International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 20, No. 7 (2006) 1093–1116
© World Scientific Publishing Company

REDUCING THE OVERFITTING OF ADABOOST BY CONTROLLING ITS DATA DISTRIBUTION SKEWNESS

YIJUN SUN*,†,‡, SINISA TODOROVIC§ and JIAN LI‡

†*Interdisciplinary Center for Biotechnology Research*
‡*Department of Electrical and Computer Engineering, University of Florida*
Gainesville, FL 32611-6130, USA

§*3021 Beckman Institute of Advanced Science and Technology*
University of Illinois at Urbana-Champaign
405 N. Mathews Ave., Urbana, IL 61801, USA
**sun@dsp.ufl.edu*

AdaBoost rarely suffers from overfitting problems in low noise data cases. However, recent studies with highly noisy patterns have clearly shown that overfitting can occur. A natural strategy to alleviate the problem is to penalize the data distribution skewness in the learning process to prevent several hardest examples from spoiling decision boundaries. In this paper, we pursue such a penalty scheme in the mathematical programming setting, which allows us to define a suitable classifier soft margin. By using two smooth convex penalty functions, based on Kullback–Leibler divergence (KL) and l_2 norm, we derive two new regularized AdaBoost algorithms, referred to as AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$, respectively. We prove that our algorithms perform stage-wise gradient descent on a cost function, defined in the domain of their associated soft margins. We demonstrate the effectiveness of the proposed algorithms through experiments over a wide variety of data sets. Compared with other regularized AdaBoost algorithms, our methods achieve at least the same or better performance.

Keywords: Adaptive boosting (AdaBoost); minimax problem; margin; soft margin; regularization.

1. Introduction

The adaptive boosting (AdaBoost) algorithm is considered one of the most important developments in the classification methodology in recent years. It has been used with great success in many applications.[8,12,21] In the low noise regime, empirical evidence indicates that AdaBoost rarely suffers from overfitting problems. One leading explanation to understand this phenomenon is contemplated to be the margin concept.[20] It has been empirically observed that AdaBoost can effectively increase

*Author for correspondence.

the margin, and a large margin, in turn, is usually conducive to good generalization, in the sense that if a large margin can be achieved with respect to given data, an upper bound on the generalization error is small.[22] Recent studies with highly noisy patterns,[5,10,14,16] however, have shown that overfitting may occur. Therefore, in the light of AdaBoost's increasing popularity, it is important to examine the overfitting phenomenon, and to seek effective solutions, which would enhance the performance of AdaBoost in noisy settings.

It has been reported that the main reason for poor classification results of AdaBoost in the high-noise regime is that the algorithm produces a skewed data distribution, by assigning too much weight onto a few hard-to-learn examples.[5] Therefore, one natural strategy is to introduce a regularization term into the algorithm, which would control the data distribution skewness. Based on this principle, one of the earliest proposed algorithms is $\text{AdaBoost}_{\text{Reg}}$.[16] It is a heuristic algorithm based on an intuitive idea of controlling the tradeoff between the margin and the sample influences to achieve a soft margin. In comparison with other available regularized boosting algorithms, $\text{AdaBoost}_{\text{Reg}}$ yields among the best generalization results on noisy data. However, since the regularization is introduced on the algorithm level, it is difficult to analyze its underlying optimization scheme, and the ultimate goal of the algorithm is obscure.[12,14]

Since it is not straightforward to include a regularization term into AdaBoost, a potentially better way to design new regularized boosting algorithms may be to exploit the close relationship between AdaBoost and the well-known minimax problem. As an advantage of this approach, some of the well-studied mathematical programming techniques can be directly utilized. One typical representative of this strategy is LP_{reg}-AdaBoost, which constitutes the underlying optimization scheme of ν-Arc,[17] and C-Barrier algorithms.[14] In LP_{reg}-AdaBoost, slack variables are introduced into an optimization problem in the primal domain, similar to Support Vector Machine (SVM) in the nonseparable data case. In the dual domain, we show this algorithm is equivalent to constraining the data distribution to a box. As such, this algorithm can be understood as a penalty scheme with a zero penalty within the box and infinity outside the box. In this sense, the scheme is somewhat heuristic, and may be too restrictive.

In this paper, we instead consider controlling the skewness of data distributions by adding a convex penalty function to the objective function of the minimax problem. By means of the generalized minimax theorem, we show that the penalty scheme can be pursued equivalently in the dual domain, wherein we specify the general framework of the proposed regularization. This general framework gives rise to a range of regularized boosting algorithms, differing in a particular specification of the penalty function. For example, we show that LP_{reg}-AdaBoost can be derived from the outlined framework if the penalty is defined as a hard-limited function, which is a novel interpretation of the algorithm. We study two penalty functions that are based on the Kullback–Leibler (KL) divergence, and l_p norm. From the minimax optimization problem, where these two penalty functions are introduced, we derive the soft margin, and two novel regularized AdaBoost algorithms, referred to as AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$. These two algorithms can be viewed as an extension of AdaBoost$_{\mathrm{Reg}}$; the main difference is in specification of the soft margin. We believe that the soft margin in our methods is more reasonable than that in AdaBoost$_{\mathrm{Reg}}$ with respect to two criteria. First, we prove that our algorithms perform the stage-wise gradient descent of a cost function, defined in the domain of the soft margin, whereas AdaBoost$_{\mathrm{Reg}}$ does not have such property. Second, AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$ outperform AdaBoost$_{\mathrm{Reg}}$. To demonstrate the effectiveness of our algorithms, we report experiments on a wide variety of artificial and real-world data sets, where we compare the performance of our algorithms with that of AdaBoost$_{\mathrm{Reg}}$, ν-Arc, C-Barrier and SVM. We record that the classification results of AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$ are among the best.

The rest of the paper is organized as follows. First, in Sec. 2, we present a brief review of AdaBoost. In Sec. 3, we propose two new algorithms, namely AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$. In Sec. 4, we report experiments on a wide variety of data sets, where we compare the performance of our algorithms with that of AdaBoost$_{\mathrm{Reg}}$, ν-Arc, C-Barrier and SVM. We conclude the paper in Sec. 5.

