On the Dual Formulation of Boosting Algorithms

Chunhua Shen and Hanxi Li

Abstract—We study boosting algorithms from a new perspective. We show that the Lagrange dual problems of ℓ_1-norm-regularized AdaBoost, LogitBoost, and soft-margin LPBoost with generalized hinge loss are all entropy maximization problems. By looking at the dual problems of these boosting algorithms, we show that the success of boosting algorithms can be understood in terms of maintaining a better margin distribution by maximizing margins and at the same time controlling the margin variance. We also theoretically prove that approximately, ℓ_1-norm-regularized AdaBoost maximizes the average margin, instead of the minimum margin. The duality formulation also enables us to develop column-generation-based optimization algorithms, which are totally corrective. We show that they exhibit almost identical classification results to that of standard stagewise additive boosting algorithms but with much faster convergence rates. Therefore, fewer weak classifiers are needed to build the ensemble using our proposed optimization technique.

Index Terms—AdaBoost, LogitBoost, LPBoost, Lagrange duality, linear programming, entropy maximization.

1 Introduction

Boosting has attracted a lot of research interests since the first practical boosting algorithm, AdaBoost, was introduced by Freund and Schapire [1]. The machine learning community has spent much effort on understanding how the algorithm works [2], [3], [4]. However, to date there are still questions about the success of boosting that are left unanswered [5]. In boosting, one is given a set of training examples $x_i \in \mathcal{X}$, $i = 1 \cdots M$, with binary labels y_i being either $+1$ or -1. A boosting algorithm finds a convex linear combination of weak classifiers (a.k.a. base learners, weak hypotheses) that can achieve much better classification accuracy than an individual base classifier. To do so, there are two unknown variables to be optimized. The first one is the base classifier. An oracle is needed to produce base classifiers. The second one is the positive weights associated with each base classifier.

AdaBoost is one of the first and the most popular boosting algorithms for classification. Various boosting algorithms have since been advocated. For example, LogitBoost by Friedman et al. [6] replaces AdaBoost's exponential cost function with the function of logistic regression. MadaBoost [7] instead uses a modified exponential loss. The authors of [6] consider boosting algorithms with a generalized additive model framework. Schapire et al. [2] showed that AdaBoost converges to a large margin solution. However, recently, it was pointed out that AdaBoost does not converge to the maximum margin solution [4], [8]. Motivated by the success of the margin theory associated with support vector machines (SVMs), LPBoost was proposed by [9], [10] with the intuition of maximizing the minimum margin of all training examples. The final optimization problem can be formulated as a linear program (LP). It is observed that the hard-margin LPBoost does not perform well in most cases, although it usually produces larger minimum margins. More often, LPBoost has worse generalization performance. In other words, a higher minimum margin would not necessarily imply a lower test error. Breiman [11] also noticed the same phenomenon: His Arc-Gv algorithm has a minimum margin that provably converges to the optimal, but Arc-Gv is inferior in terms of generalization capability. Experiments on LPBoost and Arc-Gv have put the margin theory into serious doubt. Recently, Reyzin and Schapire [12] reran Breiman's experiments by controlling weak classifiers' complexity. They found that the minimum margin is indeed larger for Arc-Gv, but the overall margin distribution is typically better for AdaBoost. The conclusion is that the minimum margin is important, but not always at the expense of other factors. They also conjectured that maximizing the average margin, instead of the minimum margin, may result in better boosting algorithms. Recent theoretical work [13] has shown the important role of the margin distribution on bounding the generalization error of combined classifiers such as boosting and bagging.

As the soft-margin SVM usually has a better classification accuracy than the hard-margin SVM, the soft-margin LPBoost also performs better by relaxing the constraints that all training examples must be correctly classified. Cross validation is required to determine an optimal value for the soft-margin trade-off parameter. Rätsch et al. [14] showed the equivalence between SVMs and boosting-like algorithms. Comprehensive overviews on boosting are given by [15] and [16].

We show in this work that the Lagrange duals of ℓ_1-norm-regularized AdaBoost, LogitBoost, and LPBoost with generalized hinge loss are all entropy maximization problems. Previous work like [17], [18], [19] noticed the connection between boosting techniques and entropy maximization based on Bregman distances. They did not show that the duals of boosting algorithms are actually entropy-regularized LPBoost, as we show in (10), (28), and

• The authors are with NICTA, Canberra Research Laboratory, Locked Bag 8001, Canberra, ACT 2601, Australia, and the Australian National University, Canberra, ACT 0200, Australia.
E-mail: {chunhua.shen, hanxi.li}@nicta.com.au.

Manuscript received 15 Mar. 2009; revised 23 Aug. 2009; accepted 12 Dec. 2009; published online 25 Feb. 2010.
Recommended for acceptance by J. Matas.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-03-0166.
Digital Object Identifier no. 10.1109/TPAMI.2010.47.

(31). By knowing this duality equivalence, we derive a general column generation (CG)-based optimization framework that can be used to optimize arbitrary convex loss functions. In other words, we can easily design totally corrective AdaBoost, LogitBoost, boosting with generalized hinge loss, etc.

Our major contributions are the following:

1. We derive the Lagrangian duals of boosting algorithms and show that most of them are entropy maximization problems.
2. The authors of [12] conjectured that "it may be fruitful to consider boosting algorithms that greedily maximize the average or median margin rather than the minimum one." We theoretically prove that, actually, ℓ_1-norm-regularized AdaBoost approximately maximizes the average margin, instead of the minimum margin. This is an important result in the sense that it provides an alternative theoretical explanation that is consistent with the margins theory and agrees with the empirical observations made by [12].
3. We propose AdaBoost-QP, which directly optimizes the asymptotic cost function of AdaBoost. The experiments confirm our theoretical analysis.
4. Furthermore, based on the duals we derive, we design column generation-based optimization techniques for boosting learning. We show that the new algorithms have almost identical results to that of standard stagewise additive boosting algorithms but with much faster convergence rates. Therefore, fewer weak classifiers are needed to build the ensemble.

The following notation is used. Typically, we use bold letters $\boldsymbol{u}, \boldsymbol{v}$ to denote vectors, as opposed to scalars u, v in lower case letters. We use capital letters U, V to denote matrices. All vectors are column vectors unless otherwise specified. The inner product of two column vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is $\boldsymbol{u}^\top \boldsymbol{v} = \sum_i u_i v_i$. Componentwise inequalities are expressed using symbols $\geq, >, \leq, <$, e.g., $\boldsymbol{u} \geq \boldsymbol{v}$ means for all of the entries $u_i \geq v_i$. $\mathbf{0}$ and $\mathbf{1}$ are column vectors with each entry being 0 and 1, respectively. The length will be clear from the context. The abbreviation s.t. means "subject to." We denote the domain of a function $f(\cdot)$ as $\mathrm{dom}\, f$.

The paper is organized as follows: Section 2 briefly reviews several boosting algorithms for self-completeness. Their corresponding duals are derived in Section 3. Our main results are also presented in Section 3. In Section 4, we then present numerical experiments to illustrate various aspects of our new algorithms obtained in Section 3. We conclude the paper in the last section.

2 Boosting Algorithms

We first review some basic ideas and the corresponding optimization problems of AdaBoost, LPBoost, and LogitBoost which are of interest in this work.

Let $\mathcal{H}$ be a class of base classifier $\mathcal{H} = \{h_j(\cdot) : \mathcal{X} \to \mathbb{R}, j = 1 \cdots N\}$. A boosting algorithm seeks for a convex linear combination:

$$F(\boldsymbol{w}) = \sum_{j=1}^{N} w_j h_j(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{w}$ is the weak classifier weights to be optimized. AdaBoost calls an oracle that selects a weak classifier $h_j(\cdot)$ at each iteration j and then calculates the weight w_j associated with $h_j(\cdot)$. It is shown in [6], [20] that AdaBoost (and many others, like LogitBoost) performs coordinate gradient descent in function space, at each iteration choosing a weak classifier to include in the combination such that the cost function is maximally reduced. It is well known that coordinate descent has a slow convergence in many cases. From an optimization point of view, there is no particular reason to keep the weights $w_1, \ldots, w_{j-1}$ fixed at iteration j. Here, we focus on the underlying mathematical programs that boosting algorithms minimize.

AdaBoost has proven to minimize the exponential loss function [17]:

$$\min_{\boldsymbol{w}} \sum_{i=1}^{M} \exp(-y_i F(\boldsymbol{x}_i)), \quad \text{s.t. } \boldsymbol{w} \geq \mathbf{0}. \tag{2}$$

Because the logarithmic function $\log(\cdot)$ is a strictly monotonically increasing function, AdaBoost equivalently solves

$$\min_{\boldsymbol{w}} \log\left(\sum_{i=1}^{M} \exp(-y_i F(\boldsymbol{x}_i))\right), \quad \text{s.t. } \boldsymbol{w} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{w} = \frac{1}{T}. \tag{3}$$

Note that in the AdaBoost algorithm, the constraint $\mathbf{1}^\top \boldsymbol{w} = \frac{1}{T}$ is not explicitly enforced. However, without this regularization constraint, in the case of separable training data, one can always make the cost function approach zero via enlarging the solution $\boldsymbol{w}$ by an arbitrarily large factor. Here, what matters is the sign of the classification evaluation function. Standard AdaBoost seems to select the value of T by selecting how many iterations it runs. Note that the relaxed version $\mathbf{1}^\top \boldsymbol{w} \leq \frac{1}{T}$ is actually equivalent to $\mathbf{1}^\top \boldsymbol{w} = \frac{1}{T}$.[1] With the constraint $\mathbf{1}^\top \boldsymbol{w} \leq \frac{1}{T}$, if the final solution has $\mathbf{1}^\top \boldsymbol{w} < \frac{1}{T}$, one can scale $\boldsymbol{w}$ such that $\mathbf{1}^\top \boldsymbol{w} = \frac{1}{T}$ and, clearly, the scaled $\boldsymbol{w}$ achieves a smaller loss. So, the optimum must be achieved at the boundary.

The boosting algorithm introduced in (3) is a ℓ_1-norm-regularized version of the original AdaBoost because it is equivalent to

$$\min_{\boldsymbol{w}} \log\left(\sum_{i=1}^{M} \exp(-y_i F(\boldsymbol{x}_i))\right) + \frac{1}{T'} \mathbf{1}^\top \boldsymbol{w}, \quad \text{s.t. } \boldsymbol{w} \geq \mathbf{0}. \tag{4}$$

For a certain T, one can always find a T'such that (3) and (4) have exactly the same solution. Hereafter, we refer to this algorithm as AdaBoost$_{\ell 1}$.

We will show that it is very important to introduce this new cost function. All of our main results on AdaBoost$_{\ell 1}$ are obtained by analyzing this logarithmic cost function, not the original cost function. Let us define the matrix $H \in \mathbb{Z}^{M \times N}$, which contains all of the possible predictions of the training data using weak classifiers from the pool $\mathcal{H}$. Explicitly $H_{ij} = h_j(\boldsymbol{x}_i)$ is the label ($\{+1, -1\}$) given by weak classifier $h_j(\cdot)$ on the training example $\boldsymbol{x}_i$. We use $H_i = [H_{i1} H_{i2} \cdots H_{iN}]$ to denote the ith row of H, which constitutes the output of all the weak classifiers on the training example $\boldsymbol{x}_i$. The cost function of AdaBoost$_{\ell 1}$ writes:

1. The reason why we do not write this constraint as $\mathbf{1}^\top \boldsymbol{w} = T$ will become clear later.

$$\min_{\boldsymbol{w}} \log\left(\sum_{i=1}^{M}\exp(-y_i H_i \boldsymbol{w})\right), \quad \text{s.t. } \boldsymbol{w} \geq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{w} = \frac{1}{T}. \tag{5}$$

We can also write the above program into

$$\min_{\boldsymbol{w}} \log\left(\sum_{i=1}^{M}\exp\left(-\frac{y_i H_i \boldsymbol{w}}{T}\right)\right), \quad \text{s.t. } \boldsymbol{w} \geq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{w} = 1, \tag{6}$$

which is exactly the same as (5). In [4], the smooth margin that is similar to but different from the logarithmic cost function is used to analyze AdaBoost's convergence behavior. The smooth margin in [4] is defined as

$$\frac{-\log(\sum_{i=1}^{M}\exp(-y_i H_i \boldsymbol{w}))}{\boldsymbol{1}^\top \boldsymbol{w}}.$$

Problem (5) (or (6)) is a convex problem in $\boldsymbol{w}$. We know that the log-sum-exp function

$$\operatorname{lse}(\boldsymbol{x}) = \log\left(\sum_{i=1}^{M}\exp x_i\right)$$

is convex [21]. Composition with an affine mapping preserves convexity. Therefore, the cost function is convex. The constraints are linear, hence convex, too. For completeness, we include the description of the standard stagewise AdaBoost and Arc-Gv in Algorithm 1. The only difference between these two algorithms is the way to calculate w_j (step (2) of Algorithm 1). For AdaBoost:

$$w_j = \frac{1}{2}\log\frac{1+r_j}{1-r_j}, \tag{7}$$

where r_j is the edge of the weak classifier $h_j(\cdot)$ defined as $r_j = \sum_{i=1}^{M} u_i y_i h_j(\boldsymbol{x}_i) = \sum_{i=1}^{M} u_i y_i H_{ij}$. Arc-Gv modifies (7) in order to maximize the minimum margin:

$$w_j = \frac{1}{2}\log\frac{1+r_j}{1-r_j} - \frac{1}{2}\log\frac{1+\varrho_j}{1-\varrho_j}, \tag{8}$$

where ϱ_j is the minimum margin over all training examples of the combined classifier up to the current round:

$$\varrho_j = \min_i \left\{ y_i \sum_{s=1}^{j-1} w_s h_s(\boldsymbol{x}_i) \Big/ \sum_{s=1}^{j-1} w_s \right\},$$

with $\varrho_1 = 0$. Arc-Gv clips w_j into $[0,1]$ by setting $w_j = 1$ if $w_j > 1$ and $w_j = 0$ if $w_j < 0$ [11]. Other work, such as [8], [22], has used different approaches to determine ϱ_j in (8).

