Two New Regularized AdaBoost Algorithms

Yijun Sun Jian Li
Department of Electrical and Computer Engineering
University of Florida Gainesville, FL 32611, USA
Email: sun@dsp.ufl.edu, li@dsp.ufl.edu
William Hager
Department of Mathematics
University of Florida Gainesville, FL 32611, USA
Email: hager@math.ufl.edu

Abstract— **AdaBoost rarely suffers from overfitting problems in low noise data cases. However, recent studies with highly noisy patterns clearly showed that overfitting can occur. A natural strategy to alleviate the problem is to penalize the distribution skewness in the learning process to prevent several hardest examples from spoiling decision boundaries. In this paper, we describe in detail how a penalty scheme can be pursued in the mathematical programming setting as well as in the Boosting setting. By using two smooth convex penalty functions, two new soft margin concepts are defined and two new regularized AdaBoost algorithms are proposed. The effectiveness of the proposed algorithms is demonstrated through a large scale experiment. Compared with other regularized AdaBoost algorithms, our methods can achieve at least the same or much better performances.**

I. Introduction

The adaptive boosting (AdaBoost) algorithm is considered one of the most important developments in the classification methodologies in recent years and has been used with great success in many applications [1],[2],[3]. It has been shown that in the low noise regime AdaBoost rarely suffers from overfitting problems. However, recent studies [4],[5],[6] with highly noisy patterns clearly showed that overfitting can occur and several regularized boosting algorithms [6],[7],[8] have been proposed to extend the applicability of AdaBoost to noisy data. AdaBoost$_{\mathrm{Reg}}$ [6] is one of the first boosting-like algorithms that achieved the state-of-the-art generalization results on noisy data. It implemented an intuitive idea of controlling the tradeoff between the margin and the sample influences to achieve a soft margin. In their experimental evaluations, it was found to be among the best performing ones. However, the problem with AdaBoost$_{\mathrm{Reg}}$ is that it is difficult to analyze the underlying optimization scheme since the modification is done on the algorithm level [1],[5].

In this paper, we study the regularized AdaBoost algorithms from the viewpoint of mathematical programming and propose two regularization schemes to improve the robustness of AdaBoost against noisy data. By studying the connection between the minimax optimization problem and AdaBoost, we show that the good generalization performance of AdaBoost can be explained by the fact that the classification performance in the worst case is optimized, which also explains that in noise data cases AdaBoost will eventually lead to overfitting since the data samples can be highly overlapped and even mislabelled. Therefore some forms of regularization are mandatory. A natural regularization strategy is to use the concept of soft margin, i.e., the algorithm does not attempt to classify all of the training samples according to their labels but allows for some errors. One typical example is LP$_{\mathrm{reg}}$-AdaBoost which introduces slack variables into an optimization problem in the primal domain. It is equivalent to constraining the distributions into a box in the dual domain, which can be understood as adding a penalty of 0 within the box and ∞ outside the box. Therefore, this scheme is somewhat heuristic and may be too restrictive [10]. In this paper, we instead consider controlling the distribution skewness by adding a convex penalty function to the objective function in a minimax problem formulation, which leads to a piecewise convex optimization problem. Through a linear approximation, this problem can be solved in both the dual and primal domains. In particular, two algorithms based on the different penalty functions, referred to as AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$, are proposed. These two algorithms can be considered as an extension of AdaBoost$_{\mathrm{Reg}}$ in term of pursuing a soft margin. However, they can achieve better performances than AdaBoost$_{\mathrm{Reg}}$. In particular, the performance of AdaBoost$_{\mathrm{KL}}$ is the best among the regularized AdaBoost algorithms we test in this paper.

The rest of the paper is organized as follows. First, in Section II we present a brief review of AdaBoost. In Section III we study the connection between the minimax optimization problem and AdaBoost. Based on these discussions, two new algorithms, namely AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$, are proposed. In Section IV a large scale experiment based on several artificial and real world datasets are performed. The results are compared with those of the AdaBoost$_{\mathrm{Reg}}$, ν-Arc [8] and C-Barrier [5] algorithms. We finally conclude the paper in Section V. Throughout this paper, we use a bold lowcase letter to denote a vector and a bold uppercase letter to denote a matrix. The ij^{th} entry of a matrix $\mathbf{Z}$ is written as z_{ij}. $\mathbf{z}_{.i}$ and $\mathbf{z}_{j.}$ are the i^{th} column and j^{th} row of $\mathbf{Z}$, respectively. We also use $\tilde{\mathbf{a}}$ to denote the unnormalized vector of $\mathbf{a}$, i.e., $\mathbf{a} = \frac{\tilde{\mathbf{a}}}{\|\tilde{\mathbf{a}}\|_1}$, where $\|\cdot\|_p$ is the p-norm.

II. AdaBoost

We begin with some notations. Suppose we have a training data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \in \mathcal{R}^l \times \{\pm 1\}$. Given a class of hypothesis functions $\mathcal{H} = \{h(\mathbf{x}) : \mathbf{x} \to \pm 1\}$, called *weak learners*, we are interested in finding an ensemble function $F(\mathbf{x})$ which is constructed as follows: $F(\mathbf{x}) = \sum_t \tilde{\alpha}_t h_t(\mathbf{x})$ and $f(\mathbf{x}) = F(\mathbf{x})/\sum_t \tilde{\alpha}_t$ such that a cost function is minimized. Both the combination coefficients $\tilde{\boldsymbol{\alpha}}$ and the hypothesis functions $h_t(\mathbf{x})$ are learned in the learning process. Toward this end, in the past several years, several ensemble methods [3],[11],[12] have been developed. Among them, adaptive boosting (AdaBoost) is the most popular one and is generally considered as a first step towards more practical Boosting algorithm development. (See, for example, a good tutorial paper [1] for more detailed discussions.) The pseudocode of AdaBoost is presented as follows:

```
AdaBoost
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, Maximum iteration number $T$, $d^{(1)}(n) = 1/N$
for $t = 1 : T$
    1. Train weak learner with respect to distribution $\mathbf{d}^{(t)}$ and get hypothesis $h_t(\mathbf{x}) : \mathbf{x} \to \{\pm 1\}$.
    2. Calculate the weighted training error $\epsilon_t$ of $h_t$:
       $$\epsilon_t = \sum_{n=1}^N d^{(t)}(n)\mathbf{I}(y_n \neq h_t(\mathbf{x}_n))$$
       where $\mathbf{I}(\cdot)$ is the indicator function.
    3. Compute the combination coefficient:
       $$\tilde{\alpha}_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$
    4. Update weights:
       $$d^{(t+1)}(n) = d^{(t)}(n)\exp\left(-\tilde{\alpha}_t y_n h_t(\mathbf{x}_n)\right)/C_t$$
       where $C_t$ is the normalization constant such that $\sum_{n=1}^N d^{(t+1)}(n) = 1$.
end
Output : $F(\mathbf{x}) = \sum_{t=1}^T \tilde{\alpha}_t h_t(\mathbf{x})$
```