Throughout, a vector is denoted as a boldface low-case letter, and a matrix, as a boldface upper-case letter. The ijth entry of a matrix $\mathbf{Z}$ is written as z_{ij}. $\mathbf{z}_{.i}$ and $\mathbf{z}_{j.}$ are the ith column and jth row of $\mathbf{Z}$, respectively. The unnormalized vector of $\mathbf{a}$ is denoted as $\tilde{\mathbf{a}}$, that is, $\mathbf{a} = \tilde{\mathbf{a}}/\|\tilde{\mathbf{a}}\|_1$, where $\|\cdot\|_p$ is the p-norm.

2. AdaBoost

In this section, we briefly review AdaBoost, as well as its interpretation as a functional gradient-descent procedure. For a thorough description, the interested reader is referred to a good tutorial paper,[12] and references therein.

Suppose we are given a training data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N \in \mathcal{X}\times\mathcal{Y}$, where $\mathcal{X}$ is a pattern space and $\mathcal{Y} = \{\pm 1\}$ is a label space. Given a class of hypothesis functions $\mathcal{H} = \{h(\mathbf{x}) : \mathbf{x} \to \pm 1\}$, called *weak learners* or *base learners*, we are interested in finding an ensemble function $F(\mathbf{x}) = \sum_t \tilde{\alpha}_t h_t(\mathbf{x})$, or $f(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x})$, such that a certain cost function is minimized, where $\alpha_t \triangleq \tilde{\alpha}_t / \sum_t \tilde{\alpha}_t$. Both the vector of combination coefficients $\tilde{\boldsymbol{\alpha}}$ and hypothesis functions $h_t(\mathbf{x})$ are learned in the learning process. Several ensemble methods have been developed for this purpose, among which AdaBoost is the most popular.[8] The pseudo code of AdaBoost is presented in Fig. 1.

AdaBoost can be viewed as an algorithm performing stage-wise gradient descent of a cost function of margins G defined as

$$\begin{aligned} G &\triangleq \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n F(\mathbf{x}_n)), \\ &= \frac{1}{N}\sum_{n=1}^{N} \exp\left(-\rho(\mathbf{x}_n)\sum_t \tilde{\alpha}_t\right), \end{aligned} \tag{4}$$

```
AdaBoost
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, maximum number of iteration steps $T$, $d_n^{(1)} = 1/N$, $n = 1, \ldots, N$
  for $t = 1 : T$
      1. Train the 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 $h_t$:
         $$\epsilon_t = \sum_{n=1}^N d_n^{(t)} \mathbf{I}(y_n \neq h_t(\mathbf{x}_n)) \ , \qquad (1)$$
         where $\mathbf{I}(\cdot)$ is the indicator function.
      3. Compute the combination coefficient:
         $$\tilde{\alpha}_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) \ , \qquad (2)$$
      4. Update weights for $n = 1, \ldots, N$:
         $$d_n^{(t+1)} = d_n^{(t)} \exp\left(-\tilde{\alpha}_t y_n h_t(\mathbf{x}_n)\right) / C_t \qquad (3)$$
         where $C_t$ is the normalization constant,
         such that $\sum_{n=1}^N d_n^{(t+1)} = 1$.
  end
Output : $F(\mathbf{x}) = \sum_{t=1}^T \tilde{\alpha}_t h_t(\mathbf{x})$
```

Fig. 1. Pseudo code of AdaBoost.

where $\rho(\mathbf{x}_n) \triangleq y_n f(\mathbf{x}_n)$ denotes the margin of sample $\mathbf{x}_n$ with respect to $f(\mathbf{x}_n)$. At the tth iteration, the negative functional derivative of G at F_{t-1} is given by

$$-\nabla G(F_{t-1})(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \neq \mathbf{x}_n \\ \frac{1}{N} y_n \exp(-y_n F_{t-1}(\mathbf{x}_n)), & \text{if } \mathbf{x} = \mathbf{x}_n, n = 1, \ldots, N. \end{cases} \qquad (5)$$

Equation (5) represents the direction, in which the cost function most rapidly decreases. Since the choice of the new h_t is constrained to $\mathcal{H}$, it may not be possible to choose $h_t = -\nabla G(F_{t-1})(\mathbf{x})$.[11] Instead, the search for h_t is conducted such that the inner product given by

$$\begin{aligned} \langle -\nabla G, h_t \rangle &= \frac{1}{N} \sum_{n=1}^N \exp(-y_n F_{t-1}(\mathbf{x}_n)) y_n h_t(\mathbf{x}_n) \ , \\ &= \frac{\sum_{i=1}^N \exp(-y_n F_{t-1}(\mathbf{x}_n))}{N} \sum_{n=1}^N \frac{\exp(-y_n F_{t-1}(\mathbf{x}_n))}{\sum_{i=1}^N \exp(-y_i F_{t-1}(\mathbf{x}_i))} y_n h_t(\mathbf{x}_n), \end{aligned} \qquad (6)$$

is maximized.[9] By unravelling Eq. (3) in Fig. 1, we get

$$d_n^{(t)} = d_n^{(t-1)} \exp(-\tilde{\alpha}_{t-1} y_n h_{t-1}(\mathbf{x}_n))/C_{t-1} = \frac{\exp(-y_n F_{t-1}(\mathbf{x}_n))}{\sum_{i=1}^N \exp(-y_i F_{t-1}(\mathbf{x}_i))}. \tag{7}$$

From Eqs. (6) and (7), it immediately follows that $h_t(\mathbf{x})$ is chosen to minimize the weighted error in Eq. (1). After $h_t(\mathbf{x})$ is selected, coefficient $\tilde{\alpha}_t$ can be found by a line search to minimize the intermediate cost function:

$$G^{(t)} = \frac{1}{N} \sum_{n=1}^N \exp\left(-y_n \left(\sum_{i=1}^{t-1} \tilde{\alpha}_i h_i(\mathbf{x}_n) + \tilde{\alpha}_t h_t(\mathbf{x}_n) \right) \right). \tag{8}$$

In the binary classification case, i.e. $\mathcal{H} = \{h(\mathbf{x}) : \mathbf{x} \to \pm 1\}$, $\tilde{\alpha}_t$ can be computed analytically as a solution to $\partial G_t / \partial \tilde{\alpha}_t = 0$, which is equal to the closed form in Eq. (2).