Algorithm 1 Stage-wise AdaBoost, and Arc-Gv.

```
  Input: Training set $(\boldsymbol{x}_i, y_i), y_i = \{+1,-1\}, i = 1 \cdots M$;
         maximum iteration $N_{\max}$.
1 Initialization: $u_i^0 = \frac{1}{M}, \forall i = 1 \cdots M$.
2 for $j = 1, \cdots, N_{\max}$ do
      1) Find a new base $h_j(\cdot)$ using the distribution $\boldsymbol{u}^j$;
      2) Choose $w_j$;
      3) Update $\boldsymbol{u}$: $u_i^{j+1} \propto u_i^j \exp(-y_i w_j h_j(\boldsymbol{x}_i))$, $\forall i$; and
         normalize $\boldsymbol{u}^{j+1}$.
  Output: The learned classifier $F(\boldsymbol{x}) = \sum_{j=1}^{N} w_j h_j(\boldsymbol{x})$.
```

3 Lagrange Dual of Boosting Algorithms

Our main derivations are based on a form of duality termed convex conjugate or Fenchel duality.

Definition 3.1 (Convex conjugate). *Let $f : \mathbb{R}^n \to \mathbb{R}$. The function $f^* : \mathbb{R}^n \to \mathbb{R}$, defined as*

$$f^*(\boldsymbol{u}) = \sup_{\boldsymbol{x} \in \operatorname{dom} f} \left(\boldsymbol{u}^\top \boldsymbol{x} - f(\boldsymbol{x})\right), \tag{9}$$

is called the convex conjugate (or Fenchel duality) of the function $f(\cdot)$. The domain of the conjugate function consists of $\boldsymbol{u} \in \mathbb{R}^n$ for which the supremum is finite.

$f^*(\cdot)$ is always a convex function because it is the pointwise supremum of a family of affine functions of $\boldsymbol{u}$. This is true even if $f(\cdot)$ is nonconvex [21].

Proposition 3.1 (Conjugate of log-sum-exp). *The conjugate of the log-sum-exp function is the negative entropy function, restricted to the probability simplex. Formally, for $\operatorname{lse}(\boldsymbol{x}) = \log(\sum_{i=1}^{M}\exp x_i)$, its conjugate is*

$$\operatorname{lse}^*(\boldsymbol{u}) = \begin{cases} \sum_{i=1}^{M} u_i \log u_i, & \text{if } \boldsymbol{u} \geq \boldsymbol{0} \text{ and } \boldsymbol{1}^\top \boldsymbol{u} = 1; \\ \infty, & \text{otherwise.} \end{cases}$$

We interpret $0 \log 0$ as 0.

This result is given in [21, Chapter 3.3].

Theorem 3.1. *The dual of AdaBoost$_{\ell 1}$ is a Shannon entropy maximization problem which writes*

$$\begin{aligned} \max_{r,\boldsymbol{u}} \quad & \frac{r}{T} - \sum_{i=1}^{M} u_i \log u_i \\ \text{s.t.} \quad & \sum_{i=1}^{M} u_i y_i H_i \leq -r\boldsymbol{1}^\top, \\ & \boldsymbol{u} \geq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{u} = 1. \end{aligned} \tag{10}$$

Proof. To derive a Lagrange dual of AdaBoost$_{\ell 1}$, we first introduce a new variable $\boldsymbol{z} \in \mathbb{R}^M$, such that its ith entry $z_i = -y_i H_i \boldsymbol{w}$, to obtain the equivalent problem

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \log\left(\sum_{i=1}^{M}\exp z_i\right) \\ \text{s.t.} \quad & z_i = -y_i H_i \boldsymbol{w} \ (\forall i = 1, \ldots, M), \\ & \boldsymbol{w} \geq \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{w} = \frac{1}{T}. \end{aligned} \tag{11}$$

The Lagrangian $L(\cdot)$ associated with the problem (5) is

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{q}, r) = {} & \log\left(\sum_{i=1}^{M}\exp z_i\right) - \sum_{i=1}^{M} u_i(z_i + y_i H_i \boldsymbol{w}) \\ & - \boldsymbol{q}^\top \boldsymbol{w} - r\left(\boldsymbol{1}^\top \boldsymbol{w} - \frac{1}{T}\right), \end{aligned} \tag{12}$$

with $\boldsymbol{q} \geq 0$. The dual function is

$$\begin{aligned}
\inf_{\boldsymbol{z},\boldsymbol{w}} L &= \inf_{\boldsymbol{z},\boldsymbol{w}} \log\left(\sum_{i=1}^{M} \exp z_i\right) - \sum_{i=1}^{M} u_i z_i + \frac{r}{T} \\
&\quad - \overbrace{\left(\sum_{i=1}^{M} u_i y_i H_i + \boldsymbol{q}^\top + r\mathbf{1}^\top\right)}^{\text{must be } \mathbf{0}} \boldsymbol{w} \\
&= \inf_{\boldsymbol{z}} \log\left(\sum_{i=1}^{M} \exp z_i\right) - \boldsymbol{u}^\top \boldsymbol{z} + \frac{r}{T} \qquad (13) \\
&= \overbrace{-\sup_{\boldsymbol{z}}\left[\boldsymbol{u}^\top \boldsymbol{z} - \log\left(\sum_{i=1}^{M} \exp z_i\right)\right]}^{-\operatorname{lse}^*(\boldsymbol{u})(\text{see Proposition 3.1})} + \frac{r}{T} \\
&= -\sum_{i=1}^{M} u_i \log u_i + \frac{r}{T}.
\end{aligned}$$

By collecting all of the constraints and eliminating $\boldsymbol{q}$, the dual of Problem (5) is (10). □

Keeping two variables $\boldsymbol{w}$ and $\boldsymbol{z}$ and introducing new equality constraints $z_i = -y_i H_i \boldsymbol{w}$, $\forall i$, is essential to derive the above simple and elegant Lagrange dual. Simple equivalent reformulations of a problem can lead to very different dual problems. Without introducing new variables and equality constraints, one would not be able to obtain (10). Here, we have considered the *negative margin* z_i to be the central objects of study. In [23], a similar idea has been used to derive different duals of kernel methods, which leads to the so-called *value regularization*. We focus on boosting algorithms instead of kernel methods in this work. Also, note that we would have the following dual if we work directly on the cost function in (3):

$$\begin{aligned}
&\max_{r,\boldsymbol{u}} \frac{r}{T} - \sum_{i=1}^{M} u_i \log u_i + \mathbf{1}^\top \boldsymbol{u} \\
&\text{s.t.} \sum_{i=1}^{M} u_i y_i H_i \leq -r\mathbf{1}^\top, \boldsymbol{u} \geq \mathbf{0}.
\end{aligned} \tag{14}$$

No normalization requirement $\mathbf{1}^\top \boldsymbol{u} = 1$ is imposed. Instead, $\mathbf{1}^\top \boldsymbol{u}$ works as a regularization term. The connection between AdaBoost and LPBoost is not clear with this dual.

Lagrange duality between problems (5) and (10) assures that weak duality and strong duality hold. Weak duality says that any feasible solution of (10) produces a lower bound of the original problem (5). Strong duality tells us the optimal value of (10) is the same as the optimal value of (5). The weak duality is guaranteed by the Lagrange duality theory. The strong duality holds since the primal problem (5) is a convex problem that satisfies Slater's condition [21].

To show the connection with LPBoost, we equivalently rewrite the above formulation by reversing the sign of r and multiplying the cost function with T ($T > 0$):

$$\begin{aligned}
&\min_{r,\boldsymbol{u}} r + T\sum_{i=1}^{M} u_i \log u_i \\
&\text{s.t.} \sum_{i=1}^{M} u_i y_i H_{ij} \leq r (\forall j = 1, \ldots, N), \\
&\qquad \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1.
\end{aligned} \tag{15}$$

Note that the constraint $\boldsymbol{u} \geq \mathbf{0}$ is implicitly enforced by the logarithmic function, and thus, it can be dropped when one solves (3).

3.1 Connection between AdaBoost$_{\ell 1}$ and Gibbs Free Energy

Gibbs free energy is the chemical potential that is minimized when a system reaches equilibrium at constant pressure and temperature.

Let us consider a system that has M states at temperature T. Each state has energy v_i and probability u_i of likelihood of occurring. The Gibbs free energy of this system is related with its average energy and entropy, namely:

$$G(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{u}^\top \boldsymbol{v} + T\sum_{i=1}^{M} u_i \log u_i. \tag{16}$$

When the system reaches equilibrium, $G(\boldsymbol{v}, \boldsymbol{u})$ is minimized. So, we have

$$\min_{\boldsymbol{u}} \ G(\boldsymbol{v}, \boldsymbol{u}), \ \text{s.t.}\ \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1. \tag{17}$$

The constraints ensure that $\boldsymbol{u}$ is a probability distribution.

Now, let us define vector $\boldsymbol{v}_j$ with its entries being $v_{ij} = y_i H_{ij}$, where v_{ij} is the energy associated with state i for case j. v_{ij} can only take discrete binary values $+1$ or -1. We rewrite our dual optimization problem (15) into

$$\begin{aligned}
&\min_{\boldsymbol{u}} \quad \overbrace{\max_j \{\boldsymbol{u}^\top \boldsymbol{v}_j\}}^{\text{worst case energy vector } \boldsymbol{v}_j} + T\sum_{i=1}^{M} u_i \log u_i, \\
&\text{s.t.} \quad \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1.
\end{aligned} \tag{18}$$

This can be interpreted as finding the minimum Gibbs free energy for the *worst*-case energy vector.

3.2 Connection between AdaBoost$_{\ell 1}$ and LPBoost

First, let us recall the basic concepts of LPBoost. The idea of LPBoost is to maximize the minimum margin because it is believed that the minimum margin plays a critically important role in terms of generalization error [2]. The hard-margin LPBoost [9] can be formulated as

$$\max_{\boldsymbol{w}} \ \overbrace{\min_i \{y_i H_i \boldsymbol{w}\}}^{\text{minimum margin}}, \ \text{s.t.}\ \boldsymbol{w} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{w} = 1. \tag{19}$$

This problem can be solved as an LP. Its dual is also an LP:

$$\begin{aligned}
&\min_{r,\boldsymbol{u}} \ r \ \text{ s.t.} \ \sum_{i=1}^{M} u_i y_i H_{ij} \leq r (\forall j = 1, \ldots, N), \\
&\qquad\qquad \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1.
\end{aligned} \tag{20}$$

Arc-Gv has been shown asymptotically to a solution of the above LPs [11].