In [13], an interesting interpretation of AdaBoost as an algorithm performing the stage-wise gradient descent procedure in the sample average of a cost function of the margin distributions was provided. In particular, the cost function is defined as:

$$G = \frac{1}{N}\sum_{n=1}^N \exp\left(-y_n F(\mathbf{x}_n)\right) = \frac{1}{N}\sum_{n=1}^N \exp\left(-\rho(\mathbf{x}_n)\sum_t \tilde{\alpha}_t\right)$$

where $\rho(\mathbf{x}_n) = y_n f(\mathbf{x}_n)$ is the margin of the sample $\mathbf{x}_n$ with respect to the classifier $f(\mathbf{x}_n)$. At the t^{th} iteration, a hypothesis $h_t(\mathbf{x})$ is trained to minimize the weighted error and then the associated combination coefficient $\tilde{\alpha}_t$ is found by a line search to minimize the intermediate cost function: $G_t = \frac{1}{N}\sum_{n=1}^N \exp\left(-y_n(\sum_{i=1}^{t-1}\tilde{\alpha}_i h_i(\mathbf{x}_n) + \tilde{\alpha}_t h_t(\mathbf{x}_n))\right)$. In the binary classification case $\tilde{\alpha}_t$ can be computed analytically as a solution to $\partial G_t/\partial\tilde{\alpha}_t = 0$ which gives the closed form in the pseudocode. Another similar interpretation of AdaBoost as a gradient descent method in a hypothesis space $\mathcal{H}$ presented in [6],[14] is to consider the updating of the distribution $d^{(t)}(n)$ as to normalizing the gradient of G_t with respect to $\rho(\mathbf{x}_n)$: $d^{(t)}(n) = \partial G_{t-1}/\partial\rho(\mathbf{x}_n)/\sum_j \partial G_{t-1}/\partial\rho(\mathbf{x}_j)$, which provides the answers to the question of which pattern should increase the margin most strongly in order to decrease G maximally.

All of the above discussions greatly facilitate our understanding of the impressive generalization capability of AdaBoost. Although the strict proofs are still missing, it is widely believed by many researchers that AdaBoost asymptotically approximates the solution to the following linear programming (LP) problem [6],[8],[14],[15]:

$$\begin{array}{ll} \max_{(\rho,\boldsymbol{\alpha})} & \rho \\ \text{subject to} & \rho(\mathbf{x}_n) \geq \rho, \quad n = 1,\cdots,N \\ & \sum_t \alpha_t = 1, \boldsymbol{\alpha} \geq 0 \end{array} \tag{1}$$

This observation has also motivated many researchers to design new ensemble classifiers directly in the mathematical optimization setting and introduce new ideas into the Boosting setting for new AdaBoost-like algorithms [4],[5],[6],[8],[16]. This is also the main strategy of this paper.

III. Regularized AdaBoost

We start with the minimax problem. The connection between the well-known minimax problem [17] and the AdaBoost algorithm was first noted in [14],[18] and was used to determine the maximum margin that one can achieve given a hypothesis class by exploiting the dual relationship in linear programming. For the sake of simplicity, at the moment, we assume that the cardinality of the hypothesis function set is finite and is equal to T. Define a gain matrix $\mathbf{Z}$ where $z_{nt} = y_n h_t(\mathbf{x}_n)$ is the margin of the sample $\mathbf{x}_n$ with respect to the t^{th} hypothesis function h_t. Now let us look at the following minimax optimization problem:

$$\max_{\boldsymbol{\alpha}\in\Gamma^T}\min_{\mathbf{d}\in\Gamma^N} \mathbf{d}^T\mathbf{Z}\boldsymbol{\alpha} \tag{2}$$

where Γ^T is the distribution simplex defined as $\Gamma^T = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} \in \mathcal{R}^T, \sum_{t=1}^T \alpha_t = 1, \boldsymbol{\alpha} \geq 0\}$. The optimization scheme can be simply understood as finding a set of combination coefficients $\boldsymbol{\alpha} \in \Gamma^T$, such that the performance of the ensemble classifier in the worst case is maximized. It is easy to show that this classification scheme will lead to the maximum margin scheme in (1). In the separable case, a large margin is usually conducive to good generalization in the sense that if a large margin can be achieved with respect to the data, an upper bound on the generalization error is small. However, in the noisy data case where the data distribution is highly overlapped and some data samples can even be mislabelled, the maximum margin scheme can be easily misled by the outlier data. Consequently it will lead to a classifier with a suboptimal performance. Note that in the minimax problem, the minimization takes place over the entire probability space, which is not sufficiently restrictive. A natural strategy is to constrain the distribution or add a penalty term to the cost function to control the distribution skewness so that the algorithm is not allowed to use all of its resources to deal with several hard-to-learn samples. In the following subsection, we will present three regularized AdaBoost algorithms that fall in this framework.