It has been empirically observed that AdaBoost can effectively increase the margin.[20] For this reason, since the invention of AdaBoost, it has been conjectured that AdaBoost, in the limit (i.e. $t \to \infty$), solves the following linear programming (LP) problem:

$$\begin{aligned} &\max \quad \rho, \\ &\text{s.t.} \quad \rho(\mathbf{x}_n) \geq \rho, n = 1, \ldots, N, \end{aligned} \tag{9}$$

where the margin is directly maximized. In the recent paper, however, the equivalence of the two algorithms has been proven not to hold always.[18] Nevertheless, these two algorithms are closely connected in the sense that both algorithms try to maximize the margin. This observation motivates researchers to design new ensemble classifiers either directly in the mathematical optimization setting,[4,10] or by borrowing ideas from the optimization setting, and introducing them in boosting.[14,16,17] Thereby, some of the well-studied optimization techniques can be utilized as novel boosting techniques.

3. Regularized AdaBoost

We begin the derivation of our regularization scheme by investigating the minimax problem. The connection between the well-known minimax problem,[23] and AdaBoost was first noted by Breiman,[1] and Freund and Schapire.[7] They determined the maximum achievable margin, given a hypothesis class, by exploiting the duality relationships in linear programming. For the time being, we assume that the cardinality of the set of hypothesis functions, $\mathcal{H}$, is finite and equal to T. We define a gain matrix, $\mathbf{Z}$, where $z_{nt} = y_n h_t(\mathbf{x}_n)$ is the margin of sample $\mathbf{x}_n$ with respect to the tth hypothesis function h_t. Let us, now, examine the following minimax optimization problem:

$$\max_{\boldsymbol{\alpha} \in \Gamma^T} \min_{\mathbf{d} \in \Gamma^N} \mathbf{d}^{\mathrm{T}} \mathbf{Z} \boldsymbol{\alpha} \tag{10}$$

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 in Eq. (10) can be interpreted as finding a set of combination coefficients $\boldsymbol{\alpha}$, such that the performance of the ensemble classifier in the worst case is optimized. It is straightforward to show that this optimization scheme leads to the maximum margin scheme in Eq. (9).

Generally speaking, a large margin is usually conducive to good generalization; however, for noisy data, where the data samples are highly overlapped and/or a few samples are mislabeled, the maximum margin scheme can be easily misled by outliers, yielding a classifier with suboptimal performance. Note that in Eq. (10), the minimization takes place over the entire probability space of the data distribution, which is not sufficiently restrictive. A natural strategy, therefore, is to constrain the data distribution, or add a penalty term to the cost function to control the skewness of the data distribution. Thereby, the algorithm will not be allowed to waste all of its resources dealing with a few hard-to-learn samples. Below, we present three regularized AdaBoost algorithms that fall into this framework.

3.1. *LP*$_{\mathbf{reg}}$*-AdaBoost*

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

$$\max_{\boldsymbol{\alpha}\in\Gamma^T} \min_{\{\mathbf{d}\in\Gamma^N, \mathbf{d}\leq\mathbf{c}\}} \mathbf{d}^{\mathrm{T}}\mathbf{Z}\boldsymbol{\alpha}, \tag{11}$$

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 optimization scheme in Eq. (11) can be understood as 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 Eq. (11) is

$$\begin{aligned} &\max_{(\rho,\boldsymbol{\lambda},\boldsymbol{\alpha})} \quad \rho - \sum_{n=1}^{N} c_n \lambda_n, \\ &\text{subject to} \sum_{t=1}^{T} \alpha_t z_{nt} \geq \rho - \lambda_n, n = 1, \ldots, N, \\ &\qquad\qquad \lambda_n \geq 0, n = 1, \ldots, N, \boldsymbol{\alpha} \in \Gamma^T. \end{aligned} \tag{12}$$

LP$_{\text{reg}}$-AdaBoost is a special case of Eq. (12) obtained by setting $c_1 = c_2 = \cdots = c_N = C$.[16] A similar scheme is also used in Support Vector Machine for nonseparable data cases.[2] The optimization scheme in Eq. (12) introduces a nonnegative slack variable λ_n into the optimization problem to achieve the soft margin, $\rho_s(\mathbf{x}_n)$, of pattern $\mathbf{x}_n$, defined as

$$\rho_s(\mathbf{x}_n) = \rho(\mathbf{x}_n) + \lambda_n. \tag{13}$$

The relaxation of the hard margin allows some patterns to have a smaller margin than ρ. Consequently, the algorithm does not classify all of the patterns according to their associated class labels.

The dual of Eq. (12) is given by

$$\begin{aligned} &\min_{(\gamma,\mathbf{d})} \quad \gamma, \\ &\text{subject to} \sum_{n=1}^{N} d_n z_{nt} \le \gamma, \quad t = 1, \ldots, T, \\ &\qquad\qquad \mathbf{d} \le \mathbf{c}, \mathbf{d} \in \Gamma^N. \end{aligned} \tag{14}$$

By working directly in the dual domain, we lose the clarity of pursuing regularization through the margin concept. Yet, the dual domain proves advantageous, since the primal domain is not suitable for specifying the soft margin, except in the case defined in Eq. (13).