The performance deteriorates when no linear combination of weak classifiers can be found that separates the training examples. By introducing slack variables, we get the soft-margin LPBoost algorithm:

$$\begin{aligned}
&\max_{\boldsymbol{w},\varrho,\boldsymbol{\xi}} \varrho - D\mathbf{1}^\top \boldsymbol{\xi} \\
&\quad \text{s.t.}\ y_i H_i \boldsymbol{w} \geq \varrho - \xi_i, (\forall i = 1, \ldots, M), \\
&\qquad \boldsymbol{w} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{w} = 1, \boldsymbol{\xi} \geq \mathbf{0}.
\end{aligned} \tag{21}$$

Here, D is a trade-off parameter that controls the balance between training error and margin maximization. The dual of (21) is similar to the hard-margin case except that the dual variable $\boldsymbol{u}$ is capped:

$$\min_{r,\boldsymbol{u}} \ r \ \text{ s.t.} \sum_{i=1}^{M} u_i y_i H_{ij} \le r \ (\forall j = 1, \ldots, N), \quad D\mathbf{1} \ge \boldsymbol{u} \ge \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1. \tag{22}$$

Comparing (15) with hard-margin LPBoost's dual, it is easy to see that the only difference is the entropy term in the cost function. If we set $T = 0$, (15) reduces to the hard-margin LPBoost. In this sense, we can view AdaBoost$_{\ell 1}$'s dual as entropy-regularized hard-margin LPBoost. Since the regularization coefficient T is always positive, the effects of the entropy regularization term are to encourage the distribution $\boldsymbol{u}$ to be as uniform as possible (the negative entropy $\sum_{i=1}^{M} u_i \log u_i$ is the Kullback-Leibler distance between $\boldsymbol{u}$ and the uniform distribution). This may explain the underlying reason for AdaBoost's success over hard-margin LPBoost: limiting the weight distribution $\boldsymbol{u}$ leads to better generalization performance. But, *why and how*? We will discover the mechanism in Section 3.3.

When the regularization coefficient T is sufficiently large, the entropy term in the cost function dominates. In this case, all discrete probabilities u_i become almost the same and therefore gather around the center of the simplex $\{\boldsymbol{u} \ge \mathbf{0}, \mathbf{1}^\top \boldsymbol{u} = 1\}$. As T decreases, the solution will gradually shift to the boundaries of the simplex to find the best mixture that best approximates the maximum. Therefore, T can also be viewed as a homotopy parameter that bridges a maximum entropy problem with uniform distribution $u_i = 1/M$ $(i = 1, \ldots, M)$ to a solution of the max-min problem (19).

This observation is also consistent with the soft-margin LPBoost. We know that soft-margin LPBoost often outperforms hard-margin LPBoost [9], [10]. In the primal, it is usually explained that the hinge loss of soft-margin is more appropriate for classification. The introduction of slack variables in the primal actually results in box constraints on the weight distribution in the dual. In other words, the ℓ_∞ norm of $\boldsymbol{u}$, $\|\boldsymbol{u}\|_\infty$, is capped. This capping mechanism is *harder* than the entropy regularization mechanism of AdaBoost$_{\ell 1}$. Nevertheless, both are beneficial on inseparable data. In [24], it is proven that soft-margin LPBoost actually maximizes the average of $1/D$ smallest margins.

Now, let us take a look at the cost function of AdaBoost and LPBoost in the primal. The log-sum-exp cost employed by AdaBoost can be viewed as a smooth approximation of the maximum function because of the following inequality:

$$\max_i \ a_i \le \log\left(\sum_{i=1}^{M} \exp a_i\right) \le \max_i a_i + \log M.$$

Therefore, LPBoost uses a hard maximum (or minimum) function, while AdaBoost uses a soft approximation of the maximum (minimum) function. We try to explain why AdaBoost's soft cost function is better than LPBoost's[2] hard cost function next.

2. Hereafter, we use LPBoost to denote hard-margin LPBoost unless otherwise specified.

3.3 AdaBoost$_{\ell 1}$ Controls the Margin Variance via Maximizing the Entropy of the Weights on the Training Examples

In AdaBoost training, there are two sets of weights: the weights of the weak classifiers $\boldsymbol{w}$ and the weights on the training examples $\boldsymbol{u}$. In the last section, we suppose that to limit $\boldsymbol{u}$ is beneficial for classification performance. By looking at the Karush-Kuhn-Tucker (KKT) conditions of the convex program that we have formulated, we are able to reveal the relationship between the two sets of weights. More precisely, we show how AdaBoost$_{\ell 1}$ (and AdaBoost[3]) controls the margin variance by optimizing the entropy of weights $\boldsymbol{u}$.

Recall that we have to introduce new equalities $z_i = -y_i H_i \boldsymbol{w}, \forall i$, in order to obtain the dual (10) (and (15)). Obviously, z_i is the negative margin of sample $\boldsymbol{x}_i$. Notice that the Lagrange multiplier $\boldsymbol{u}$ is associated with these equalities. Let $(\boldsymbol{w}^\star, \boldsymbol{z}^\star)$ and $(\boldsymbol{u}^\star, \boldsymbol{q}^\star, r^\star)$ be any primal and dual optimal points with zero duality gap. One of the KKT conditions tells us

$$\nabla_{\boldsymbol{z}} L(\boldsymbol{w}^\star, \boldsymbol{z}^\star, \boldsymbol{u}^\star, \boldsymbol{q}^\star, r^\star) = 0. \tag{23}$$

The Lagrangian $L(\cdot)$ is defined in (12). This equation follows

$$u_i^\star = \frac{\exp z_i^\star}{\sum_{i=1}^{M} \exp z_i^\star}, \quad \forall i = 1, \ldots M. \tag{24}$$

Equation (24) guarantees that $\boldsymbol{u}^\star$ is a probability distribution. Note that (24) is actually the same as the update rule used in AdaBoost. The optimal value[4] of the Lagrange dual problem (10), which we denote $\mathrm{Opt}^\star_{(10)}$, equals to the optimal value of the original problem (5) (and (11)) due to the strong duality; hence, $\mathrm{Opt}^\star_{(5)} = \mathrm{Opt}^\star_{(10)}$.

From (24), at optimality, we have

$$\begin{aligned} -z_i^\star &= -\log u_i^\star - \log\left(\sum_{i=1}^{M} \exp z_i^\star\right) \\ &= -\log u_i^\star - \mathrm{Opt}^\star_{(10)} \\ &= -\log u_i^\star - \mathrm{Opt}^\star_{(5)}, \quad \forall i = 1, \ldots M. \end{aligned} \tag{25}$$

This equation suggests that, after convergence, the margins' values are determined by the weights on the training examples $\boldsymbol{u}^\star$ and the cost function's value. From (25), the margin's variance is entirely determined by $\boldsymbol{u}^\star$:

$$\mathrm{var}\{-\boldsymbol{z}^\star\} = \mathrm{var}\{\log \boldsymbol{u}^\star\} + \mathrm{var}\{\mathrm{Opt}^\star_{(5)}\} = \mathrm{var}\{\log \boldsymbol{u}^\star\}. \tag{26}$$

We now understand the reason why capping $\boldsymbol{u}$ as LPBoost does or uniforming $\boldsymbol{u}$ as AdaBoost does can improve the classification performance. These two equations reveal the important role that the weight distribution $\boldsymbol{u}$ plays in AdaBoost. All that we knew previously is that the weights on the training examples measure how difficult an individual example can be to correctly classify. In fact, besides that, the weight distribution on the training

3. We believe that the only difference between AdaBoost$_{\ell 1}$ and AdaBoost is on the optimization method employed by each algorithm. We conjecture that some theoretical results on AdaBoost$_{\ell 1}$ derived in this paper may also apply to AdaBoost.

4. Hereafter, we use the symbol $\mathrm{Opt}^\star_{(\cdot)}$ to denote the optimal value of Problem $(\cdot)$.

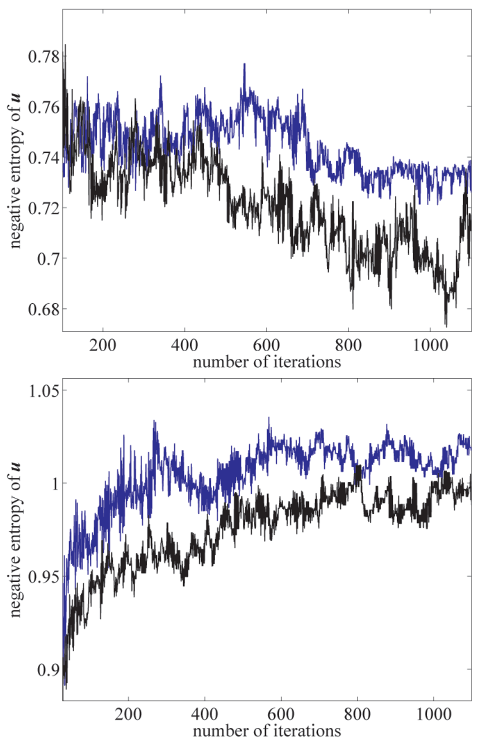

Fig. 1. Negative entropy of $\boldsymbol{u}$ produced by the standard AdaBoost and Arc-Gv at each iteration on data sets *breast-cancer* and *australian*, respectively. The negative entropy produced by AdaBoost (black) is consistently lower than the one by Arc-Gv (blue).

examples is also a *proxy* for minimizing the margin's distribution divergence. From the viewpoint of optimization, this is an interesting finding. In AdaBoost, one of the main purposes is to control the divergence of the margin distribution, which may not be easy to optimize *directly* because a margin can take a value out of the range $[0, 1]$, where entropy is not applicable. AdaBoost's cost function allows one to do so *implicitly* in the primal but *explicitly* in the dual. A future research topic is to apply this idea to other machine learning problems.

The connection between the dual variable $\boldsymbol{u}$ and margins tells us that AdaBoost often seems to optimize the minimum margin (or average margin? we will answer this question in the next section), but it also considers another quantity related to the variance of the margins. In the dual problem (15), minimizing the maximum edge on the weak classifiers contributes to maximizing the margin. At the same time, minimizing the negative entropy of weights on training examples contributes to controlling the margin's variance. We make this useful observation by examining the dual problem as well as the KKT optimality conditions. But, the exact statistics measures that AdaBoost optimizes remain unclear. The next section presents a complete answer to this question through analyzing AdaBoost's primal optimization problem.

We know that Arc-Gv chooses $\boldsymbol{w}$ in a different way from AdaBoost. Therefore, Arc-Gv optimizes a different cost function and does not minimize the negative entropy of $\boldsymbol{u}$ any more. We expect that AdaBoost will have a more uniform distribution of $\boldsymbol{u}$. We run AdaBoost and Arc-Gv with decision stumps on two data sets *breast-cancer* and *australian* (all data sets used in this paper are available at

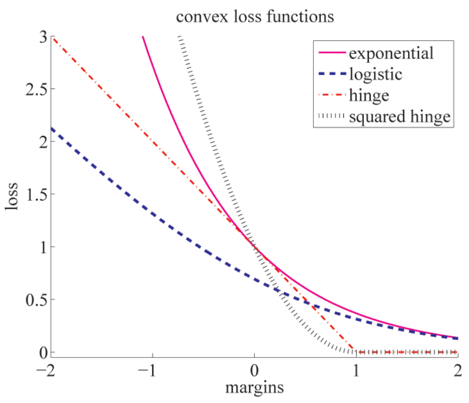

Fig. 2. Various loss functions used in classification. Exponential: $\exp -s$; logistic: $\log(1+\exp -s)$; hinge: $\max\{0, 1-s\}$; squared hinge: $(\max\{0, 1-s\})^2$. Here, $s = yF(\boldsymbol{x})$.

[25] unless otherwise specified). Fig. 1 displays the results. AdaBoost indeed has a small negative entropy of $\boldsymbol{u}$ in both experiments, which agrees with our prediction.

It is evident now that AdaBoost$_{\ell 1}$ controls the variance of margins by regularizing the Shannon entropy of the corresponding dual variable $\boldsymbol{u}$. For online learning algorithms, there are two main families of regularization strategies: entropy regularization and regularization using squared euclidean distance. A question that naturally arises here is: What happens if we use squared euclidean distance to replace the entropy in the dual of AdaBoost$_{\ell 1}$ (15)? In other words, *can we directly minimize the variance of the dual variable* $\boldsymbol{u}$ *to achieve the purpose of controlling the variance of margins*? We answer this question by having a look at the convex loss functions for classification.