A. LP$_{reg}$-AdaBoost

By constraining the distribution to a box $\mathbf{0} \leq \mathbf{d} \leq \mathbf{c}$, we get the following optimization problem:

$$\max_{\boldsymbol{\alpha}\in\Gamma^T} \min_{\{\mathbf{d}\in\Gamma^N, \mathbf{d}\leq\mathbf{c}\}} \mathbf{d}^T\mathbf{Z}\boldsymbol{\alpha} \tag{3}$$

where $\mathbf{c}$ is a constant vector and usually takes a form of $\mathbf{c} = C\mathbf{1}$ with C being a predefined parameter and $\mathbf{1} \in \mathcal{R}^N$ being a vector of all ones. The physical meaning of (3) can be understood as to finding a set of combination coefficients $\boldsymbol{\alpha}$ such that the classification performance in the worst case within the distribution box is maximized. The LP equivalent to (3) is:

$$\begin{array}{ll} \max_{(\rho,\boldsymbol{\lambda},\boldsymbol{\alpha}\in\Gamma^T)} & \rho - \sum_{n=1}^N c_n\lambda_n \\ \text{subject to} & \sum_{t=1}^T \alpha_t z_{nt} \geq \rho - \lambda_n, n = 1, \cdots, N \\ & \lambda_n \geq 0, n = 1, \cdots, N \end{array} \tag{4}$$

LP$_{\text{reg}}$-AdaBoost [6] is a special case of (4) obtained by setting $c_1 = c_2 = \cdots = c_N = C$. A similar scheme is also used in Support Vector Machine (SVM) [9] for nonseparable data cases. The scheme (4) introduces a nonnegative slack variable λ_n into the optimization problem to achieve a soft margin for a pattern: $\rho_s(\mathbf{x}_n) = \rho(\mathbf{x}_n) + \lambda_n$. The relaxation of the hard margin allows some patterns to have a smaller margin than ρ and the algorithm does not classify all of the patterns according to their associated class labels.

For the convenience of the following discussions, we reformulate (3) slightly as $\max_{\boldsymbol{\alpha}\in\Gamma^T} \min_{\mathbf{d}\in\Gamma^N} \mathbf{d}^T\mathbf{Z}\boldsymbol{\alpha} + \beta(\|\mathbf{d}\|_\infty)$ where $\beta(P)$ is a function defined by: $\beta(P) = 0$ if $P \leq C$ and ∞ if $P > C$. Note that the box defined by $\{\mathbf{d} : \|\mathbf{d}\|_\infty \leq C, \mathbf{d} \in \Gamma^N\}$ is centered on the distribution center $\mathbf{d}_0 = [1/N, \cdots, 1/N]$ and the parameter C reflects to some extent the distribution skewness between the box boundary and $\mathbf{d}_0$. It shows that LP$_{\text{reg}}$-AdaBoost can be considered as a penalty scheme with a penalty of 0 within the box and ∞ outside the box. Therefore, this scheme is somewhat heuristic and may be too restrictive. Some other smooth penalty functions can be considered.

We make a brief discussion on the implementation of LP$_{\text{reg}}$-AdaBoost. In practical applications, the cardinality of the hypothesis function set can be infinite and thus the gain matrix $\mathbf{Z}$ does not exist in an explicit form. As a result, the linear programming cannot be implemented directly. To overcome the problem, several algorithms have been proposed. Two typical examples are the ν-Arc [8] and C-Barrier algorithms [5]. We will compare these methods with our proposed algorithms in Section IV. From now on we use $|\mathcal{H}|$ to denote the cardinality of the hypothesis function set and reserve T as the iteration number of the AdaBoost algorithm.

B. AdaBoost$_{KL}$

To control the skewness of the distribution, one strategy is to add a penalty term $P(\mathbf{d})$, which measures the distances between the query distributions and the distribution center, to the cost function of (2). It leads to the following optimization problem:

$$\max_{\boldsymbol{\alpha}\in\Gamma^{|\mathcal{H}|}} \min_{\mathbf{d}\in\Gamma^N} \mathbf{d}^T\mathbf{Z}\boldsymbol{\alpha} + \beta P(\mathbf{d}) \tag{5}$$

where $\beta > 0$ is a predefined parameter controlling the distribution skewness and the training performance. With a mild assumption of $P(\mathbf{d})$ being a convex function of $\mathbf{d}$, we have the following lemma:

Lemma 1: If $P(\mathbf{d})$ is a convex function of $\mathbf{d}$, the following optimization schemes are equivalent:

$$\begin{array}{ll} & \max_{\boldsymbol{\alpha}\in\Gamma^{|\mathcal{H}|}} \min_{\mathbf{d}\in\Gamma^N} \mathbf{d}^T\mathbf{Z}\boldsymbol{\alpha} + \beta P(\mathbf{d}) \\ = & \min_{\mathbf{d}\in\Gamma^N} \gamma + \beta P(\mathbf{d}) \\ & \text{subject to} \sum_{n=1}^N d_n z_{nj} \leq \gamma, \quad j = 1, \cdots, |\mathcal{H}| \end{array} \tag{6}$$

Proof: Lemma 1 can be proved by interchanging the max and min on the left side of (6) since the function $\mathbf{d}^T\mathbf{Z}\boldsymbol{\alpha} + \beta P(\mathbf{d})$ is convex in $\mathbf{d}$ and concave in $\boldsymbol{\alpha}$, and the sets Γ^N and Γ^T are convex and compact (Generalized Minimax Theorem [19]). Lemma 1 tells us that if we use a convex penalty function of $\mathbf{d}$, the regularization scheme can be pursued directly in the dual domain.