For convenience, we reformulate Eq. (11) as

$$\max_{\boldsymbol{\alpha}\in\Gamma^T} \min_{\mathbf{d}\in\Gamma^N} \mathbf{d}^T \mathbf{Z} \boldsymbol{\alpha} + \beta(\|\mathbf{d}\|_\infty), \tag{15}$$

where $\|\cdot\|_p$ is the p-norm, and $\beta(P)$ is a function defined by

$$\beta(P) = \begin{cases} 0, & \text{if } P \le C, \\ \infty, & \text{if } P > C. \end{cases} \tag{16}$$

Note that the box defined by $\{\mathbf{d} : \|\mathbf{d}\|_\infty \le C, \mathbf{d} \in \Gamma^N\}$ is centered at the distribution center $\mathbf{d}_0 = [1/N, \ldots, 1/N]$. Also, the parameter C reflects to some extent the distribution skewness between the box boundary and $\mathbf{d}_0$. Equation (15) indicates that LP_{reg}-AdaBoost is a penalty scheme with a zero penalty within the box, and infinity, outside the box. In this sense, this scheme is somewhat heuristic and may be too restrictive.

With respect to the implementation of LP_{reg}-AdaBoost, we note that in practice the cardinality of $\mathcal{H}$ can be infinite. Consequently, the gain matrix $\mathbf{Z}$ may 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 ν-Arc,[17] and C-Barrier algorithms.[14]

In the following sections, we use $|\mathcal{H}|$ to denote the cardinality of the hypothesis function set, and reserve T as the number of iteration steps in AdaBoost.

3.2. *AdaBoost*$_{\mathbf{KL}}$

Motivated by Eq. (15), one plausible strategy to control the skewness of the data distribution is to add a penalty term, $P(\mathbf{d})$, to the cost function in Eq. (10). The penalty can be defined as a function of the distance between query distributions $\mathbf{d}$ and distribution center $\mathbf{d}_0$. This leads to the following optimization problem:

$$\max_{\boldsymbol{\alpha}\in\Gamma^{|\mathcal{H}|}} \min_{\mathbf{d}\in\Gamma^N} \mathbf{d}^T \mathbf{Z} \boldsymbol{\alpha} + \beta P(\mathbf{d}), \tag{17}$$

where $\beta > 0$ is a predefined parameter controlling the penalty strength. With a mild assumption that $P(\mathbf{d})$ is a convex function of $\mathbf{d}$, it can be shown that Eq. (17) is equivalent to (Generalized Minimax Theorem[6]):

$$\begin{aligned}&\min_{(\gamma,\mathbf{d}\in\Gamma^N)}\ \gamma + \beta P(\mathbf{d}),\\ &\text{subject to}\quad \sum_{n=1}^{N} d_n z_{nj} \le \gamma, \quad j = 1,\ldots,|\mathcal{H}|.\end{aligned} \tag{18}$$

We refer to the formulation in Eq. (18) as regularized scheme in the dual domain.

One commonly used distance metric for two discrete distributions is the Kullback–Leibler (KL) divergence.[3] In our case, we have

$$\mathrm{KL}(\mathbf{d},\mathbf{d}_0) = \sum_{n=1}^{N} d_n \ln\frac{d_n}{1/N}. \tag{19}$$

$\mathrm{KL}(\mathbf{d},\mathbf{d}_0)$ is convex over the region $\mathbf{d} > 0$, because its Hessian matrix is positive definite.

By substituting Eq. (19) into Eq. (18), we derive

$$\begin{aligned}&\min_{(\gamma,\mathbf{d}\in\Gamma^N)}\ \gamma + \beta\sum_{n=1}^{N} d_n \ln\frac{d_n}{1/N},\\ &\text{subject to}\quad \sum_{n=1}^{N} d_n z_{nj} \le \gamma, \quad j = 1,\ldots,|\mathcal{H}|,\end{aligned} \tag{20}$$

which can be reformulated as

$$\begin{aligned}&\min_{(\gamma,\mathbf{d}\in\Gamma^N)}\ \gamma,\\ &\text{subject to}\quad \sum_{n=1}^{N} d_n z_{nj} + \beta\sum_{n=1}^{N} d_n \ln\frac{d_n}{1/N} \le \gamma, \quad j = 1,\cdots,|\mathcal{H}|.\end{aligned} \tag{21}$$

The above optimization problem is illustrated in Fig. 4(a) ($|\mathcal{H}| = 2$). To facilitate the following discussions, we introduce the following auxiliary terms:

$$s_j(\mathbf{d}) = \sum_{n=1}^{N} d_n z_{nj} + \beta\sum_{n=1}^{N} d_n \ln\frac{d_n}{1/N}, \tag{22}$$

$$s(\mathbf{d}) = \max_{1\le j\le|\mathcal{H}|} s_j(\mathbf{d}). \tag{23}$$

Note that $s(\mathbf{d})$ is also a convex function.

Suppose now we are given a set of query distributions $\{\mathbf{d}^{(t)}\}_{t=1}^{T}$. For each query distribution $\mathbf{d}^{(t)}$, we can define a supporting hyperplane to the epigraph of $s(\mathbf{d})$, given by

$$\gamma = s(\mathbf{d}^{(t)}) + \partial s(\mathbf{d}^{(t)})(\mathbf{d} - \mathbf{d}^{(t)}), \tag{24}$$

Due to the convexity of $s(\mathbf{d})$, a supporting hyperplane gives an underestimate of s. More precisely, Eq. (24) can be written as

$$\begin{aligned}\gamma &= \max_{1\le j\le|\mathcal{H}|} s_j(\mathbf{d}^{(t)}) + \partial s(\mathbf{d}^{(t)})(\mathbf{d}-\mathbf{d}^{(t)}),\\ &= \mathbf{z}_{.t}^{\mathrm{T}}\mathbf{d}^{(t)} + \beta\sum_{n=1}^{N} d_n^{(t)}\ln\frac{d_n^{(t)}}{1/N} + \left(\mathbf{z}_{.t}+\beta\begin{bmatrix}\ln\frac{d_1^{(t)}}{1/N}+1\\ \vdots\\ \ln\frac{d_N^{(t)}}{1/N}+1\end{bmatrix}\right)^{\mathrm{T}}(\mathbf{d}-\mathbf{d}^{(t)}),\\ &= \left(\mathbf{z}_{.t}+\beta\ln\frac{\mathbf{d}^{(t)}}{1/N}\right)^{\mathrm{T}}\mathbf{d},\end{aligned} \tag{25}$$

where

$$\mathbf{z}_{.t} = [y_1h_t(\mathbf{x}_1),\ldots,y_Nh_t(\mathbf{x}_N)]^{\mathrm{T}}, \tag{26}$$

and

$$h_t = \arg\max_{h\in\mathcal{H}}\sum_{n=1}^{N} d_n^{(t)}h(\mathbf{x}_n)y_n. \tag{27}$$

