}{\sum_{i\in J} w_i}\right),$$

which coincides with the choice of α in AdaBoost.

5 EXPERIMENTS

In this section, we evaluate the empirical performance of EBBoost and AdaBoost. This is our *primary comparison* as the goal of this article is to compare between empirical risk minimization (AdaBoost) and sample variance penalization (EBBoost). There are numerous variants of boosting algorithms; many of these variants are exploring other intricacies of the classification problems (sparsity, robustness to outliers and so forth). While performance is reported for three variants, the goal is not to find a method that outperforms every variant of boosting under every possible choice of weak learners. In fact, many of these boosting variants could also be modified to incorporate sample variance penalization. The emphasis, rather, is on comparing AdaBoost with EBBoost with a simple family of weak learners such as decision stumps. Experiments with three other boosting variants are included simply to give broader perspective. In our experiments, we consider the following boosting variants:

Regularized LP and QP Boost Given a dataset, first AdaBoost is run to obtain training predictions from weak learners. LP and QP Boost algorithms (Raetsch et al., 2001) then optimize the weights on weak learners obtained by AdaBoost to maximize the margin (along with regularization on the weights). Once the optimizations are solved, predictions are obtained based on the outputs of the same weak learners on the test set.

Boosting with Soft Margin $\text{AdaBoost}_{\text{REG}}$ (ABR) (Raetsch et al., 2001) optimizes a "soft margin" by allowing slacks on examples; this is better suited to handle noisy situations.

We performed experiments on a number of publicly available datasets. We did experiments only on datasets that had at least 400 examples so that both validation and test sets had at least 100 examples. For each dataset, we took the minimum of half of the examples (or 500 examples) as training. This was done

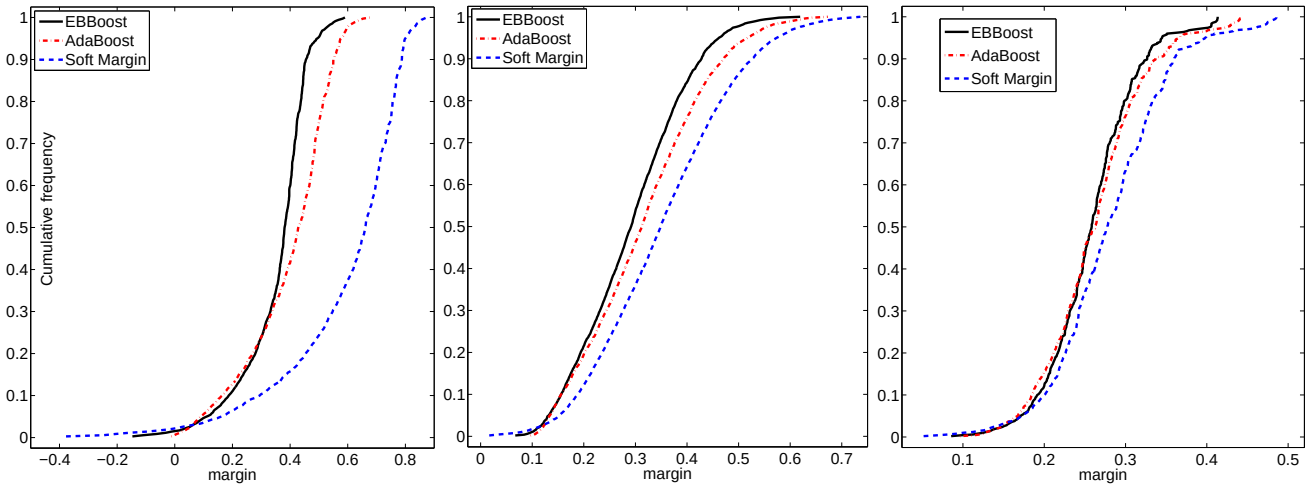

Figure 1: Cumulative margin distributions on three different datasets (wisconsin, mnist27, mushrooms). ABR obtains a long tail indicating its "slackness". EBBoost's margins are characterized by a smaller variance.

since solving an LP or QP with a large number of examples can be quite expensive compared to boosting. Moreover, ABR requires a line search which can also be much slower than AdaBoost. The remaining examples in each dataset were divided equally into validation and test sets by a random split. For AdaBoost, EBBoost, and ABR, we considered 500 randomly generated decision stumps as the weak learners. Each algorithm was run until there was no drop in the validation error rate in 50 iterations; the corresponding test error rate was then noted. The set of weak learners recovered by AdaBoost was given to regularized LP and QP boosting procedures. For all methods (other than AdaBoost) there is an extra parameter to tune. We found the value of the parameter that resulted in the minimum error on the validation set and then report the corresponding test error. The experiment was repeated 20 times over random splits of training, test, and validation sets. Results are reported in Table 1.

EBBoost shows significant improvement over AdaBoost on most of the datasets; in fact, it shows an improvement over every single dataset. ABR's performance comes closest to EBBoost even though the methods are qualitatively quite different. In fact, it is straightforward to obtain a soft margin version of EBBoost by replacing our choice of loss function. Moreover, in the following section, it will be shown that the performance gains of EBBoost and ABR emerge for completely different reasons and the intuitions underlying the two may be complementary.

5.1 DISCUSSION

Since EBBoost and ABR showed similar performance overall, it is interesting to see how the solutions differ. We looked at the margin distribution of the training examples on all the datasets. The effectiveness of boosting can be (to some extent), explained by the margin distribution (Schapire et al., 1998; Koltchinskii & Panchenko, 2002). Recall the definition of margin on an example X_i: $\gamma(X_i, y_i) = y_i \sum_{s=1}^{S} \alpha_s G^s(X_i) / \sum_{s=1}^{S} \alpha_s$, based on the additive model (7).