One commonly used metric for the discrete distribution is the Kullback-Leibler distance which is given as: $\text{KL}(\mathbf{d}, \mathbf{d}_0) = \sum_{n=1}^N d_n \ln \frac{d_n}{1/N}$. $\text{KL}(\mathbf{d}, \mathbf{d}_0)$ is convex over the region $\mathbf{d} > 0$ since the Hessian matrix is positive definite. Following Lemma 1, the regularized scheme in the dual domain can be written as:

$$\begin{array}{ll} \min_{(\gamma,\mathbf{d}\in\Gamma^N)} & \gamma + \beta \sum_{n=1}^N d_n \ln \frac{d_n}{1/N} \\ \text{subject to} & \sum_{n=1}^N d_n z_{nj} \leq \gamma, j = 1, \cdots, |\mathcal{H}| \end{array} \tag{7}$$

This scheme is also suggested in [20] from the viewpoint of the Total Corrective Algorithm [21]. The problem (7) can be reformulated as:

$$\begin{array}{ll} \min_{(\gamma,\mathbf{d}\in\Gamma^N)} & \gamma \\ \text{subject to} & s_j(\mathbf{d}) = \sum_{n=1}^N d_n z_{nj} + \beta \sum_{n=1}^N d_n \ln \frac{d_n}{1/N} \leq \gamma \\ & j = 1, \cdots, |\mathcal{H}| \end{array} \tag{8}$$

Define $s(\mathbf{d}) = \max_{1\leq j\leq|\mathcal{H}|} s_j(\mathbf{d})$. Note that $s(\mathbf{d})$ is also a convex function. Suppose now we have a set of query distributions $\mathcal{S} = \{\mathbf{d}^{(t)}\}_{t=1}^T$. For each query distribution $\mathbf{d}^{(t)}$, we can find a supporting hyperplane to the epigraph of $s(\mathbf{d})$. The equation of the supporting hyperplane is given by: $\gamma = s(\mathbf{d}^{(t)}) + \zeta_t^s(\mathbf{d} - \mathbf{d}^{(t)})$ where ζ_t^s is an element of the subdifferential $\partial s(\mathbf{d}^{(t)})$ of s at $\mathbf{d}^{(t)}$. Due to the convexity of $s(\mathbf{d})$, a supporting hyperplane gives an underestimate of s. More precisely, the equation of a supporting hyperplane can be written as:

$$\begin{aligned} \gamma &= \max_{1\leq j\leq|\mathcal{H}|} s_j(\mathbf{d}^{(t)}) + \zeta_t^s(\mathbf{d} - \mathbf{d}^{(t)}) \\ &= \mathbf{z}_{.t}^T\mathbf{d}^{(t)} + \beta \sum_{n=1}^N d_n^{(t)} \ln \frac{d_n^{(t)}}{1/N} \\ &\quad + \left(\mathbf{z}_{.t} + \beta \begin{bmatrix} \ln \frac{d_1^{(t)}}{1/N} + 1 \\ \vdots \\ \ln \frac{d_N^{(t)}}{1/N} + 1 \end{bmatrix} \right)^T (\mathbf{d} - \mathbf{d}^{(t)}) \\ &= \left(\mathbf{z}_{.t} + \beta \ln \frac{\mathbf{d}^{(t)}}{1/N} \right)^T \mathbf{d} \end{aligned} \tag{9}$$

where

$$\mathbf{z}_{.t} = [y_1 h_t(\mathbf{x}_1), \cdots, y_N h_t(\mathbf{x}_N)]^T$$

and

$$h_t = \arg\max_{h\in\mathcal{H}} \sum_{n=1}^{N} d_n^{(t)} h(\mathbf{x}_n) y_n.$$

Define $\widetilde{\mathbf{Z}} = \mathbf{Z} + \beta\left[\ln\frac{\mathbf{d}^{(1)}}{1/N}, \cdots, \ln\frac{\mathbf{d}^{(T)}}{1/N}\right]$. Note that $\widetilde{\mathbf{Z}}$ can be interpreted as a new gain matrix and it means that adding a penalty function to (2) ends up with a modification of the gain matrix which encodes the distribution information in the hypothesis decisions. Now the optimization problem (7) can be approximated as:

$$\min_{(\gamma,\mathbf{d}\in\Gamma^N)} \gamma \qquad \text{subject to} \quad \tilde{\mathbf{z}}_{\cdot t}^T \mathbf{d} \le \gamma, \quad t = 1,\cdots,T \qquad (10)$$

It is a linear programming that is easier to deal with than the original problem. However, this is only a linear approximation that gets better as more constraints are added. The query distributions can be obtained through the column generation technique and the finite convergence of the optimization problem (7) can be guaranteed. However, column generation usually shows a pattern of slow convergence due to the degeneracy of (10) and produces many unnecessary query distributions or columns. In [16] column generation was used for implementing LP$_{\text{reg}}$-AdaBoost. The problem of the slow convergence was alleviated by setting a lower bound for each query distribution, i.e., $C_l\mathbf{1} \le \mathbf{d}^{(t)} \le C\mathbf{1}$ with $C_l \ll C$ and consequently the possible query distributions are constrained into an even smaller area. In our case of $\mathbf{d} \in \Gamma^N$, other more sophisticated stabilized column generation techniques may be needed and this topic should be the subject of our future research. In this paper, we only focus on forming a regularized AdaBoost classifier to approximately solve (10).

The dual form of (10) is:

$$\begin{array}{ll} \max_{(\rho,\boldsymbol{\alpha})} & \rho \\ \text{subject to} & \sum_{t=1}^T \alpha_t z_{nt} + \beta\sum_{t=1}^T \alpha_t \ln\frac{d_n^{(t)}}{1/N} = \rho_s(\mathbf{x}_n) \ge \rho \\ & n = 1,\cdots,N \\ & \boldsymbol{\alpha} \in \Gamma^T \end{array} \qquad (11)$$

The soft margin of a pattern $\mathbf{x}_n$ can be defined as: $\rho_s(\mathbf{x}_n) = \sum_{t=1}^T \alpha_t z_{nt} + \beta\sum_{t=1}^T \alpha_t \ln\frac{d_n^{(t)}}{1/N}$. The term $\beta\sum_{t=1}^T \alpha_t \ln\frac{d_n^{(t)}}{1/N}$ can be understood as a "mistrust" in examples. The rationale is: a pattern which is often misclassified (i.e., hard to classify correctly) will have a high average distribution and should have less influence on the outcome of the final classifier. Note also that the mistrust is calculated with respect to the center distribution. For example, if the query distribution $d_n^{(t)} \le 1/N, t = 1,\cdots,T$, the "mistrust" can take a negative value. As a result, the soft margin penalizes some hard examples and at the same time rewards some easy examples. In [6],[22], it was experimentally observed that AdaBoost increases the margin of the most hard-to-learn examples at the cost of reducing the margins of the rest of the data. Therefore, defining a soft margin as above can be understood as reversing the AdaBoost process with the strength being controlled by β.