Let us, now, define:

$$\widetilde{\mathbf{Z}} = \mathbf{Z} + \beta\left[\ln\frac{\mathbf{d}^{(1)}}{1/N},\ldots,\ln\frac{\mathbf{d}^{(T)}}{1/N}\right], \tag{28}$$

whose tth column reads $\tilde{\mathbf{z}}_{.t} = \mathbf{z}_{.t} + \beta\ln\frac{\mathbf{d}^{(t)}}{1/N}$, where $\mathbf{z}_{.t}$ is given by Eq. (26). Note that $\widetilde{\mathbf{Z}}$ can be interpreted as a new gain matrix. This means that adding a penalty function to Eq. (10) results in a modification of the gain matrix that encodes the distribution information into the hypothesis decisions. By using Eq. (28), the optimization problem in Eq. (21) can be approximated as

$$\begin{aligned}&\min\nolimits_{(\gamma,\mathbf{d}\in\Gamma^N)}\ \gamma,\\ &\text{subject to}\quad \tilde{\mathbf{z}}_{.t}^{\mathrm{T}}\mathbf{d}\le\gamma,\quad t=1,\ldots,T.\end{aligned} \tag{29}$$

Equation (29) represents a linear programming problem that is much easier to deal with than the original one in Eq. (21). However, this is only a linear approximation that becomes better as more constraints are added. The above linear approximation process is illustrated in Fig. 4(b).

The only remaining problem to be considered is the generation of the query distributions. The distributions can be obtained by using a standard technique called column generation.[13] However, there are several drawbacks associated with the column generation approach. Usually, it exhibits slow convergence due to the degeneracy of Eq. (29). Moreover, a highly efficient LP solver is needed to compute

Eq. (29) iteratively. Therefore, to find $\mathbf{d}$, we use another strategy in which we change the domain of our optimization problem, by deriving the dual form of Eq. (29) as

$$\begin{aligned}&\max\nolimits_{(\rho,\boldsymbol{\alpha}\in\Gamma^T)}\ \rho,\\ &\text{subject to}\quad \sum_{t=1}^{T}\alpha_t z_{nt}+\beta\sum_{t=1}^{T}\alpha_t\ln\frac{d_n^{(t)}}{1/N}\geq\rho,\quad n=1,\ldots,N.\end{aligned}\tag{30}$$

The above formulation gives rise to a new definition of the *soft margin* of pattern $\mathbf{x}_n$, which can be defined as

$$\rho_s(\mathbf{x}_n)=\sum_{t=1}^{T}\alpha_t z_{nt}+\beta\sum_{t=1}^{T}\alpha_t\ln\frac{d_n^{(t)}}{1/N}.\tag{31}$$

Here, the term $\beta\sum_{t=1}^{T}\alpha_t\ln\frac{d_n^{(t)}}{1/N}$ can be interpreted as "mistrust" in data samples. Note that the mistrust is calculated with respect to the initial uniform distribution $\mathbf{d}^{(1)}=[1/N,\ldots,1/N]$. This implies that if, for example, for all query distributions of $\mathbf{x}_n$, $d_n^{(t)}\leq 1/N$, $t=1,\ldots,T$, then the mistrust can take negative values. As a result, the soft margin provides the mechanism to penalize difficult-to-learn samples, and at the same time to award easy-to-learn samples. It has been 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.[16,19] Therefore, by defining the soft margin as in Eq. (31), we seek to reverse the AdaBoost process to some extent, the strength of which is controlled by β.

The concept of soft margin allows us to formulate a novel regularized AdaBoost algorithm, which we refer to as AdaBoost$_{\text{KL}}$. Recall that AdaBoost can be viewed as an algorithm performing stage-wise gradient descent of the cost function defined in Eq. (4). In light of the relationship between AdaBoost and LP, we use the LP formulation in Eq. (30) to define a new cost function, G_{KL}, in the domain of the soft margin:

$$\begin{aligned}G_{\text{KL}}&=\sum_{n=1}^{N}\exp\left\{-\rho_s(\mathbf{x}_n)\sum_t\tilde{\alpha}_t\right\},\\ &=\sum_{n=1}^{N}\exp\left\{-\left[\sum_t\alpha_t z_{nt}+\beta\sum_t\alpha_t\ln\frac{d_n^{(t)}}{1/N}\right]\sum_t\tilde{\alpha}_t\right\}.\end{aligned}\tag{32}$$