We visualized the training margin distributions of all datasets. These plots show the average margin distribution over the experiments at the setting of the

Table 2: Mean and standard deviation of margins. Some dataset names have been abbreviated due to space constraints.

	AdaBoost	EBBoost	ABR
a5a	0.21 ± 0.20	0.19 ± 0.17	0.20 ± 0.19
abal	0.12 ± 0.12	0.12 ± 0.12	0.13 ± 0.13
image	0.14 ± 0.08	0.13 ± 0.06	0.14 ± 0.08
nist09	0.45 ± 0.13	0.44 ± 0.12	0.48 ± 0.13
nist14	0.47 ± 0.12	0.38 ± 0.07	0.51 ± 0.12
nist27	0.32 ± 0.12	0.29 ± 0.10	0.35 ± 0.13
nist38	0.22 ± 0.10	0.20 ± 0.08	0.24 ± 0.10
nist56	0.30 ± 0.12	0.29 ± 0.11	0.32 ± 0.13
mush	0.26 ± 0.06	0.26 ± 0.05	0.28 ± 0.07
musk	0.18 ± 0.09	0.15 ± 0.06	0.18 ± 0.09
ring	0.15 ± 0.07	0.14 ± 0.06	0.15 ± 0.07
spam	0.21 ± 0.13	0.19 ± 0.10	0.23 ± 0.13
splice	0.19 ± 0.12	0.18 ± 0.10	0.22 ± 0.14
twon	0.29 ± 0.14	0.26 ± 0.11	0.30 ± 0.14
w4a	0.27 ± 0.11	0.23 ± 0.07	0.38 ± 0.12
wave	0.25 ± 0.17	0.22 ± 0.14	0.28 ± 0.19
wine	0.13 ± 0.15	0.13 ± 0.14	0.12 ± 0.14
wisc	0.39 ± 0.15	0.35 ± 0.12	0.59 ± 0.21

parameters selected by validation. We only present results for AdaBoost, EBBoost, and ABR due to space constraints. Three typical cumulative margin distribution plots are shown in Figure 1. Even though ABR and EBBoost showed similar test error rates, they have fairly different margin distributions. Typically, ABR has a long tail over incorrect predictions (due to its use of slack on hard to classify examples) whereas EBBoost is characterized by a small variance (not surprisingly, since we are minimizing the variance with an exponential loss). In addition, we obtained the mean and standard deviation of all the margin values on all datasets. Table 2 summarizes those results. ABR obtains larger mean margin as well as large standard deviations. EBBoost typically obtains slightly smaller margins compared to AdaBoost but with much smaller variances. However, both EBBoost and ABR show accuracy improvements over AdaBoost. We believe, the improvements of ABR are due to its ability to handle noisy situations and outliers more gracefully. The performance advantage of EBBoost is justified by the empirical Bernstein bound (our initial motivation). Typically, the margin distribution bounds do not explicitly account for variance information; an interesting direction for future research is to explore the relationship between the empirical Bernstein bounds as well as previous analyses of the margin distribution.

6 CONCLUSIONS

We proposed a novel boosting algorithm based on the empirical Bernstein inequality. The algorithm is as easy to implement as AdaBoost and is as efficient computationally (it does not require an expensive line search). EBBoost showed significant empirical advantages over AdaBoost. This paper demonstrates that it is possible to design efficient algorithms based on sample variance penalization and to obtain improvements in test error. In addition to additional theoretical work to refine these generalization bounds, another direction of future research is to automatically estimate the parameter λ without cross-validation in order to make EBBoost free from additional parameters.

Acknowledgments The authors thank Cynthia Rudin, Phil Long and Rocco Servedio for helpful discussions. The authors acknowledge support from DHS Contract N66001-09-C-0080—"Privacy Preserving Sharing of Network Trace Data (PPSNTD) Program".

References

Audibert, J.-Y., Munos, R., & Szepesvári, C. (2007). Tuning bandit algorithms in stochastic environments. *18th Algorithmic Learning Theory* (pp. 150–165).

Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, *101*, 138–156.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2002). A second-order perceptron algorithm. *in Proc. 5th Annu. Conf. Computational Learning Theory (Lecture Notes in Artificial Intelligence)* (pp. 121–137).

Crammer, K., Dredze, M., & Pereira, F. (2009a). Exact convex confidence-weighted learning. *Advances in Neural Information Processing Systems 21*. Cambridge, MA: MIT Press.

Crammer, K., Mohri, M., & Pereira, F. (2009b). Gaussian margin machines. *Proceedings of the Artificial Intelligence and Statistics*.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer-Verlag.

Koltchinskii, V., & Panchenko, D. (2002). Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, *30*.

Maurer, A., & Pontil, M. (2009). Empirical Bernstein bounds and sample variance penalization. *22nd Annual Conference on Learning Theory*.

Mnih, V., Szepesvári, C., & Audibert, J.-Y. (2008). Empirical Bernstein stopping. *Proceedings of the Twenty-Fifth International Conference on Machine Learning* (pp. 672–679).

Raetsch, G., Onoda, T., & Muller, K.-R. (2001). Soft margins for adaboost. *Machine Learning*, *43*, 287–320.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*, 322–330.

Shivaswamy, P., & Jebara, T. (2010). Maximum relative margin and data-dependent regularization. *Journal of Machine Learning Research*, *11*, 747–788.

Vapnik, V. (1995). *The nature of statistical learning*. New York, NY: Springer.