Now with a soft margin defined as above and following the same strategy as that used in deriving AdaBoost$_{\text{Reg}}$ [5] where AdaBoost is used as a general machine for solving minimax problems, a new AdaBoost-like algorithm, which we refer to as AdaBoost$_{\text{KL}}$, can be formulated. Specifically, we define a new cost function:

$$\begin{aligned} G_{\text{KL}} &= \sum_{n=1}^N \exp\{-\rho_s(\mathbf{x}_n)\sum_t \tilde{\alpha}_t\} &(12)\\ &= \sum_{n=1}^N \exp\left\{-\sum_t \tilde{\alpha}_t z_{nt} - \beta\sum_t \tilde{\alpha}_t \ln\frac{d_n^{(t)}}{1/N}\right\} \end{aligned}$$

where $\tilde{\boldsymbol{\alpha}}$ is the unnormalized version of $\boldsymbol{\alpha}$. The combination coefficient $\tilde{\alpha}_t$ for the t^{th} hypothesis h_t is computed as:

$$\begin{aligned} \tilde{\alpha}_t &= \arg\min_{\tilde{\alpha}_t \ge 0} G_{\text{KL}}^{(t)} &(13)\\ &= \arg\min_{\tilde{\alpha}_t \ge 0} \sum_{n=1}^N \exp\left\{-\sum_{j=1}^t \tilde{\alpha}_j z_{nj} - \beta\sum_{j=1}^t \tilde{\alpha}_j \ln\frac{d_n^{(j)}}{1/N}\right\} \end{aligned}$$

It is difficult to compute $\tilde{\alpha}_t$ analytically. However, we can get $\tilde{\alpha}_t$ efficiently by a line search since $\partial^2 G_{\text{KL}}/\partial^2\tilde{\alpha}_t \ge 0$. The updated distribution $d_n^{(t+1)}$ is computed as the derivative of G_{KL} with respect to $\rho_s(\mathbf{x}_n)$:

$$\begin{aligned} d_n^{(t+1)} &= \frac{\partial G_{\text{KL}}/\partial\rho_s(\mathbf{x}_n)}{\sum_j \partial G_{\text{KL}}/\partial\rho_s(\mathbf{x}_j)} &(14)\\ &= d_n^{(t)} \exp\left\{-\tilde{\alpha}_t h_t(\mathbf{x}_n) y_n - \beta\tilde{\alpha}_t \ln\frac{d_n^{(t)}}{1/N}\right\} / C_t \end{aligned}$$

where C_t is the normalization constant such that $\sum_{n=1}^N d_n^{(t+1)} = 1$. AdaBoost$_{\text{KL}}$ is summarized as follows:

```
AdaBoost_KL
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, Maximum iteration number $T$, $d_n^{(1)} = 1/N$, parameter $\beta$
for $t = 1 : T$
    1. Train weak learner with respect to distribution $\mathbf{d}^{(t)}$ and get hypothesis $h_t(\mathbf{x}) : \mathbf{x} \to \{\pm 1\}$.
    2. Calculate the coefficient $\tilde{\alpha}_t$ of $h_t$ as (13).
    4. Update weights as (14).
end
Output : $F(\mathbf{x}) = \sum_{t=1}^T \tilde{\alpha}_t h_t(\mathbf{x})$
```

It is clear that if $\beta = 0$, we retreat to the original AdaBoost algorithm and if $\beta \to \infty$, it is not difficult to prove that only the first classifier h_1 is kept, i.e., $\alpha_t = 0$, for $t \ge 2$, which corresponds to the single classifier design. It means that by varying the parameter β we are able to *control* the boosting strength of the learning process to alleviate the overfitting problem.

C. *AdaBoost$_{norm2}$*

We can also consider using an l_p norm function as the penalty function. It is easy to show that $\|\mathbf{d} - \mathbf{d}_0\|_p$ is a convex function of $\mathbf{d}$ and following Lemma 1, we get the following regularized scheme in the dual domain:

$$\begin{array}{ll} \min_{(\gamma,\mathbf{d}\in\Gamma^N)} & \gamma + \beta\|\mathbf{d} - \mathbf{d}_0\|_p \\ \text{subject to} & \sum_{n=1}^N d_n z_{nj} \le \gamma, j = 1,\cdots,|\mathcal{H}| \end{array} \qquad (15)$$

Particularly, we only focus on the case of l_2 in this paper. The optimization problem can be reformulated as:

$$\begin{array}{ll} \min_{(\gamma,\mathbf{d}\in\Gamma^N)} & \gamma \\ \text{subject to} & s_j(\mathbf{d}) = \sum_{n=1}^N d_n z_{nj} + \beta\|\mathbf{d}-\mathbf{d}_0\|_2 \le \gamma, \\ & j = 1,\cdots,|\mathcal{H}| \end{array} \tag{16}$$

Define $s(\mathbf{d}) = \max_{1\le j\le|\mathcal{H}|} s_j(\mathbf{d})$. Suppose now we have a set of query distributions $\mathcal{S} = \{\mathbf{d}^{(t)}\}_{t=1}^T$. For each query distribution $\mathbf{d}^{(t)}$, we can find one supporting hyperplane whose equation is given by:

$$\begin{aligned} \gamma &= s(\mathbf{d}^{(t)}) + \zeta_t^s(\mathbf{d}-\mathbf{d}^{(t)}) \\ &= \mathbf{z}_{\cdot t}^T\mathbf{d}^{(t)} + \beta\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2 \\ &\quad + \left(\mathbf{z}_{\cdot t} + \beta\frac{\mathbf{d}^{(t)}-\mathbf{d}_0}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2}\right)^T(\mathbf{d}-\mathbf{d}^{(t)}) \\ &= \left(\mathbf{z}_{\cdot t} + \beta\frac{\mathbf{d}^{(t)}-\mathbf{d}_0}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2}\right)^T\mathbf{d} \end{aligned} \tag{17}$$