To minimize the cost function, in each iteration step t, we first find h_t as the one minimizing the weighted training error, and then calculate the combination coefficient $\tilde{\alpha}_t$ as

$$\begin{aligned}\tilde{\alpha}_t&=\arg\min_{\tilde{\alpha}_t\geq0}G_{\text{KL}}^{(t)}\\ &=\arg\min_{\tilde{\alpha}_t\geq0}\sum_{n=1}^{N}\exp\left\{-\left[\sum_{j=1}^{t}\alpha_j z_{nj}+\beta\sum_{j=1}^{t}\alpha_j\ln\frac{d_n^{(j)}}{1/N}\right]\sum_{j=1}^{t}\tilde{\alpha}_j\right\}.\end{aligned}\tag{33}$$

```
AdaBoost$_{\mathrm{KL}}$
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, maximum number of iteration steps $T$, $d_n^{(1)} = 1/N$, $n = 1, \cdots, N$, parameter $\beta$.
  for $t = 1 : T$
      1. Train the 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
         $\tilde{\alpha}_t = \arg\min_{\tilde{\alpha}_t \geq 0} \sum_{n=1}^N \exp\left\{-\left[\sum_{j=1}^t \alpha_j z_{nj} + \beta \sum_{j=1}^t \alpha_j \ln \frac{d_n^{(j)}}{1/N}\right] \sum_{j=1}^t \tilde{\alpha}_j\right\}$.
      4. Update weights:
         $d_n^{(t+1)} = \frac{d_n^{(t)}}{C_t} \exp\left\{-\tilde{\alpha}_t h_t(\mathbf{x}_n) y_n - \beta \tilde{\alpha}_t \ln \frac{d_n^{(t)}}{1/N}\right\}$,
         where $C_t$ is the normalization constant, such that $\sum_{n=1}^N d_n^{(t+1)} = 1$.
  end
Output : $F(\mathbf{x}) = \sum_{t=1}^T \tilde{\alpha}_t h_t(\mathbf{x})$.
```

Fig. 2. Pseudo code of AdaBoost$_{\mathrm{KL}}$.

It is difficult to compute $\tilde{\alpha}_t$ analytically from Eq. (33). Therefore, we resort to an iterative line search. The line search, in this case, is very efficient, because $\partial^2 G_{\mathrm{KL}}^{(t)} / \partial^2 \tilde{\alpha}_t \geq 0$. Further, similar to the derivation steps in prior work,[9,16] to update $d_n^{(t+1)}$, we find the derivative of $G_{\mathrm{KL}}^{(t)}$ with respect to $\rho_s(\mathbf{x}_n)$ as

$$d_n^{(t+1)} = \frac{\partial G_{\mathrm{KL}} / \partial \rho_s(\mathbf{x}_n)}{\sum_j \partial G_{\mathrm{KL}} / \partial \rho_s(\mathbf{x}_j)} = \frac{d_n^{(t)}}{C_t} \exp\left\{-\tilde{\alpha}_t h_t(\mathbf{x}_n) y_n - \beta \tilde{\alpha}_t \ln \frac{d_n^{(t)}}{1/N}\right\}, \tag{34}$$

where C_t is the normalization constant, such that $\sum_{n=1}^N d_n^{(t+1)} = 1$. The pseudo code of AdaBoost$_{\mathrm{KL}}$ is summarized in Fig. 2.

Note that for $\beta = 0$ AdaBoost$_{\mathrm{KL}}$ reduces to the original AdaBoost algorithm. Moreover, if $\beta \to \infty$, in Appendix we provide the proof that the ensemble classifier will only include the first hypothesis, h_1, that is, $\alpha_t = 0$, for $t \geq 2$, which corresponds to the single classifier design. It means that, by varying the values of parameter β, we are able to *control* the boosting strength of the learning process, mitigating the overfitting of AdaBoost.

3.3. *AdaBoost*$_{\mathbf{norm2}}$

As discussed in Introduction, by employing different convex penalty terms to the objective function of the minimax problem in Eq. (10), we can derive various types of the soft margin, resulting in different regularized AdaBoost algorithms. In this section, we consider the l_p norm, $\|\mathbf{d} - \mathbf{d}^{(1)}\|_p$, as the penalty function, which is a convex function of $\mathbf{d}$. More specifically, we focus only on the l_2 norm; however, generalization of the derivation steps below is straightforward.

Similar to the derivations in Sec. 3.2, from the optimization problem in Eq. (17), we obtain the following regularized scheme in the dual domain:

$$\begin{aligned} &\min_{(\gamma,\mathbf{d}\in\Gamma^N)}\ \gamma + \beta\|\mathbf{d} - \mathbf{d}_0\|_2, \\ &\text{subject to} \quad \sum_{n=1}^{N} d_n z_{nj} \le \gamma, \quad j = 1,\ldots,|\mathcal{H}| \end{aligned} \tag{35}$$

which can be linearly approximated as

$$\begin{aligned} &\min_{(\gamma,\mathbf{d}\in\Gamma^N)}\ \gamma, \\ &\text{subject to} \quad \tilde{\mathbf{z}}_{.t}^T\mathbf{d} \le \gamma, \quad t = 1,\ldots,T, \end{aligned} \tag{36}$$

where $\tilde{\mathbf{z}}_{.t}$ is the tth column of the new gain matrix, $\widetilde{\mathbf{Z}}$, defined as

$$\widetilde{\mathbf{Z}} = \mathbf{Z} + \beta\left[\frac{\mathbf{d}^{(1)} - \mathbf{d}_0}{\|\mathbf{d}^{(1)} - \mathbf{d}_0\|_2},\ldots,\frac{\mathbf{d}^{(T)} - \mathbf{d}_0}{\|\mathbf{d}^{(T)} - \mathbf{d}_0\|_2}\right]. \tag{37}$$

The dual form of Eq. (36) reads

$$\begin{aligned} &\max_{(\rho,\boldsymbol{\alpha}\in\Gamma^T)}\ \rho, \\ &\text{subject to} \quad \sum_{t=1}^{T}\alpha_t z_{nt} + \beta\sum_{t=1}^{T}\alpha_t\frac{d_n^{(t)} - 1/N}{\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2} \ge \rho, \quad n = 1,\ldots,N, \end{aligned} \tag{38}$$

which gives rise to the soft margin of pattern $\mathbf{x}_n$, $\rho_s(\mathbf{x}_n)$, defined as

$$\rho_s(\mathbf{x}_n) = \sum_{t=1}^{T}\alpha_t z_{nt} + \beta\sum_{t=1}^{T}\alpha_t\frac{d_n^{(t)} - 1/N}{\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2}. \tag{39}$$

Similar to the discussion in Sec. 3.2, $\beta\sum_{t=1}^{T}\alpha_t\frac{d_n^{(t)} - 1/N}{\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2}$ can be interpreted as “mistrust” in samples with respect to the center distribution. The term in the denominator, $\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2$, can be roughly understood as follows: the closer the query distribution to the center distribution, the more trust the outcome of the hypothesis deserves. Interestingly, the soft margin in Eq. (39) resembles that of AdaBoost$_{\mathrm{Reg}}$,[14] defined as

$$\rho_{\mathrm{Reg}}(\mathbf{x}_n) = \sum_{t=1}^{T}\alpha_t z_{nt} + \beta\sum_{t=1}^{T}\alpha_t d_n^{(t)}. \tag{40}$$

Obviously, from Eqs. (39) and (40), the main difference is that our soft margin is computed with respect to the center distribution.