Fig. 2 plots four popular convex loss functions. It is shown in [26] that, as the data size increases, practically all popular convex loss functions are Bayes-consistent, although convergence rates and other measures of consistency may vary. In the context of boosting, AdaBoost, LogitBoost, and soft-margin LPBoost use exponential loss, logistic loss, and hinge loss, respectively. Here we are interested in the squared hinge loss. LogitBoost will be discussed in the next section. As mentioned, in theory there is no particular reason to prefer hinge loss to squared hinge loss. Now, if squared hinge loss is adopted, the cost function of soft-margin LPBoost (21) becomes

$$\max_{\boldsymbol{w},\varrho,\boldsymbol{\xi}} \; \varrho - D\sum_{i=1}^{M}\xi_i^2,$$

and the constraints remain the same as in (21). Its dual is easily derived[5]

$$\begin{aligned} &\min_{r,\boldsymbol{u}} r + \frac{1}{4D}\sum_{i=1}^{M} u_i^2 \\ &\text{s.t.} \sum_{i=1}^{M} u_i y_i H_{ij} \le r \quad (\forall j = 1,\ldots,N), \\ &\quad \boldsymbol{u} \ge \boldsymbol{0}, \boldsymbol{1}^\top \boldsymbol{u} = 1. \end{aligned} \tag{27}$$

5. The primal constraint $\boldsymbol{\xi} \ge 0$ can be dropped because it is implicitly enforced.

TABLE 1
Dual Problems of Boosting Algorithms Are Entropy-Regularized LPBoost

algorithm	loss in primal	entropy regularized LPBoost in dual
AdaBoost	exponential loss	Shannon entropy
LogitBoost	logistic loss	binary relative entropy
soft-margin $\ell_p (p > 1)$ LPBoost	generalized hinge loss	Tsallis entropy

We can view the above optimization problem as variance-regularized LPBoost. In short, to minimize the variance of the dual variable $\boldsymbol{u}$ for controlling the margin's variance, one can simply replace soft-margin LPBoost's hinge loss with the squared hinge loss. Both the primal and dual problems are quadratic programs (QPs), and hence, can be efficiently solved using off-the-shelf QP solvers like MOSEK [27] and CPLEX [28].

Actually, we can generalize the hinge loss into

$$(\max\{0, 1 - yF(\boldsymbol{x})\})^p.$$

When $p \ge 1$, the loss is convex. $p = 1$ is the hinge loss and $p = 2$ is the squared hinge loss. If we use a generalized hinge loss ($p > 1$) for boosting, we end up with a regularized LPBoost which has the format:

$$\min_{r,\boldsymbol{u}} \; r + D^{1-q}\left(p^{1-q} - p^{-q}\right)\sum_{i=1}^{M} u_i^q, \tag{28}$$

subject to the same constraints as in (27). Here, p and q are dual to each other by $\frac{1}{p} + \frac{1}{q} = 1$. It is interesting that (28) can also be seen as entropy-regularized LPBoost; more precisely, Tsallis entropy [29] regularized LPBoost.

Definition 3.2 (Tsallis entropy). *Tsallis entropy is a generalization of the Shannon entropy, defined as*

$$S_q(\boldsymbol{u}) = \frac{1 - \sum_i u_i^q}{q - 1}, \quad (\boldsymbol{u} \ge 0, \mathbf{1}^\top \boldsymbol{u} = 1), \tag{29}$$

where q is a real number. In the limit, as $q \to 1$, we have

$$u_i^{q-1} = \exp((q-1)\log u_i) \simeq 1 + (q-1)\log u_i.$$

So, $S_1 = -\sum_i u_i \log u_i$, which is Shannon entropy.

Tsallis entropy [29] can also be viewed as a q-deformation of Shannon entropy because $S_q(\boldsymbol{u}) = -\sum_i u_i \log_q u_i$, where $\log_q(u) = \frac{u^{1-q}-1}{1-q}$ is the q-logarithm. Clearly, $\log_q(u) \to \log(u)$ when $q \to 1$.

In summary, we conclude that, although the primal problems of boosting with different loss functions seem dissimilar, their corresponding dual problems share the same formulation. Most of them can be interpreted as entropy-regularized LPBoost. Table 1 summarizes the result. The analysis of LogitBoost will be presented in the next section.

3.4 Lagrange Dual of LogitBoost

Thus far, we have discussed AdaBoost and its relation to LPBoost. In this section, we consider LogitBoost [6] from its dual.

Theorem 3.2. *The dual of LogitBoost is a binary relative entropy maximization problem, which writes*

$$\begin{aligned} \max_{r,\boldsymbol{u}} \; & \frac{r}{T} - \sum_{i=1}^{M}[(-u_i)\log(-u_i) + (1+u_i)\log(1+u_i)] \\ \text{s.t.} \; & \sum_{i=1}^{M} u_i y_i H_{ij} \ge r \quad (\forall j = 1, \ldots, N). \end{aligned} \tag{30}$$

We can also rewrite it into an equivalent form:

$$\begin{aligned} \min_{r,\boldsymbol{u}} \; & r + T\sum_{i=1}^{M}[u_i \log u_i + (1-u_i)\log(1-u_i)] \\ \text{s.t.} \; & \sum_{i=1}^{M} u_i y_i H_{ij} \le r \quad (\forall j = 1, \ldots, N). \end{aligned} \tag{31}$$

The proof follows the fact that the conjugate of the logistic loss function $\operatorname{logit}(x) = \log(1 + \exp -x)$ is

$$\operatorname{logit}^*(u) = \begin{cases} (-u)\log(-u) + (1+u)\log(1+u), & 0 \ge u \ge -1; \\ \infty, & \text{otherwise}, \end{cases}$$

with $0 \log 0 = 0$. $\operatorname{logit}^*(u)$ is a convex function in its domain. The corresponding primal is

$$\begin{aligned} \min_{\boldsymbol{w}} \; & \sum_{i=1}^{M} \operatorname{logit}(z_i) \\ \text{s.t.} \; & z_i = y_i H_i \boldsymbol{w}, \quad (\forall i = 1, \ldots, M), \\ & \boldsymbol{w} \ge \mathbf{0}, \mathbf{1}^\top \boldsymbol{w} = \frac{1}{T}. \end{aligned} \tag{32}$$

In (31), the dual variable $\boldsymbol{u}$ has a constraint $\mathbf{1} \ge \boldsymbol{u} \ge \mathbf{0}$ which is automatically enforced by the logarithmic function. Another difference of (31) from duals of AdaBoost, LPBoost, etc., is that $\boldsymbol{u}$ does not need to be normalized. In other words, in LogitBoost, the weight associated with each training sample is not necessarily a distribution. As in (24) for AdaBoost, we can also relate a dual optimal point $\boldsymbol{u}^\star$ and a primal optimal point $\boldsymbol{w}^\star$ (between (31) and (32)) by

$$u_i^\star = \frac{\exp - z_i^\star}{1 + \exp - z_i^\star}, \quad \forall i = 1, \ldots M. \tag{33}$$

So, the margin of $\boldsymbol{x}_i$ is solely determined by $u_i^\star$: $z_i^\star = \log \frac{1-u_i^\star}{u_i^\star}$, $\forall i$. For a positive margin ($\boldsymbol{x}_i$ is correctly classified), we must have $u_i^\star < 0.5$.

Similarly, we can also use CG to solve LogitBoost. As shown in Algorithm 2 in the case of AdaBoost, the only modification is to solve a different dual problem (here, we need to solve (31)).

3.5 AdaBoost$_{\ell 1}$ Approximately Maximizes the Average Margin and Minimizes the Margin Variance

Before we present our main result, a lemma is needed.

Lemma 3.1. *The margin of AdaBoost$_{\ell 1}$ and AdaBoost follows the Gaussian distribution. In general, the larger the number of* *weak classifiers, the more closely the margin follows the form of Gaussian under the assumption that selected weak classifiers are uncorrelated.*

Proof. The central limit theorem [30] states that the sum of a set of i.i.d. random variables x_i $(i = 1 \cdots N)$ is approximately distributed following a Gaussian distribution if the random variables have finite mean and variance.

Note that the central limit theorem applies when each variable x_i has an *arbitrary* probability distribution $\mathcal{Q}_i$ as long as the mean and variance of $\mathcal{Q}_i$ are finite.

As mentioned, the normalized margin of AdaBoost for the ith example is defined as

$$\varrho_i = \left(y_i \sum_{j=1}^{N} h_j(\boldsymbol{x}_i) w_j \right) \Big/ \mathbf{1}^\top \boldsymbol{w} = -z_i / \mathbf{1}^\top \boldsymbol{w}. \tag{34}$$

In the following analysis, we ignore the normalization term $\mathbf{1}^\top \boldsymbol{w}$ because it does not have any impact on the margin's distribution. Hence, the margin ϱ is the sum of N variables $\hat{w}_j$ with $\hat{w}_j = y_i h_j(\boldsymbol{x}_i) w_j$. It is easy to see that each $\hat{w}_j$ follows a discrete distribution with binary values either w_j or $-w_j$. Therefore, w_j must have finite mean and variance. Using the central limit theorem, we know that the distribution of ϱ_i is a Gaussian.

In the case of discrete variables (ϱ_i can be discrete), the assumption identical distributions can be substantially weakened [31]. The generalized central limit theorem essentially states that anything that can be thought of as being made up of the sum of many small independent variables is approximately normally distributed.

A condition of the central limit theorem is that the N variables must be independent. In the case of the number of weak hypotheses being finite, as the margin is expressed in (34), each $h_j(\cdot)$ is fixed beforehand and, assuming that all the training examples are randomly independently drawn, the variable ϱ_i would be independent too. When the set of weak hypotheses is infinite, it is well known that AdaBoost usually selects independent weak classifiers such that each weak classifier makes different errors on the training data set [15]. In this sense, w_j might be viewed as roughly independent from each other. More diverse weak classifiers will make the selected weak classifiers less dependent.[6] $\square$

Here, we give some empirical evidence for approximate Gaussianity. The normal (Gaussian) probability plot is used to visually assess whether the data follow a Gaussian distribution. If the data are Gaussian, the plot forms a straight line. Other distribution types introduce curvature in the plot. We run AdaBoost with decision stumps on the data set *australian*. Fig. 3 shows two plots of the margins with 50 and 1,100 weak classifiers, respectively. We see that with 50 weak classifiers, the margin distribution can be reasonably approximated by a Gaussian; with 1,100 classifiers, the distribution is very close to a Gaussian. The kurtosis of a 1D data provides a numerical evaluation of the Gaussianity. We know that the kurtosis of a Gaussian distribution is zero and almost all of the other distributions have nonzero kurtosis. In our experiment, the kurtosis is -0.056 for the case with 50 weak classifiers and -0.34 for 1,100 classifiers. Both are close to zero, which indicates that AdaBoost's margin distribution can be well approximated by Gaussian.

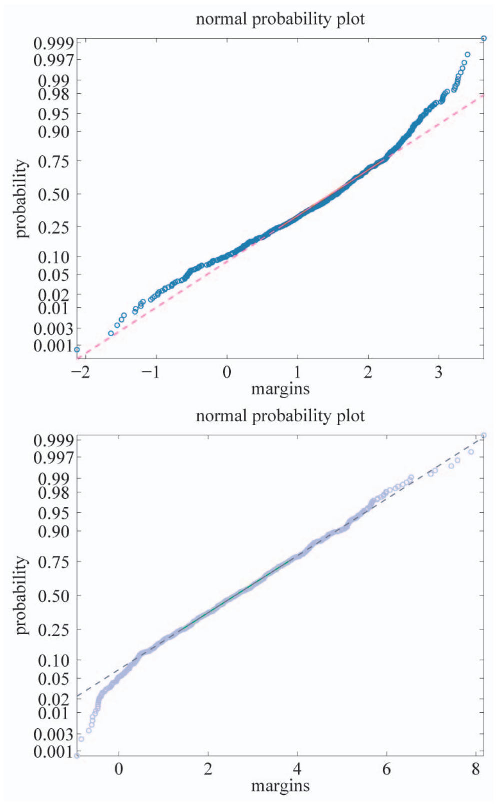

Fig. 3. Gaussianity test for the margin distribution with 50 and 1,100 weak classifiers, respectively. A Gaussian distribution will form a straight line. The data set used is *australian*.

Theorem 3.3. *AdaBoost*$_{\ell 1}$ *approximately maximizes the unnormalized average margin and at the same time minimizes the variance of the margin distribution under the assumption that the margin follows a Gaussian distribution.*

Proof. From (6) and (34), the cost function that AdaBoost$_{\ell 1}$ minimizes is

$$f_{\rm ab}(\boldsymbol{w}) = \log \left(\sum_{i=1}^{M} \exp - \frac{\varrho_i}{T} \right). \tag{35}$$

As proven in Lemma 3.1, ϱ_i follows a Gaussian

$$\mathcal{G}(\varrho; \bar{\varrho}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{(\varrho - \bar{\varrho})^2}{2\sigma^2},$$

with mean $\bar{\varrho}$, variance σ^2, and $\sum_{i=1}^{M} \varrho_i = 1$. We assume that the optimal value of the regularization parameter T is known a priori.

The Monte Carlo integration method can be used to compute a continuous integral

$$\int g(x) f(x) {\rm d}x \simeq \frac{1}{K} \sum_{k=1}^{K} f(x_k), \tag{36}$$

where $g(x)$ is a probability distribution such that $\int g(x){\rm d}x = 1$ and $f(x)$ is an arbitrary function. x_k $(k = 1 \cdots K)$ are randomly sampled from the distribution $g(x)$.