Define $\widetilde{\mathbf{Z}} = \mathbf{Z} + \beta\left[\frac{\mathbf{d}^{(1)}-\mathbf{d}_0}{\|\mathbf{d}^{(1)}-\mathbf{d}_0\|_2},\cdots,\frac{\mathbf{d}^{(T)}-\mathbf{d}_0}{\|\mathbf{d}^{(T)}-\mathbf{d}_0\|_2}\right]$. Now the optimization problem (16) can be linearly approximated as:

$$\begin{array}{ll} \min_{(\gamma,\mathbf{d}\in\Gamma^N)} & \gamma \\ \text{subject to} & \tilde{\mathbf{z}}_{\cdot t}^T\mathbf{d} \le \gamma \quad t = 1,\cdots,T \end{array} \tag{18}$$

The dual form of (18) is:

$$\begin{array}{ll} \max_{(\rho,\boldsymbol{\alpha}\in\Gamma^T)} & \rho \\ \text{subject to} & \sum_{t=1}^T \alpha_t z_{nt} + \beta\sum_{t=1}^T \alpha_t \frac{d_n^{(t)}-1/N}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2} \ge \rho \\ & n = 1,\cdots,N \end{array} \tag{19}$$

The soft margin of a pattern $\mathbf{x}_n$ can be defined as: $\rho_s(\mathbf{x}_n) = \sum_{t=1}^T \alpha_t z_{nt} + \beta\sum_{t=1}^T \alpha_t \frac{d_n^{(t)}-1/N}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2}$. Again the term $\beta\sum_{t=1}^T \alpha_t \frac{d_n^{(t)}-1/N}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2}$ can be understood as a "mistrust" in examples with respect to the center distribution. The parameter β controls the tradeoff between margin and "mistrust". It is interesting to note that our soft margin definition is very similar to that in AdaBoost$_{\text{Reg}}$, which is defined as: $\rho_{\text{Reg}}(\mathbf{x}_n) = \sum_{t=1}^T \alpha_t z_{nt} + \beta\sum_{t=1}^T \alpha_t d_n^{(t)}$. The main difference is that our soft margin is calculated with respect to the center distribution and the term $\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2$ can be roughly understood as follows: the closer the query distribution to the center distribution, the more trust the outcome of the hypothesis deserves. Now following the same strategy used in deriving AdaBoost$_{\text{KL}}$, the optimization problem can be easily reformulated into an AdaBoost-like algorithm, which we call AdaBoost$_{\text{norm2}}$. We can define a new cost function:

$$G_{\text{norm2}} = \sum_{n=1}^N \exp\left\{-\sum_t \tilde{\alpha}_t z_{nt} - \beta\sum_t \tilde{\alpha}_t \frac{d_n^{(t)}-1/N}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2}\right\} \tag{20}$$

The combination coefficient $\tilde{\alpha}_t$ is computed as:

$$\tilde{\alpha}_t = \arg\min_{\tilde{\alpha}_t\ge 0} G_{\text{norm2}}^{(t)} \tag{21}$$

Again we can get $\tilde{\alpha}_t$ efficiently by a line search since $\partial^2 G_{\text{norm2}}/\partial^2\tilde{\alpha}_t \ge 0$. The updated distribution $d_n^{(t+1)}$ is computed as the derivative of G_{norm2} with respect to $\rho_s(\mathbf{x}_n)$:

$$\begin{aligned} d_n^{(t+1)} &= \frac{\partial G/\partial\rho_s(\mathbf{x}_n)}{\sum_j \partial G/\partial\rho_s(\mathbf{x}_j)} \\ &= d_n^{(t)}\exp\left\{-\tilde{\alpha}_t h_t(\mathbf{x}_n)y_n - \beta\tilde{\alpha}_t\frac{d_n^{(t)}-1/N}{\|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2}\right\}/C_t \end{aligned} \tag{22}$$

where C_t is the normalization constant such that $\sum_{n=1}^N d_n^{(t+1)} = 1$. AdaBoost$_{\text{norm2}}$ is summarized as follows:

```
**AdaBoost_norm2**
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, Maximum iteration
        number $T$, $d_n^{(1)} = 1/N$, parameter $\beta$
for $t = 1 : T$
        1. Train weak learner with respect to distribution $\mathbf{d}^{(t)}$ and get hypothesis $h_t(\mathbf{x}) : \mathbf{x} \to \{\pm 1\}$.
        2. Calculate the coefficient $\tilde{\alpha}_t$ of $h_t$ as (21).
        4. Update weights as (22).
end
Output : $F(\mathbf{x}) = \sum_{t=1}^T \tilde{\alpha}_t h_t(\mathbf{x})$
```

IV. Experimental Results

To demonstrate the effectiveness of the two newly proposed algorithms, a large scale experiment is conducted and the results are compared with the AdaBoost$_{\text{Reg}}$, ν-Arc and C-Barrier algorithms.

For the fairness of comparison, the experimental setup herein is the same as those used in [6]. Thanks to Rätsch's effort, the detailed information about the experimental setup as well as the benchmark dataset can be found at http://mlg.anu.edu.au/raetsch/data/index.html. We use 13 artificial and real-world data sets originally from the UCI, DELVE and STATLOG benchmark repositions. Each dataset is partitioned into 100 realizations of training and testing data (splice and image have only 20 realizations). For each partition, a classifier is trained and the test error is computed. The RBF (radial basis function) net is used as the weak learner. All of the RBF parameters are the same as those used in [6]. We use the cross-validation method based on the first five realizations of each dataset to estimate the parameter β of the regularized AdaBoost algorithms. The maximum iteration number T is chosen to be 200.

As an example, in Figure 1 we present some training and testing results and margin plots of three methods: AdaBoost, AdaBoost$_{\text{norm2}}$ and AdaBoost$_{\text{KL}}$ based on one realization of the waveform data. AdaBoost tries to maximize the margin of each pattern and hence it can reduce the training error to zero effectively. However it quickly leads to overfitting. In contrast, AdaBoost$_{\text{norm2}}$ and AdaBoost$_{\text{KL}}$ try to maximize the soft margin and allow some hardest examples to have a small margin. The two regularized methods can effectively alleviate the overfitting problem. To provide a more comprehensive comparison, in Table I, the average classification results (standard deviations) over the 100 realizations of the 13 datasets are presented.