Now, following the same strategy used in deriving AdaBoost$_{\mathrm{KL}}$, we reformulate the optimization problem in Eq. (38) into an AdaBoost-like algorithm, which we call AdaBoost$_{\mathrm{norm2}}$. To this end, we define a new cost function, G_{norm2}, as

$$G_{\mathrm{norm2}} = \sum_{n=1}^{N} \exp\left\{ -\left[\sum_t \alpha_t z_{nt} + \beta \sum_t \alpha_t \frac{d_n^{(t)} - 1/N}{\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2} \right] \sum_t \tilde{\alpha}_t \right\}. \tag{41}$$

To minimize the cost function, in each iteration step t, we first find h_t as the one minimizing the weighted training error, and then calculate the combination coefficient $\tilde{\alpha}_t$ as

$$\begin{aligned} \tilde{\alpha}_t &= \arg\min_{\tilde{\alpha}_t \geq 0} G_{\mathrm{norm2}}^{(t)} \\ &= \arg\min_{\tilde{\alpha}_t \geq 0} \sum_{n=1}^{N} \exp\left\{ -\left[\sum_{j=1}^{t} \alpha_j z_{nj} + \beta \sum_{j=1}^{t} \alpha_j \frac{d_n^{(j)} - 1/N}{\|\mathbf{d}^{(j)} - \mathbf{d}_0\|_2} \right] \sum_{j=1}^{t} \tilde{\alpha}_j \right\}. \end{aligned} \tag{42}$$

The updated distribution $d^{(t+1)}(\mathbf{x}_n)$ is computed as the derivative of G_{norm2} with respect to $\rho_s(\mathbf{x}_n)$

$$d_n^{(t+1)} = \frac{\partial G/\partial \rho_s(\mathbf{x}_n)}{\sum_j \partial G/\partial \rho_s(\mathbf{x}_j)} = \frac{d_n^{(t)}}{C_t} \exp\left\{ -\tilde{\alpha}_t h_t(\mathbf{x}_n) y_n - \beta \tilde{\alpha}_t \frac{d_n^{(t)} - 1/N}{\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2} \right\}, \tag{43}$$

where C_t is the normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$. The pseudo code of AdaBoost$_{\mathrm{norm2}}$ is summarized in Fig. 3.

3.4. *General framework*

In this subsection, we summarize the proposed regularization of boosting algorithms by constraining their data distributions $\mathbf{d}$. The general formulation of such regularization can be specified as the following optimization problem in the dual domain:

$$\begin{aligned} &\min\nolimits_{(\gamma, \mathbf{d}\Gamma^N)} \ \gamma + \beta(P)P(\mathbf{d}) \\ &\text{subject to } \ \mathbf{z}_{\cdot j}^T \mathbf{d} \leq \gamma, \quad j = 1, \ldots, |\mathcal{H}|\,, \end{aligned} \tag{44}$$

where $P(\mathbf{d})$ is a penalty function, and $\beta(P)$ is a function of $P(\mathbf{d})$.

Depending on the specification of $P(\mathbf{d})$ and $\beta(P)$, it is possible to derive a range of regularized boosting algorithms. For example, if $\beta(P) = \beta$ is a predefined constant, and $P(\mathbf{d})$ is the KL divergence, or Euclidean distance between $\mathbf{d}$ and the center distribution $\mathbf{d}_0$, we obtain AdaBoost$_{\mathrm{KL}}$ or AdaBoost$_{\mathrm{Norm2}}$, respectively. Then, if we specify $P(\mathbf{d}) = \|\mathbf{d}\|_\infty$, and $\beta(P) = 0$ if $P \leq C$, and $\beta(P) = \infty$ if $P > C$, we get LP$_{\mathrm{reg}}$-AdaBoost. We point out that this is a novel interpretation of LP$_{\mathrm{reg}}$-AdaBoost. Note that both $\nu-$Arc and C-Barrier algorithms, although being implemented in a different manner than the above algorithms, also fall into this category.

```
AdaBoost$_{\mathrm{norm2}}$
Initialization: $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, maximum number of iteration steps $T$, $d_n^{(1)} = 1/N$, $n = 1, \cdots, N$, parameter $\beta$.
  for $t = 1 : T$
      1. Train the 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
         $\tilde{\alpha}_t = \arg\min_{\tilde{\alpha}_t \geq 0} \sum_{n=1}^N \exp\left\{-\left[\sum_{j=1}^t \alpha_j z_{nj} + \beta \sum_{j=1}^t \alpha_j \frac{d_n^{(j)} - 1/N}{\|\mathbf{d}^{(j)} - \mathbf{d}_0\|_2}\right] \sum_{j=1}^t \tilde{\alpha}_j\right\}$,
      4. Update weights:
         $d_n^{(t+1)} = \frac{d_n^{(t)}}{C_t} \exp\left\{-\tilde{\alpha}_t h_t(\mathbf{x}_n) y_n - \beta \tilde{\alpha}_t \frac{d_n^{(t)} - 1/N}{\|\mathbf{d}^{(t)} - \mathbf{d}_0\|_2}\right\}$,
         where $C_t$ is the normalization constant, such that $\sum_{n=1}^N d_n^{(t+1)} = 1$.
  end
Output : $F(\mathbf{x}) = \sum_{t=1}^T \tilde{\alpha}_t h_t(\mathbf{x})$.
```

Fig. 3. Pseudo code of AdaBoost$_{\mathrm{norm2}}$.