6. Nevertheless, this statement is not rigid.

The more samples are used, the more accurate the approximation is.

Equation (35) can be viewed as a discrete Monte Carlo approximation of the following integral (we omit a constant term $\log M$, which is irrelevant to the analysis):

$$\begin{aligned}\hat{f_{\rm ab}}(\boldsymbol{w}) &= \log \int_{\varrho_1}^{\varrho_2} \mathcal{G}(\varrho; \bar{\varrho}, \sigma) \exp\Big(-\frac{\varrho}{T}\Big) \mathrm{d}\varrho \\ &= \log \int_{\varrho_1}^{\varrho_2} \frac{1}{\sqrt{2\pi}\sigma} \exp\Bigg(-\frac{(\varrho - \bar{\varrho})^2}{2\sigma^2} - \frac{\varrho}{T}\Bigg) \mathrm{d}\varrho \\ &= \log\Bigg[\frac{1}{2}\exp\Big(-\frac{\bar{\varrho}}{T} + \frac{\sigma^2}{2T^2}\Big) \mathrm{erf}\Big(\frac{\varrho - \bar{\varrho}}{\sqrt{2}\sigma} + \frac{\sigma}{\sqrt{2}T}\Big)\Bigg|_{\varrho_1}^{\varrho_2}\Bigg] \\ &= -\log 2 - \frac{\bar{\varrho}}{T} + \frac{\sigma^2}{2T^2} + \log\Bigg[\mathrm{erf}\Big(\frac{\varrho - \bar{\varrho}}{\sqrt{2}\sigma} + \frac{\sigma}{\sqrt{2}T}\Big)\Bigg|_{\varrho_1}^{\varrho_2}\Bigg],\end{aligned} \tag{37}$$

where $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp -s^2 \mathrm{d}s$ is the Gauss error function. The integral range is $[\varrho_1, \varrho_2]$. With no explicit knowledge about the integration range, we may roughly calculate the integral from $-\infty$ to $+\infty$. Then, the last term in (37) is $\log 2$ and the result is analytical and simple:

$$\hat{f_{\rm ab}}(\boldsymbol{w}) = -\frac{\bar{\varrho}}{T} + \frac{1}{2}\frac{\sigma^2}{T^2}. \tag{38}$$

This is a reasonable approximation because Gaussian distributions drop off quickly (Gaussian is not considered a heavy-tailed distribution). Also, this approximation implies that we are considering the case where the number of samples goes to $+\infty$.

Consequently, AdaBoost approximately maximizes the cost function:

$$-\hat{f_{\rm ab}}(\boldsymbol{w}) = \frac{\bar{\varrho}}{T} - \frac{1}{2}\frac{\sigma^2}{T^2}. \tag{39}$$

This cost function has a clear and simple interpretation: The first term $\bar{\varrho}/T$ is the unnormalized average margin and the second term σ^2/T^2 is the unnormalized margin variance. So, AdaBoost maximizes the unnormalized average margin and also takes minimizing the unnormalized margin variance into account. This way a better *margin distribution* can be obtained. □

Note that Theorem 3.3 depends on Lemma 3.1, which does not necessarily hold in practice.

Theorem 3.3 is an important result in the sense that it tries to contribute to the open question of why AdaBoost works so well. Much previous work intends to believe that AdaBoost maximizes the minimum margin. We have theoretically shown that AdaBoost$_{\ell 1}$ optimizes the entire margin distribution by maximizing the mean and minimizing the variance of the margin distribution.

We notice that when $T \to 0$, Theorem 3.3 becomes invalid because the Monte Carlo integration cannot approximate the cost function of AdaBoost (35) well. In practice, T cannot approach zero arbitrarily in AdaBoost.

One may suspect that Theorem 3.3 contradicts the observation of similarity between LPBoost and AdaBoost, as shown in Section 3.2. LPBoost maximizes the minimum margin and the dual of AdaBoost is merely an entropy-regularized LPBoost. At first glance, the dual variable r in (15), (20), and (22) should have the same meaning, i.e., maximum edge, which in turn corresponds to the minimum margin in the primal. Why average margin? To answer this question, let us again take a look at the optimality conditions. Let us denote the optimal values of (15) $r^\star$ and $\boldsymbol{u}^\star$. At convergence, we have

$$\frac{1}{T}\Bigg(-r^\star + T\sum_{i=1}^M u_i^\star \log u_i^\star\Bigg) = \mathrm{Opt}^\star_{(10)} = \mathrm{Opt}^\star_{(5)} = \mathrm{Opt}^\star_{(6)}.$$

Hence, we have

$$r^\star = T\sum_{i=1}^M u^\star \log u^\star - T\log\Bigg(\sum_{i=1}^M \exp -\frac{\varrho_i^\star}{T}\Bigg),$$

where $\varrho_i^\star$ is the normalized margin for $\boldsymbol{x}_i$. Clearly, this is very different from the optimality conditions of LPBoost, which shows that $r^\star$ is the minimum margin. Only when $T \to 0$ does the above relationship reduce to $r^\star = \min_i\{\varrho_i^\star\}$—the same as the case of LPBoost.

3.6 AdaBoost-QP: Direct Optimization of the Margin Mean and Variance Using Quadratic Programming

The above analysis suggests that we can directly optimize the cost function (39). In this section, we show that (39) is a convex programming (more precisely, QP) problem in the variable $\boldsymbol{w}$ if we know all of the base classifiers, and hence, it can be efficiently solved. Next, we formulate the QP problem in detail. We call the proposed algorithm AdaBoost-QP.[7]

In kernel methods like SVMs, the original space $\mathcal{X}$ is mapped to a feature space $\mathcal{F}$. The mapping function $\Phi(\cdot)$ is not explicitly computable. It is shown in [14] that, in boosting, one can think of the mapping function $\Phi(\cdot)$ being *explicitly* known:

$$\Phi(\boldsymbol{x}) : \boldsymbol{x} \mapsto [h_1(\boldsymbol{x}), \ldots, h_N(\boldsymbol{x})]^\top, \tag{40}$$

using the weak classifiers. Therefore, any weak classifier set $\mathcal{H}$ spans a feature space $\mathcal{F}$. We can design an algorithm that optimizes (39):

$$\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^\top A\boldsymbol{w} - T\boldsymbol{b}^\top\boldsymbol{w}, \quad \text{s.t. } \boldsymbol{w} \geq 0, \boldsymbol{1}^\top\boldsymbol{w} = 1, \tag{41}$$

where[8]

$$\begin{aligned}\boldsymbol{b} &= \frac{1}{M}\sum_{i=1}^M y_i H_i^\top = \frac{1}{M}\sum_{i=1}^M y_i\Phi(\boldsymbol{x}_i) \text{ and} \\ A &= \frac{1}{M}\sum_{i=1}^M \big(y_i H_i^\top - \boldsymbol{b}\big)\big(y_i H_i^\top - \boldsymbol{b}\big)^\top \\ &= \frac{1}{M}\sum_{i=1}^M (y_i\Phi(\boldsymbol{x}_i) - \boldsymbol{b})(y_i\Phi(\boldsymbol{x}_i) - \boldsymbol{b})^\top.\end{aligned}$$

Clearly, A must be positive semidefinite and this is a standard convex QP problem. The nonnegativeness constraint $\boldsymbol{w} \geq 0$ introduces sparsity, as in SVMs. Without this

7. In [32], the authors proposed QP$_{\rm reg}$-AdaBoost for soft-margin AdaBoost learning, which is inspired by SVMs. Their QP$_{\rm reg}$-AdaBoost is completely different from ours.

8. To show the connection of AdaBoost-QP with kernel methods, we have written $\Phi(\boldsymbol{x}_i) = H_i^\top$.

constraint, the above QP can be analytically solved using eigenvalue decomposition—the largest eigenvector is the solution. Usually, all entries of this solution would be active (nonzero values).

In the kernel space,

$$\boldsymbol{b}^\top \boldsymbol{w} = \frac{1}{M}\left(\sum_{y_i=1} \Phi(\boldsymbol{x}_i) - \sum_{y_i=-1} \Phi(\boldsymbol{x}_i)\right)^\top \boldsymbol{w}$$

can be viewed as the projected ℓ_1 norm distance between two classes because, typically, this value is positive, assuming that each class has the same number of examples. The matrix A *approximately* plays a role as the total scatter matrix in kernel linear discriminant analysis (LDA). Note that AdaBoost does not take the number of examples in each class into consideration when it models the problem. In contrast, LDA (kernel LDA) takes training example number into consideration. This may explain why an LDA postprocessing on AdaBoost gives a better classification performance on face detection [33], which is a highly imbalanced classification problem. This observation of similarity between AdaBoost and kernel LDA may inspire new algorithms. We are also interested in developing a CG-based algorithm for iteratively generating weak classifiers.

3.7 AdaBoost-CG: Totally Corrective AdaBoost Using Column Generation

The number of possible weak classifiers may be infinitely large. In this case, it may be infeasible to solve the optimization *exactly*. AdaBoost works on the primal problem directly by switching between the estimating weak classifiers and computing optimal weights in a coordinate descent way. There is another method for working out of this problem by using an optimization technique termed CG [10], [34]. CG mainly works on the dual problem. The basic concept of the CG method is to add one constraint at a time to the dual problem until an optimal solution is identified. More columns need to be generated and added to the problem to achieve optimality. In the primal space, the CG method solves the problem on a subset of variables which corresponds to a subset of constraints in the dual. When a column is not included in the primal, the corresponding constraint does not appear in the dual. That is to say, a relaxed version of the dual problem is solved. If a constraint absent from the dual problem is violated by the solution to the restricted problem, this constraint needs to be included in the dual problem to further restrict its feasible region. In our case, instead of solving the optimization of AdaBoost directly, one computes the most violated constraint in (15) iteratively for the current solution and adds this constraint to the optimization problem. In theory, any column that violates dual feasibility can be added. To do so, we need to solve the following subproblem:

$$h'(\cdot) = \arg\max_{h(\cdot)} \sum_{i=1}^{M} u_i y_i h(\boldsymbol{x}_i). \tag{42}$$

This strategy is exactly the same as the one that stagewise AdaBoost and LPBoost use for generating the best weak classifier, that is, to find the weak classifier that produces minimum weighted training error. Putting all of the above analysis together, we summarize our AdaBoost-CG in Algorithm 2.

Algorithm 2 AdaBoost-CG.

```
Input: Training set $(\boldsymbol{x}_i, y_i), i = 1 \cdots M$; termination
       threshold $\varepsilon > 0$; regularization parameter $T$;
       (optional) maximum iteration $N_{\max}$.
1 Initialization:
    1) $N = 0$ (no weak classifiers selected);
    2) $\boldsymbol{w} = \mathbf{0}$ (all primal coefficients are zeros);
    3) $u_i = \frac{1}{M}$, $i = 1 \cdots M$ (uniform dual weights).
2 while true do
    1) Find a new base $h'(\cdot)$ by solving Problem (42);
    2) Check for optimal solution:
       if $\sum_{i=1}^{M} u_i y_i h'(\boldsymbol{x}_i) < r + \varepsilon$, then break (problem
       solved);
    3) Add $h'(\cdot)$ to the restricted master problem, which
       corresponds to a new constraint in the dual;
    4) Solve the dual to obtain updated $r$ and $u_i$
       ($i = 1, \cdots, M$): for AdaBoost, the dual is (15);
    5) $N = N + 1$ (weak classifier count);
    6) (optional) if $N \geq N_{\max}$, then break (maximum
       iteration reached).
Output:
    1) Calculate the primal variable $\boldsymbol{w}$ from the optimality
       conditions and the last solved dual problem;
    2) The learned classifier $F(\boldsymbol{x}) = \sum_{j=1}^{N} w_j h_j(\boldsymbol{x})$.
```

The CG optimization (Algorithm 2) is so general that it can be applied to all of the boosting algorithms considered in this paper by solving the corresponding dual. The convergence follows general CG algorithms, which is easy to establish. When a new $h'(\cdot)$ that violates dual feasibility is added, the new optimal value of the dual problem (maximization) would decrease. Accordingly, the optimal value of its primal problem decreases too because they have the same optimal value due to zero duality gap. Moreover, the primal cost function is convex; therefore, eventually, it converges to the global minimum. A comment on the last step of Algorithm 2 is that we can get the value of $\boldsymbol{w}$ easily. Primal-dual interior point (PD-IP) methods work on the primal and dual problems simultaneously, and therefore, both primal and dual variables are available after convergence. We use MOSEK [27], which implements PD-IP methods. The primal variable $\boldsymbol{w}$ is obtained *for free* when solving the dual problem (15).