- AdaBoost performs worse than a single RBF classifier in almost all cases. It is clearly due to the overfitting

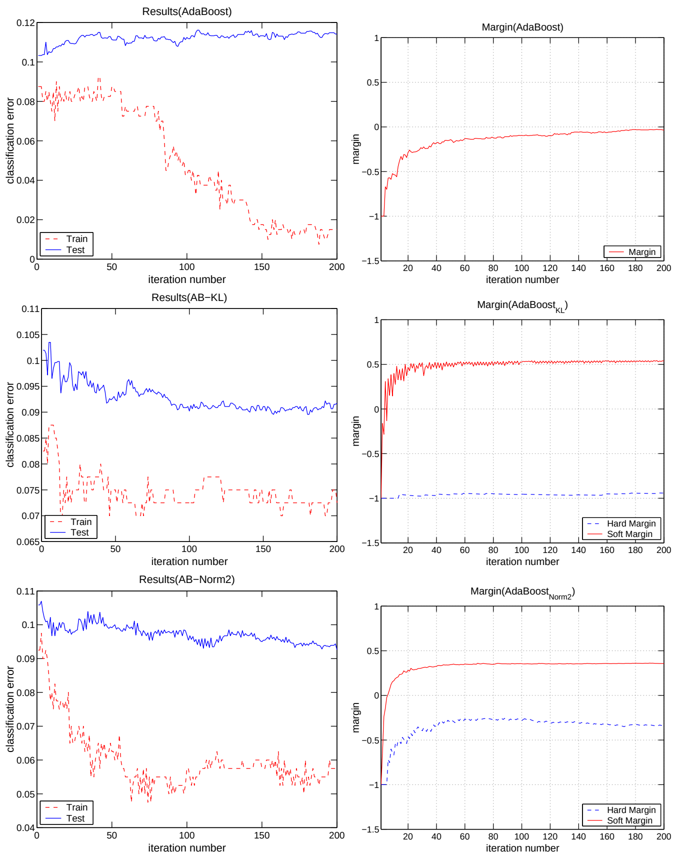

Fig. 1. Training and testing results, and margin plots of three methods: AdaBoost, AdaBoost$_{\text{norm2}}$ and AdaBoost$_{\text{KL}}$ based on the waveform data. AdaBoost quickly leads to overfitting while the regularized methods effectively alleviate this problem.

of AdaBoost. In ten (out of 13) cases AdaBoost$_{\text{Reg}}$ performs significantly better than AdaBoost and in ten cases AdaBoost$_{\text{Reg}}$ performs better than a single RBF classifier.

- The results of AdaBoost$_{\text{norm2}}$ are slightly better than those of AdaBoost$_{\text{Reg}}$ in eight cases, and except for two cases (heart and image), the results of AdaBoost$_{\text{KL}}$ are better than those of AdaBoost$_{\text{Reg}}$. For a more rigorous comparison, a 90% significant test is reported in Table II. For some data sets the performance differences are small (e.g. titanic). This is because AdaBoost$_{\text{Reg}}$ is already a good classifier which was reported to be slightly better than SVM (RBF kernel) [6]. Nevertheless, significant improvements are observed for AdaBoost$_{\text{KL}}$ in five datasets (out of 13) (Table II).
- In Table I, we also compare our algorithms with other regularized boosting algorithms, including ν-Arc and C-Barrier. Again, our algorithms perform better in most cases, which may be explained as due to a hard limited penalty function used in the supporting optimization scheme for the ν-Arc and C-Barrier algorithms.
- An interesting observation is that although AdaBoost is useful for the low noise data case, the results of ringnorm, thyroid and twonorm suggest that the regularized AdaBoost are effective even in the low noise regime. Moreover, in almost all cases, the standard deviations of our regularized AdaBoost algorithms are smaller than those of the single RBF and AdaBoost classifiers.

V. Conclusions

We have made a detailed study on AdaBoost and its regularization variations. Two regularized AdaBoost algorithms have been proposed. By studying the connection between the minimax optimization problem with the maximum margin classification scheme, we have shown that the impressive generalization capability of AdaBoost in the low noise data cases may stem from the fact that the classification performance in the worst case is maximized. It also explained that the overfitting of AdaBoost is inevitable. It is natural to control the distribution skewness in the learning process to prevent the outerlier samples from spoiling decision boundaries. We control the skewness by adding a convex penalty function to the objective of the minimax problem. Through the generalized minimax theorem, we have shown that the penalty scheme can be pursued equivalently in the dual domain and the LP$_{\text{reg}}$-AdaBoost is a special case of the penalty scheme with the penalty function being chosen as a hard limited function. By using two smooth convex penalty functions, two new soft margin concepts have been defined and thereby two new regularized AdaBoost algorithms have been proposed. The regularization is naturally incorporated into the AdaBoost process adaptively. To demonstrate the effectiveness of the proposed algorithms, a large scale experiment has been conducted. Compared with AdaBoost$_{\text{Reg}}$, ν-Arc and C-Barrier, our methods can achieve at least the same or much better performances.

Acknowledgment

This work was in part supported by the U.S. Army under contract No. DAAB15-00-C-1024. We would like to thank Dr. Gunnar Rätsch for providing us with the RBF code and the benchmark data.

References

[1] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," *In S. Mendelson and A. Smola, editors, Advanced Lectures on Machine Learning, LNCS, Springer*, pp. 119–184, 2003.
[2] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, pp. 297–336, December 1999.
[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, pp. 23–37, 1995.
[4] A. J. Grove and D. Schuurmans, "Boosting in the limit: Maximizing the margin of learned ensembles," in *AAAI/IAAI*, pp. 692–699, 1998.
[5] G. Rätsch, "Robust boosting via convex optimization: theory and application," PhD Thesis, University of Potsdam, Germany, October 2001.
[6] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Machine Learning*, vol. 42, pp. 287–320, March 2001.
[7] Y. Freund, "An adaptive version of the boost by majority algorithm," *Machine Learning*, pp. 293–318, June 2001.
[8] G. Rätsch, B. Scholkopf, A. Smola, S. Mika, T. Onoda, and K.-R. Muller, "Robust ensemble learning," *In B. Scholkopf, A. Smola, P. Bartlett and D. Schuurmans, editors, Advances in Large Margin Classifiers. MIT Press*, pp. 207–220, 2000.
[9] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, pp. 273–297, 1995.