The dual subproblem we need to solve has one constraint added at each iteration. Hence, after many iterations, solving the dual problem could become intractable in theory. In practice, AdaBoost-CG converges quickly on our tested data sets. As pointed out in [35], usually only a small number of the added constraints are active and those inactive ones may be removed. This strategy prevents the dual problem from growing too large.

AdaBoost-CG is totally corrective in the sense that the coefficients of all weak classifiers are updated at each iteration. In [36], an additional correction procedure is inserted to AdaBoost's weak classifier selection cycle for achieving totally correction. The inserted correction procedure aggressively reduces the *upper bound* of the training error. Like AdaBoost, it works in the primal. In contrast, our algorithm optimizes the regularized loss function directly and mainly works in the dual space. In [37], a totally corrective boosting is proposed by optimizing the entropy,

TABLE 2
Test Results of AdaBoost (AB) and AdaBoost-QP (QP)

dataset	algorithm	test error	minimum margin	average margin
australian	AB	0.153 ± 0.034	$\mathbf{-0.012 \pm 0.005}$	0.082 ± 0.006
	QP	$\mathbf{0.13 \pm 0.038}$	-0.227 ± 0.081	$\mathbf{0.18 \pm 0.052}$
b-cancer	AB	0.041 ± 0.013	$\mathbf{0.048 \pm 0.009}$	0.209 ± 0.02
	QP	$\mathbf{0.03 \pm 0.012}$	-0.424 ± 0.250	$\mathbf{0.523 \pm 0.237}$
diabetes	AB	0.270 ± 0.043	$\mathbf{-0.038 \pm 0.007}$	0.055 ± 0.005
	QP	$\mathbf{0.262 \pm 0.047}$	-0.107 ± 0.060	$\mathbf{0.075 \pm 0.031}$
fourclass	AB	$\mathbf{0.088 \pm 0.032}$	$\mathbf{-0.045 \pm 0.012}$	0.084 ± 0.009
	QP	0.095 ± 0.028	-0.211 ± 0.059	$\mathbf{0.128 \pm 0.027}$
g-numer	AB	0.283 ± 0.033	$\mathbf{-0.079 \pm 0.017}$	0.042 ± 0.006
	QP	$\mathbf{0.249 \pm 0.033}$	-0.151 ± 0.058	$\mathbf{0.061 \pm 0.020}$
heart	AB	0.210 ± 0.032	$\mathbf{0.02 \pm 0.008}$	0.104 ± 0.013
	QP	$\mathbf{0.190 \pm 0.058}$	-0.117 ± 0.066	$\mathbf{0.146 \pm 0.059}$
ionosphere	AB	$\mathbf{0.121 \pm 0.044}$	$\mathbf{0.101 \pm 0.010}$	0.165 ± 0.012
	QP	0.139 ± 0.055	-0.035 ± 0.112	$\mathbf{0.184 \pm 0.063}$
liver	AB	0.321 ± 0.040	$\mathbf{-0.012 \pm 0.007}$	0.055 ± 0.005
	QP	$\mathbf{0.314 \pm 0.060}$	-0.107 ± 0.044	$\mathbf{0.079 \pm 0.021}$
mushrooms	AB	$\mathbf{0 \pm 0}$	$\mathbf{0.102 \pm 0.001}$	0.181 ± 0.001
	QP	0.005 ± 0.002	-0.134 ± 0.086	$\mathbf{0.221 \pm 0.084}$
sonar	AB	$\mathbf{0.145 \pm 0.046}$	$\mathbf{0.156 \pm 0.008}$	0.202 ± 0.013
	QP	0.171 ± 0.048	0.056 ± 0.066	$\mathbf{0.220 \pm 0.045}$
splice	AB	0.129 ± 0.025	$\mathbf{-0.009 \pm 0.008}$	0.117 ± 0.009
	QP	$\mathbf{0.106 \pm 0.029}$	-0.21 ± 0.037	$\mathbf{0.189 \pm 0.02}$
svmguide1	AB	$\mathbf{0.035 \pm 0.009}$	$\mathbf{-0.010 \pm 0.008}$	0.157 ± 0.016
	QP	0.040 ± 0.009	-0.439 ± 0.183	$\mathbf{0.445 \pm 0.155}$
svmguide3	AB	0.172 ± 0.023	$\mathbf{-0.011 \pm 0.009}$	0.052 ± 0.005
	QP	$\mathbf{0.167 \pm 0.022}$	-0.113 ± 0.084	$\mathbf{0.085 \pm 0.038}$
w1a	AB	0.041 ± 0.014	$\mathbf{-0.048 \pm 0.010}$	0.084 ± 0.005
	QP	$\mathbf{0.029 \pm 0.009}$	-0.624 ± 0.38	$\mathbf{0.577 \pm 0.363}$

All tests are run 10 times. The mean and standard deviation are reported. AdaBoost-QP outperforms AdaBoost on nine data sets.

which is inspired by [18]. As discussed, no explicit primal-dual connection is established. That is why an LPBoost procedure is needed over the obtained weak classifiers in order to calculate the primal variable w. In this sense, [37] is also similar to the work of [32].

The following diagram summarizes the relationships that we have derived on the boosting algorithms that we have considered:

$$\begin{tikzcd}
\text{AdaBoost}_{\ell 1}\text{ primal} \arrow[r, "\text{AdaBoost}-\text{CG}", "\text{Lagrange duality}"'] \arrow[d, "\text{Theorem 3.3}"'] & \text{AdaBoost}_{\ell 1}\text{dual} \\
\text{AdaBoost-QP} & \text{LPBoost dual} \arrow[u, "\text{entropy}", "\text{regularization}"']
\end{tikzcd}$$

4 Experiments

In this section, we provide experimental results to verify the presented theory. We have mainly used decision stumps as weak classifiers due to its simplicity and well-controlled complexity. In some cases, we have also used one of the simplest linear classifiers, LDA, as weak classifiers. To avoid the singularity problem when solving LDA, we add a scaled identity matrix $10^{-4}\mathbf{I}$ to the within-class matrix. For the CG optimization framework, we have confined ourself to AdaBoost-CG although the technique is general and applicable for optimizing other boosting algorithms.

4.1 AdaBoost-QP

We compare AdaBoost-QP against AdaBoost. We have used 14 benchmark data sets [25]. Except for *mushrooms*, *svmguide1*, *svmguide3*, and *w1a*, all of the other data sets have been scaled to $[-1, 1]$. We randomly split each data set into training, cross validation, and test sets at a ratio of $70 : 15 : 15$.

The stopping criterion of AdaBoost is determined by cross validation on $\{600, 800, 1{,}000, 1{,}200, 1{,}500\}$ rounds of boosting. For AdaBoost-QP, the best value for the parameter T is chosen from $\{\frac{1}{10}, \frac{1}{20}, \frac{1}{30}, \frac{1}{40}, \frac{1}{50}, \frac{1}{100}, \frac{1}{200}, \frac{1}{500}\}$ by cross validation. In this experiment, decision stumps are used as the weak classifier such that the complexity of the base classifiers is well controlled.

AdaBoost-QP must access all weak classifiers a priori. Here, we run AdaBoost-QP on the 1,500 weak classifiers generated by AdaBoost. Clearly, this number of hypotheses may not be optimal. Theoretically, the larger the size of the weak classifier pool is, the better results AdaBoost-QP may produce. Table 2 reports the results. The experiments show that among these 14 data sets, AdaBoost-QP outperforms AdaBoost on nine data sets in terms of generalization error. On *mushrooms*, both perform very well. On the other four data sets, AdaBoost is better.

We have also computed the normalized version of the cost function value of (39). In most cases, AdaBoost-QP has a larger value. This is not surprising since AdaBoost-QP directly maximizes (39), while AdaBoost approximately maximizes it. Furthermore, the normalized loss function value is close to the normalized average margin because the margin variances for most data sets are very small compared with their means.

We also compute the largest minimum margin and average margin on each data set. On all of the data sets, AdaBoost has a larger minimum margin than AdaBoost-QP. This confirms that the minimum margin is not crucial for the

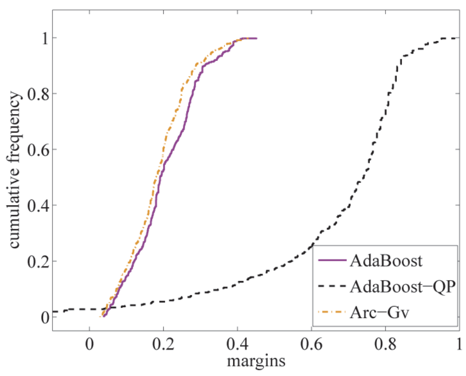

Fig. 4. Cumulative margins for AdaBoost, AdaBoost-QP, and Arc-Gv for the breast cancer data set using decision stumps. Overall, the margin distribution of AdaBoost-QP is the best and it has the smallest test error. AdaBoost and Arc-Gv run 600 rounds of boosting. Test errors for AdaBoost, AdaBoost-QP, and Arc-Gv are 0.029, 0.027, and 0.058, respectively.

generalization error. On the other hand, the average margin produced by AdaBoost-QP, which is the first term of the cost function (39), is consistently larger than the one obtained by AdaBoost. Indirectly, we have shown that a better overall margin distribution is more important than the largest minimum margin. In Fig. 4, we plot cumulative margins for AdaBoost-QP and AdaBoost on the breast cancer data set with decision stumps. We can see that while Arc-Gv has a largest minimum margin, it has the worst margins distribution overall. If we examine the average margins, AdaBoost-QP is the largest, AdaBoost is second, and Arc-Gv is least. Clearly, a better overall distribution does lead to a smaller generalization error. When Arc-Gv and AdaBoost run for more rounds, their margin distributions seem to converge. That is what we see in Fig. 4. These results agree well with our theoretical analysis (Theorem 3.3). Another observation from this experiment is that, to achieve the same performance, AdaBoost-QP tends to use fewer weak classifiers than AdaBoost does.

We have also tested AdaBoost-QP on full sets of weak classifiers because the number of possible decision stumps is finite (less than (number of features -1) $\times$ (number of examples)). Table 3 reports the test error of AdaBoost-QP on some small data sets. As expected, in most cases, the test error is slightly better than the results using 1,500 decision stumps in Table 2 and no significant difference is observed. This verifies the capability of AdaBoost-QP for selecting and combining relevant weak classifiers.

4.2 AdaBoost-CG

We run AdaBoost and AdaBoost-CG with decision stumps on the data sets of [25]. Seventy percent of examples are used for training, 15 percent are used for test, and the other 15 percent are not used because we do not do cross validation here. The convergence threshold for AdaBoost-CG (ε in Algorithm 2) is set to 10^{-5}. Another important parameter to tune is the regularization parameter T. For the first experiment, we have set it to $1/\mathbf{1}^\top \boldsymbol{w}$, where $\boldsymbol{w}$ is obtained by running AdaBoost on the same data for $1{,}000$ iterations. Also, for fair comparison, we have deliberately forced AdaBoost-CG to run $1{,}000$ iterations even if the stopping criterion is met. Both test and training results for AdaBoost and AdaBoost-CG are reported in Table 4 for a maximum number of iterations of 100, 500, and 1,000.

As expected, in terms of test error, no algorithm statistically outperforms the other one since they optimize the same cost function. As we can see, AdaBoost does slightly better on six data sets. AdaBoost-CG outperforms AdaBoost on seven data sets and, on *svmguide1*, both algorithms perform almost identically. Therefore, empirically, we conclude that in terms of generalization capability, AdaBoost-CG is the same as the standard AdaBoost.

However, in terms of training error and convergence speed of the training procedure, there is significant difference between these two algorithms. Looking at the right part of Table 4, we see that the training error of AdaBoost-CG is consistently better or no worse than AdaBoost on *all* tested data sets. We have the following conclusions:

- The convergence speed of AdaBoost-CG is faster than AdaBoost and, in many cases, better training error can be achieved. This is because AdaBoost's coordinate descent nature is slow, while AdaBoost-CG is *totally corrective*.[9] This also means that with AdaBoost-CG, we can use fewer weak classifiers to build a good strong classifier. This is desirable for real-time applications like face detection [38], in which the testing speed is critical.
- Our experiments confirm that a smaller training error does not necessarily lead to a smaller test error. This has been studied extensively in statistical learning theory. It is observed that AdaBoost sometimes suffers from overfitting and minimizing the exponential cost function of the margins does not solely determine test error.

In the second experiment, we run cross validation to select the best value for the regularization parameter T, the same as in Section 4.1. Table 5 reports the test errors on a subset of the data sets. Slightly better results are obtained compared with the results in Table 4, which uses T determined by AdaBoost.

We also use LDA as weak classifiers to compare the classification performance of AdaBoost and AdaBoost-CG. The parameter T of AdaBoost-CG is determined by cross validation from $\{\frac{1}{2},\frac{1}{5},\frac{1}{8},\frac{1}{10},\frac{1}{12},\frac{1}{15},\frac{1}{20},\frac{1}{30},\frac{1}{40},\frac{1}{50},\frac{1}{70},\frac{1}{90},\frac{1}{100},\frac{1}{120},\frac{1}{150}\}$. For AdaBoost, the smallest test error from 100, 500, and 1,000 runs is reported. We show the results in Table 6. As we can see, the test error is slightly better than with decision stumps for both AdaBoost and AdaBoost-CG. Again, AdaBoost and AdaBoost-CG's performances are very similar.

In order to show that statistically there are no differences between AdaBoost-CG and AdaBoost, the McNemar test [39] with the significance level of 0.05 is conducted. McNemar's test is based on a χ^2 test [39]. If the quantity of the χ^2 test is not greater than $\chi^2_{1,0.95} = 3.841459$, we can think that the two tested classifiers have *no statistical difference* in terms of classification capability. On the eight data sets with decision stumps and LDA (Tables 5 and 6), in all cases (five runs per data set) the results of the χ^2 test are

9. Like LPBoost, at each iteration, AdaBoost-CG updates the previous weak classifier weights $\boldsymbol{w}$.

TABLE 3
Test Results of AdaBoost-QP on Full Sets of Decision Stumps

dataset	**australian**	**b-cancer**	**fourclass**	**g-numer**	**heart**	**liver**	**mushroom**	**splice**
test error	0.131 ± 0.041	0.03 ± 0.011	0.091 ± 0.02	0.243 ± 0.026	0.188 ± 0.058	0.319 ± 0.05	0.003 ± 0.001	0.097 ± 0.02

All tests are run 10 times.

TABLE 4
Test and Training Errors of AdaBoost (AB) and AdaBoost-CG (CG)

dataset	algorithm	test error 100	test error 500	test error 1000	train error 100	train error 500	train error 1000
australian	AB	$\mathbf{0.146 \pm 0.028}$	$\mathbf{0.165 \pm 0.018}$	$\mathbf{0.163 \pm 0.021}$	0.091 ± 0.013	0.039 ± 0.011	0.013 ± 0.009
	CG	0.177 ± 0.025	0.167 ± 0.023	0.167 ± 0.023	$\mathbf{0.013 \pm 0.008}$	$\mathbf{0.011 \pm 0.007}$	$\mathbf{0.011 \pm 0.007}$
b-cancer	AB	$\mathbf{0.041 \pm 0.026}$	$\mathbf{0.045 \pm 0.030}$	$\mathbf{0.047 \pm 0.032}$	0.008 ± 0.006	0 ± 0	0 ± 0
	CG	0.049 ± 0.033	0.049 ± 0.033	0.049 ± 0.033	$\mathbf{0 \pm 0}$	0 ± 0	0 ± 0
diabetes	AB	$\mathbf{0.254 \pm 0.024}$	0.263 ± 0.028	0.257 ± 0.041	0.171 ± 0.012	0.120 ± 0.007	0.082 ± 0.006
	CG	0.270 ± 0.047	$\mathbf{0.254 \pm 0.026}$	$\mathbf{0.254 \pm 0.026}$	$\mathbf{0.083 \pm 0.008}$	$\mathbf{0.070 \pm 0.007}$	$\mathbf{0.070 \pm 0.007}$
fourclass	AB	0.106 ± 0.047	0.097 ± 0.034	0.091 ± 0.031	0.072 ± 0.023	0.053 ± 0.017	0.046 ± 0.017
	CG	$\mathbf{0.082 \pm 0.031}$	$\mathbf{0.082 \pm 0.031}$	$\mathbf{0.082 \pm 0.031}$	$\mathbf{0.042 \pm 0.015}$	$\mathbf{0.042 \pm 0.015}$	$\mathbf{0.042 \pm 0.015}$
g-numer	AB	0.279 ± 0.043	0.288 ± 0.048	0.297 ± 0.051	0.206 ± 0.047	0.167 ± 0.072	0.155 ± 0.082
	CG	$\mathbf{0.269 \pm 0.040}$	$\mathbf{0.262 \pm 0.045}$	$\mathbf{0.262 \pm 0.045}$	$\mathbf{0.142 \pm 0.077}$	$\mathbf{0.142 \pm 0.077}$	$\mathbf{0.142 \pm 0.077}$
heart	AB	0.175 ± 0.073	0.175 ± 0.088	0.165 ± 0.076	0.049 ± 0.022	0 ± 0	0 ± 0
	CG	$\mathbf{0.165 \pm 0.072}$	$\mathbf{0.165 \pm 0.072}$	$\mathbf{0.165 \pm 0.072}$	$\mathbf{0 \pm 0}$	0 ± 0	0 ± 0
ionosphere	AB	$\mathbf{0.092 \pm 0.016}$	$\mathbf{0.104 \pm 0.017}$	$\mathbf{0.100 \pm 0.016}$	0 ± 0	0 ± 0	0 ± 0
	CG	0.131 ± 0.034	0.131 ± 0.034	0.131 ± 0.034	0 ± 0	0 ± 0	0 ± 0
liver	AB	0.288 ± 0.101	$\mathbf{0.265 \pm 0.081}$	$\mathbf{0.281 \pm 0.062}$	0.144 ± 0.018	0.063 ± 0.015	0.020 ± 0.015
	CG	$\mathbf{0.288 \pm 0.084}$	0.288 ± 0.084	0.288 ± 0.084	$\mathbf{0.017 \pm 0.012}$	$\mathbf{0.017 \pm 0.011}$	$\mathbf{0.017 \pm 0.011}$
mushrooms	AB	0 ± 0.001	0 ± 0.001	0 ± 0	0 ± 0	0 ± 0	0 ± 0
	CG	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$	0 ± 0	0 ± 0	0 ± 0	0 ± 0
sonar	AB	$\mathbf{0.206 \pm 0.087}$	$\mathbf{0.213 \pm 0.071}$	$\mathbf{0.206 \pm 0.059}$	0 ± 0	0 ± 0	0 ± 0
	CG	0.232 ± 0.053	0.245 ± 0.078	0.245 ± 0.078	0 ± 0	0 ± 0	0 ± 0
splice	AB	$\mathbf{0.129 \pm 0.011}$	$\mathbf{0.143 \pm 0.026}$	$\mathbf{0.143 \pm 0.020}$	0.053 ± 0.003	0.008 ± 0.006	0.001 ± 0.001
	CG	0.161 ± 0.033	0.151 ± 0.023	0.151 ± 0.023	$\mathbf{0.002 \pm 0.002}$	$\mathbf{0.001 \pm 0.002}$	$\mathbf{0.001 \pm 0.002}$
svmguide1	AB	$\mathbf{0.036 \pm 0.012}$	$\mathbf{0.034 \pm 0.008}$	0.037 ± 0.007	0.022 ± 0.002	0.009 ± 0.002	0.002 ± 0.001
	CG	0.037 ± 0.007	0.037 ± 0.007	0.037 ± 0.007	$\mathbf{0.001 \pm 0.001}$	$\mathbf{0 \pm 0.001}$	$\mathbf{0 \pm 0.001}$
svmguide3	AB	0.184 ± 0.037	0.183 ± 0.044	0.182 ± 0.031	0.112 ± 0.009	0.037 ± 0.004	0.009 ± 0.003
	CG	$\mathbf{0.184 \pm 0.026}$	$\mathbf{0.171 \pm 0.023}$	$\mathbf{0.171 \pm 0.023}$	$\mathbf{0.033 \pm 0.012}$	$\mathbf{0.023 \pm 0.016}$	$\mathbf{0.023 \pm 0.016}$
w1a	AB	0.051 ± 0.009	0.038 ± 0.005	0.036 ± 0.004	0.045 ± 0.008	0.028 ± 0.005	0.025 ± 0.005
	CG	$\mathbf{0.018 \pm 0.001}$	$\mathbf{0.018 \pm 0.001}$	$\mathbf{0.018 \pm 0.001}$	$\mathbf{0.010 \pm 0.004}$	$\mathbf{0.010 \pm 0.004}$	$\mathbf{0.010 \pm 0.004}$

All tests are run five times. The mean and standard deviation are reported. Weak classifiers are decision stumps.

TABLE 5
Test Error of AdaBoost-CG with Decision Stumps, Using Cross Validation to Select the Optimal T

dataset	**australian**	**b-cancer**	**diabetes**	**fourclass**	**heart**	**ionosphere**	**sonar**	**splice**
test error	0.146 ± 0.027	0.033 ± 0.033	0.266 ± 0.036	0.086 ± 0.027	0.17 ± 0.082	0.115 ± 0.024	0.2 ± 0.035	0.135 ± 0.015

All tests are run five times.

TABLE 6
Test Error of AdaBoost (AB) and AdaBoost-CG (CG) with LDA as Weak Classifiers, Using Cross Validation to Select the Optimal T

dataset	**australian**	**b-cancer**	**diabetes**	**fourclass**	**heart**	**ionosphere**	**sonar**	**splice**
AB	0.150 ± 0.044	0.029 ± 0.014	0.259 ± 0.021	0.003 ± 0.004	0.16 ± 0.055	0.108 ± 0.060	0.297 ± 0.080	0.215 ± 0.027
CG	0.151 ± 0.053	0.035 ± 0.016	0.249 ± 0.038	0.022 ± 0.015	0.185 ± 0.038	0.104 ± 0.062	0.258 ± 0.085	0.235 ± 0.035

All tests are run five times.

smaller than $\chi^2_{1,0.95}$. Consequently, we can conclude that, indeed, AdaBoost-CG performs very similarly to AdaBoost for classification.

To examine the effect of parameter T, we run more experiments with various T on the *banana* data set (2D artificial data) that was used in [32]. We still use decision stumps. The maximum iteration is set to 400. All runs stop earlier than 100 iterations. Table 7 reports the results. Indeed, the training error depends on T. T also has influence on the convergence speed. But, in a wide range of T, the test error does not change significantly. We do not have a sophisticated technique to tune T. As mentioned, the sum of w from a run of AdaBoost can serve as a heuristic.

Now, let us take a close look at the convergence behavior of AdaBoost-CG. Fig. 5 shows the test and training error of AdaBoost and AdaBoost-CG for six data sets. We see that AdaBoost-CG converges much faster than AdaBoost in

TABLE 7
AdaBoost-CG on *Banana* Data Set
with Decision Stumps and LDA as Weak Classifiers

$\frac{1}{T}$	test (stumps)	train (stumps)	test (LDA)	train (LDA)
20	0.298 ± 0.018	0.150 ± 0.019	0.134 ± 0.012	0.032 ± 0.007
40	0.309 ± 0.019	0.101 ± 0.015	0.135 ± 0.008	0.001 ± 0.002
80	0.313 ± 0.019	0.033 ± 0.011	0.136 ± 0.007	0 ± 0

Experiments are run 50 times.

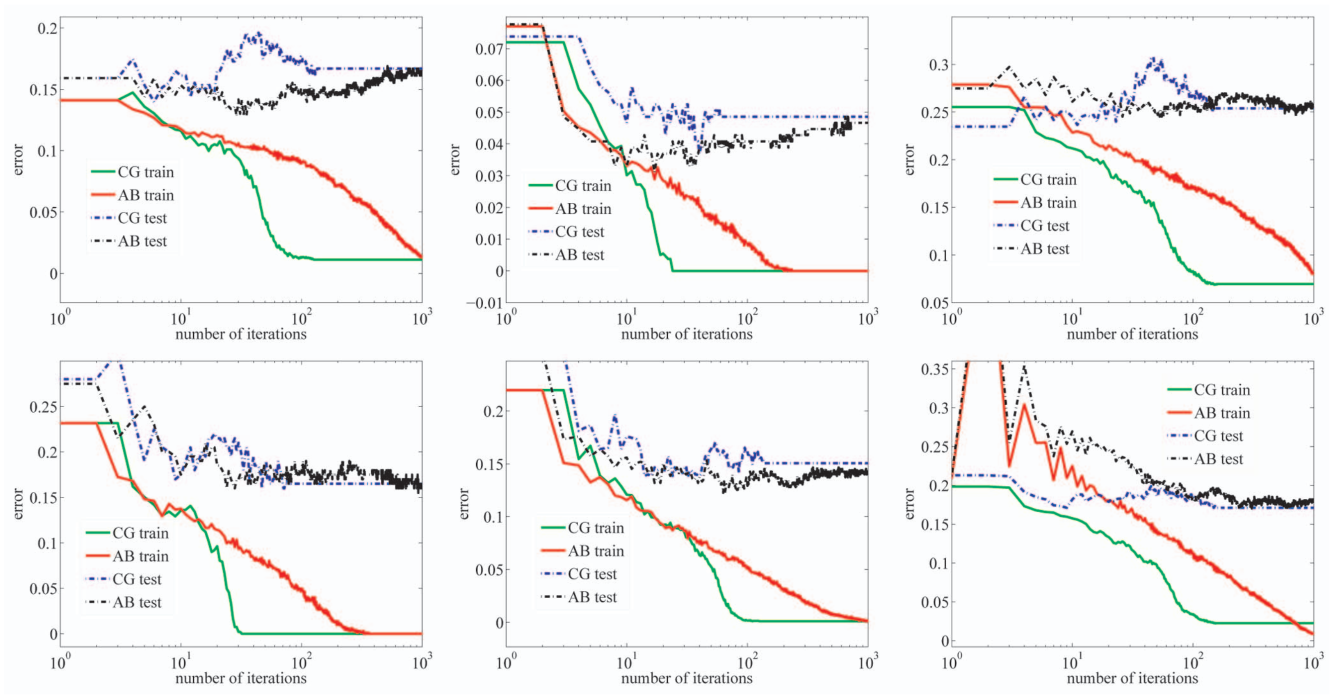

Fig. 5. Test error and training error of AdaBoost, AdaBoost-CG for *australian*, *breast-cancer*, *diabetes*, *heart*, *spline*, and *svmguide3* data sets. These convergence curves correspond to the results in Table 4. The x-axis is on a logarithmic scale for easier comparison.

terms of number of iterations. On most tested data sets, AdaBoost-CG is around 10 times faster than AdaBoost. The test error for these two methods is very close upon convergence. In some data sets such as *australian* and *breast-cancer*, we observe overfitting for AdaBoost.