TABLE I

CLASSIFICATION ERRORS (STANDARD DEVIATION) OF THE EIGHT METHODS. THE BEST CLASSIFIERS ARE MARKED IN BOLDFACE.

	RBF	AB	AB_R	AB_{KL}	AB_{norm2}	ν_{Arc}	C-Bar	SVM
Banana	10.8±0.6	12.3±0.7	10.9±0.4	**10.7±0.4**	**10.6±0.4**	10.8±0.5	10.9±0.5	11.5±0.7
Bcancer	27.6±4.7	30.4±4.7	26.5±4.5	26.1±4.4	26.0±4.4	**25.8±4.6**	*25.9±4.4*	26.0±4.7
Diabetis	24.3±1.9	26.5±2.3	23.8±1.8	**23.5±1.8**	**23.6±1.8**	23.7±2.0	23.7±1.8	**23.5±1.7**
German	24.7±2.4	27.5±2.5	24.3±2.1	24.2±2.2	24.1±2.2	24.4±2.2	24.3±2.4	**23.6±2.1**
Heart	17.6±3.3	20.3±3.4	16.5±3.5	16.9±3.2	17.0±3.1	16.5±3.5	17.0±3.4	**16.0±3.3**
Ringnorm	1.7±0.2	1.9±0.3	1.6±0.1	**1.5±0.1**	1.6±0.1	1.7±0.2	1.7±0.2	1.7±0.1
Fsolar	34.4±2.0	35.7±1.8	34.2±2.2	34.1±1.6	34.1±1.7	34.4±1.9	33.7±1.9	**32.4±1.8**
Thyroid	4.5±2.1	4.4±2.2	4.6±2.2	**4.3±1.9**	4.4±2.2	4.4±2.2	4.5±2.2	4.8±2.2
Titanic	23.3±1.3	22.6±1.2	22.6±1.2	*22.5±0.9*	*22.5±1.2*	23.0±1.4	**22.4±1.1**	**22.4±1.0**
Waveform	10.7±1.1	10.8±0.6	9.8±0.8	**9.4±0.6**	*9.5±0.4*	10.0±0.7	9.7±0.5	9.9±0.4
Splice	10.0±1.0	10.1±0.5	9.5±0.7	**9.2±0.6**	9.5±0.5	*N/A*	*N/A*	10.9±0.7
Image	3.3±0.6	**2.7±0.7**	**2.7±0.6**	**2.7±0.6**	**2.7±0.5**	*N/A*	*N/A*	3.0±0.6
Twonorm	2.9±0.3	3.0±0.3	*2.7±0.2*	**2.6±0.2**	*2.7±0.2*	*N/A*	*N/A*	3.0±0.2

TABLE II

90% SIGNIFICANT TEST COMPARING ADABOOST$_{KL}$ WITH OTHER METHODS. '0' MEANS THE TEST ACCEPTS THE NULL HYPOTHESIS: NO SIGNIFICANT DIFFERENCE IN AVERAGE PERFORMANCE. '+' MEANS THE TEST ACCEPTS THE ALTERNATIVE HYPOTHESIS: ADABOOST$_{KL}$ IS SIGNIFICANTLY BETTER AND '-' MEANS ADABOOST$_{KL}$ IS SIGNIFICANTLY WORSE.

	AB_{KL}/RBF	AB_{KL}/AB	AB_{KL}/AB_R	AB_{KL}/ν_{Arc}	AB_{KL}/C-Bar	AB_{KL}/SVM
Banana	+	+	+	+	+	+
Bcancer	+	+	0	0	0	0
Diabetis	+	+	0	0	0	0
German	+	+	0	0	0	−
Heart	0	+	0	0	0	−
Ringnorm	+	+	+	+	+	+
Fsolar	0	+	0	0	0	−
Thyroid	0	0	0	0	0	+
Titanic	+	0	0	0	0	0
Waveform	+	+	+	+	+	+
Splice	+	+	+	N/A	N/A	+
Image	+	0	0	N/A	N/A	+
Twonorm	+	+	+	N/A	N/A	+

[10] R. Meir, "Support vector machines: an introduction," *Lecture Notes*, June 2002.

[11] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[12] L. Breiman, "Arcing classifiers," *The Annals of Statistics*, vol. 26(3), pp. 801–849, 1998.

[13] L. Mason, J. Bartlett, P. Baxter, and M. Frean, "Functional gradient techniques for combining hypotheses," *In B. Scholkopf, A. Smola, P. Bartlett and D. Schuurmans, editors, Advances in Large Margin Classifiers. MIT Press*, 2000.

[14] L. Breiman, "Prediction games and arcing algorithms," *Neural Computation*, vol. 11, pp. 1493–1517, October 1999.

[15] R. E. Schapire, "Theoretical views of boosting and applications," in *In Tenth International Conference on Algorithmic Learning Theory*, vol. 1720, pp. 13–25, Springer, 1999.

[16] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Machine Learning*, vol. 46, no. 1-3, pp. 225–254, 2002.

[17] J. von Neumann, "Zur theorie der gesellschaftsspiele," *Math. Ann.*, pp. 100:295–320, 1928.

[18] Y. Freund and R. E. Schapire, "Game theory, on-line prediction and boosting," *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, 1996.

[19] I. Ekeland and R. Temam, *Convex Analysis and Variational Problems*. North-Holland Pub. Co., Amsterdam, 1976.

[20] G. Rätsch, M. Warmuth, S. Mika, T. Onoda, S. Lemm, and K.-R. Müller, "Barrier boosting," *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

[21] J. Kivinen and M. K. Warmuth, "Boosting as entropy projection," in *Computational Learing Theory*, pp. 134–144, 1999.

[22] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: a new explanation for the effectiveness of voting methods," *Proc. 14th International Conference on Machine Learning*, pp. 322–330, 1997.