4.3 LogitBoost-CG

We have also run LogitBoost-CG on the same data sets. All of the settings are the same as in the case of AdaBoost-CG. The weak classifiers are decision stumps. Table 8 reports the experiment results. Compared to Table 5, very similar results have been observed. No one achieves better results over the other one on all the data sets.

5 Discussion and Conclusion

In this paper, we have shown that the Lagrange dual problems of AdaBoost, LogitBoost, and soft-margin LPBoost with generalized hinge loss are all entropy-regularized LPBoost. We both theoretically and empirically demonstrate that the success of AdaBoost relies on maintaining a better margin distribution. Based on the dual formulation, a general column generation-based optimization framework is proposed. This optimization framework can be applied to solve all of the boosting algorithms with various loss functions mentioned in this paper. Experiments with exponential loss show that the classification performance of AdaBoost-CG is statistically almost identical to the standard stagewise AdaBoost on real data sets. In fact, since both algorithms optimize the same cost function, we would be surprised to see a significant different in their generalization error. The main advantage of the proposed algorithms is significantly faster convergence speed.

Compared with the conventional AdaBoost, a drawback of AdaBoost-CG is the introduction of a parameter, the same as in LPBoost. While one can argue that AdaBoost implicitly determines this same parameter by selecting how many iterations to run, the stopping criterion is nested, and thus, efficient to learn. In the case of AdaBoost-CG, it is not clear how to efficiently learn this parameter. Currently, one has to run the training procedure multiple times for cross validation.

With the optimization framework established here, some issues on boosting that were previously unclear may become obvious now. For example, for designing cost-sensitive boosting or boosting on uneven data sets, one can simply modify the primal cost function (5) to have a weighted cost function [40]. The training procedure follows AdaBoost-CG.

To summarize, the convex duality of boosting algorithms presented in this work generalizes the convex duality in LPBoost. We have shown some interesting properties that the derived dual formation possesses. The duality also leads to new efficient learning algorithms. The duality provides

TABLE 8
Test Error of LogitBoost-CG with Decision Stumps, Using Cross Validation to Select the Optimal T

dataset	australian	b-cancer	diabetes	fourclass	heart	ionosphere	sonar	splice
test error	0.13 ± 0.043	0.039 ± 0.012	0.238 ± 0.057	0.071 ± 0.034	0.14 ± 0.095	0.2 ± 0.069	0.169 ± 0.05	0.104 ± 0.021

All tests are run five times.

TABLE 9
Description of the Data Sets

dataset	# examples	# features	dataset	# examples	# features
australian	690	14	**liver-disorders**	345	6
breast-cancer	683	10	**mushrooms**	8124	112
diabetes	768	8	**sonar**	208	60
fourclass	862	2	**splice**	1000	60
german-numer	1000	24	**svmguide1**	3089	4
heart	270	13	**svmguide3**	1243	22
ionosphere	351	34	**w1a**	2477	300

Except for mushrooms, svmguide1, svmguide3, *and* w1a, *all of the other data sets have been scaled to* $[-1, 1]$.

useful insights on boosting that may be lacking in existing interpretations [2], [6].

In the future, we want to extend our work to boosting with nonconvex loss functions such as BrownBoost [41]. Also, it should be straightforward to optimize boosting for regression using column generation. We are currently exploring the application of AdaBoost-CG to efficient object detection due to its faster convergence, which is more promising for feature selection [38].

APPENDIX A
DESCRIPTION OF DATA SETS

Table 9 provides a description of the data sets we have used in the experiments.

ACKNOWLEDGMENTS

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program. The authors thank Sebastian Nowozin for helpful discussions.

REFERENCES

[1] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of Online Learning and an Application to Boosting," *J. Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[2] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *Annals of Statistics,* vol. 26, no. 5, pp. 1651-1686, 1998.

[3] C. Rudin, I. Daubechies, and R.E. Schapire, "The Dynamics of AdaBoost: Cyclic Behavior and Convergence of Margins," *J. Machine Learning Research,* vol. 5, pp. 1557-1595, 2004.

[4] C. Rudin, R.E. Schapire, and I. Daubechies, "Analysis of Boosting Algorithms Using the Smooth Margin Function," *Annals of Statistics,* vol. 35, no. 6, pp. 2723-2768, 2007.

[5] D. Mease and A. Wyner, "Evidence Contrary to the Statistical View of Boosting," *J. Machine Learning Research,* vol. 9, pp. 131-156, 2008.

[6] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting (with Discussion and a Rejoinder by the Authors)," *Annals of Statistics,* vol. 28, no. 2, pp. 337-407, 2000.

[7] C. Domingo and O. Watanabe, "MadaBoost: A Modification of AdaBoost," *Proc. Ann. Conf. Computational Learning Theory,* pp. 180-189, 2000.

[8] G. Rätsch and M.K. Warmuth, "Efficient Margin Maximizing with Boosting," *J. Machine Learning Research,* vol. 6, pp. 2131-2152, 2005.

[9] A.J. Grove and D. Schuurmans, "Boosting in the Limit: Maximizing the Margin of Learned Ensembles," *Proc. Nat'l Conf. Artificial Intelligence,* pp. 692-699, 1998.

[10] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor, "Linear Programming Boosting via Column Generation," *Machine Learning,* vol. 46, nos. 1-3, pp. 225-254, 2002.

[11] L. Breiman, "Prediction Games and Arcing Algorithms," *Neural Computation,* vol. 11, no. 7, pp. 1493-1517, 1999.

[12] L. Reyzin and R.E. Schapire, "How Boosting the Margin Can Also Boost Classifier Complexity," *Proc. Int'l Conf. Machine Learning,* 2006.

[13] V. Koltchinskii and D. Panchenko, "Empirical Margin Distributions and Bounding the Generalization Error of Combined Classifiers," *Annals of Statistics,* vol. 30, no. 1, pp. 1-50, 2002.

[14] G. Rätsch, S. Mika, B. Schölkopf, and K.-R. Müller, "Constructing Boosting Algorithms from SVMs: An Application to One-Class Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 9, pp. 1184-1199, Sept. 2002.

[15] R. Meir and G. Rätsch, "An Introduction to Boosting and Leveraging," *Advanced Lectures on Machine Learning,* pp. 118-183, Springer-Verlag, 2003.

[16] R.E. Schapire, "The Boosting Approach to Machine Learning: An Overview," *Nonlinear Estimation and Classification,* pp. 149-172, Springer, 2003.

[17] M. Collins, R.E. Schapire, and Y. Singer, "Logistic Regression, AdaBoost and Bregman Distances," *Machine Learning,* vol. 48, nos. 1-3, pp. 253-285, 2002.

[18] J. Kivinen and M.K. Warmuth, "Boosting as Entropy Projection," *Proc. Ann. Conf. Computational Learning Theory,* pp. 134-144, 1999.

[19] G. Lebanon and J. Lafferty, "Boosting and Maximum Likelihood for Exponential Models," *Advances in Neural Information Processing Systems,* pp. 447-454, MIT Press, 2001.

[20] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Functional Gradient Techniques for Combining Hypotheses," *Advances in Large Margin Classifiers,* ch. 12, pp. 221-247, MIT Press, 1999.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge Univ. Press, 2004.

[22] Y. Freund and R.E. Schapire, "Adaptive Game Playing Using Multiplicative Weights," *Games and Economic Behavior,* vol. 29, pp. 79-103, 1999.

[23] R.M. Rifkin and R.A. Lippert, "Value Regularization and Fenchel Duality," *J. Machine Learning Research,* vol. 8, pp. 441-479, 2007.

[24] S. Shalev-Shwartz and Y. Singer, "On the Equivalence of Weak Learnability and Linear Separability: New Relaxations and Efficient Boosting Algorithms," *Proc. Ann. Conf. Computational Learning Theory,* 2008.

[25] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/, 2001.

[26] P. Bartlett, M. Jordan, and J. McAuliffe, "Convexity, Classification, and Risk Bounds," *J. Am. Statistical Assoc.,* vol. 101, no. 473, pp. 138-156, 2004.

[27] MOSEK ApS "The MOSEK Optimization Toolbox for Matlab Manual, Version 5.0, Revision 93," http://www.mosek.com/, 2008.

[28] ILOG, Inc., "CPLEX 11.1," http://www.ilog.com/products/cplex/, 2008.

[29] C. Tsallis, "Possible Generalization of Boltzmann-Gibbs Statistics," *J. Statistical Physics,* vol. 52, pp. 479-487, 1988.

[30] O. Kallenberg, *Foundations of Modern Probability.* Springer-Verlag, 1997.

[31] W. Feller, *Introduction to Probability Theory and its Applications,* third ed., vol. 1. John Wiley & Sons, 1968.

[32] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft Margins for AdaBoost," *Machine Learning,* vol. 42, no. 3, pp. 287-320, http://theoval.cmp.uea.ac.uk/gcc/matlab/index.shtml, 2001.
[33] J. Wu, M.D. Mullin, and J.M. Rehg, "Linear Asymmetric Classifier for Cascade Detectors," *Proc. Int'l Conf. Machine Learning,* pp. 988-995, 2005.
[34] M.E. Lübbecke and J. Desrosiers, "Selected Topics in Column Generation," *Operation Research,* vol. 53, no. 6, pp. 1007-1023, 2005.
[35] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large Scale Multiple Kernel Learning," *J. Machine Learning Research,* vol. 7, pp. 1531-1565, 2006.
[36] J. Sochman and J. Malas, "AdaBoost with Totally Corrective Updates for Fast Face Detection," *Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition,* pp. 445-450, 2004.
[37] M.K. Warmuth, J. Liao, and G. Rätsch, "Totally Corrective Boosting Algorithms that Maximize the Margin," *Proc. Int'l Conf. Machine Learning,* pp. 1001-1008, 2006.
[38] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.
[39] T.G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation,* vol. 10, no. 7, pp. 1895-1923, 1998.
[40] J. Leskovec, "Linear Programming Boosting for Uneven Datasets," *Proc. Int'l Conf. Machine Learning,* pp. 456-463, 2003.
[41] Y. Freund, "An Adaptive Version of the Boost by Majority Algorithm," *Machine Learning,* vol. 43, no. 3, pp. 293-318, 2001.

Chunhua Shen received the BSc and MSc degrees from Nanjing University, China, the PhD degree from the School of Computer Science, University of Adelaide, Australia, and the MPhil degree from the Mathematical Sciences Institute, Australian National University. Since October 2005, he has been with National ICT Australia (NICTA), Canberra Research Laboratory, Australia, where he is currently a senior researcher. He is also an adjunct research follow in the Research School of Information Sciences and Engineering, Australian National University. His research interests include statistical machine learning, convex optimization, and their application in computer vision.

Hanxi Li received the BSc and MSc degrees from the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, in 2004 and 2007, respectively. He is currently working toward the PhD degree at the Research School of Information Sciences and Engineering, Australian National University. He is also attached to National ICT Australia (NICTA), Canberra Research Laboratory. His research interests include computer vision and machine learning.