3.5. *Analysis on AdaBoost*$_{\mathbf{KL}}$ *and AdaBoost*$_{\mathbf{Norm2}}$

In this section, we show that both AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$ perform the gradient descent on a cost function of the soft margin. Recall that adding two different penalty terms to the minimax problem in Eq. (10) yields two modified gain matrices in Eqs. (28) and (37), respectively. By expressing the elements of $\widetilde{\mathbf{Z}}$ as $\tilde{z}_{nt} = y_n \tilde{h}(\mathbf{x}_n; \mathbf{d}^{(t)})$, one can imagine that the two proposed algorithms operate in a new hypothesis space $\widetilde{\mathcal{H}}$, whose elements $\tilde{h}(\mathbf{x}; \mathbf{d})$ are defined as:

$$\begin{aligned} \tilde{h}(\mathbf{x}_n; \mathbf{d}) &= h^*(\mathbf{x}_n) + \beta y_n \ln \frac{d_n}{1/N}, && \text{for AdaBoost}_{\mathrm{KL}} \\ \tilde{h}(\mathbf{x}_n; \mathbf{d}) &= h^*(\mathbf{x}_n) + \beta y_n \frac{d_n - 1/N}{\|\mathbf{d} - \mathbf{d}_0\|_2}, && \text{for AdaBoost}_{\mathrm{norm2}} \end{aligned} \tag{45}$$

where $h^*(\mathbf{x}_n) = \arg\max_{h \in \mathcal{H}} \sum_{n=1}^N d_n h(\mathbf{x}_n) y_n$. Recall that the algorithms do not explicitly search for the optimal direction in $\widetilde{\mathcal{H}}$ space. Therefore, it is necessary to prove that the direction obtained in each iteration is indeed the one that maximally decreases the cost functions in Eqs. (32) and (41). Below, we present the proof.

We first consider AdaBoost$_{\mathrm{KL}}$. From Eq. (32), after the $(t-1)$th iteration, we have

$$G_{\mathrm{KL}}^{(t-1)} = \sum_{n=1}^{N} \exp\left\{-y_n \left[\sum_{j}^{t-1} \tilde{\alpha}_j \tilde{h}_j(\mathbf{x}_n)\right]\right\}. \tag{46}$$

In the tth iteration, the optimal direction $\tilde{h}_t(\mathbf{x}_n)$ in which the cost function most rapidly decreases, subject to $h_t(\mathbf{x}_n) \in \mathcal{H}$, is computed, such that the inner product

$$\langle -\nabla G_{KL}^{(t-1)}, \tilde{h}_t(\mathbf{x}_n)\rangle \propto \sum_{n=1}^{N} d_n^{(t)} y_n \tilde{h}_t(\mathbf{x}_n) = \sum_{n=1}^{N} d_n^{(t)} y_n h_t(\mathbf{x}_n) + \beta \sum_{n=1}^{N} d_n^{(t)} \ln \frac{d_n}{1/N} \tag{47}$$

is maximized, similar to the derivation in Sec. 2. It is straightforward to prove that Eq. (47) attains the maximum when $\tilde{h}_t(\mathbf{x}) = \tilde{h}(\mathbf{x}; \mathbf{d}^{(t)})$, that is, when $\mathbf{d} = \mathbf{d}^{(t)}$, which follows from the well-known properties of the KL divergence.[3]

In the case of AdaBoost$_{\mathrm{norm2}}$, by following the derivation steps in Sec. 2, we arrive at a similar expression to that in Eq. (47). To show that in the tth iteration the optimal direction $\tilde{h}_t(\mathbf{x}) = \tilde{h}(\mathbf{x}; \mathbf{d}^{(t)})$, we only need to show that $\sum_n d_n^{(t)} \frac{(d_n - 1/N)}{\|\mathbf{d}-\mathbf{d}_0\|_2}$ is maximized when $\mathbf{d} = \mathbf{d}^{(t)}$:

$$\frac{(\mathbf{d}-\mathbf{d}_0)^{\mathrm{T}}}{\|\mathbf{d}-\mathbf{d}_0\|_2}\mathbf{d}^{(t)} = \frac{(\mathbf{d}-\mathbf{d}_0)^{\mathrm{T}}(\mathbf{d}^{(t)}-\mathbf{d}_0)}{\|\mathbf{d}-\mathbf{d}_0\|_2} \le \frac{|(\mathbf{d}-\mathbf{d}_0)^{\mathrm{T}}(\mathbf{d}^{(t)}-\mathbf{d}_0)|}{\|\mathbf{d}-\mathbf{d}_0\|_2} \le \|\mathbf{d}^{(t)}-\mathbf{d}_0\|_2. \tag{48}$$

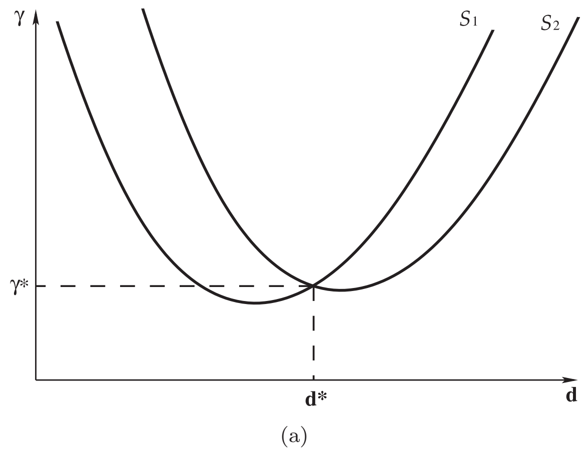
From the Cauchy–Schwartz inequality, the equality in Eq. (48) holds when $\mathbf{d} = \mathbf{d}^{(t)}$. This completes the proof.

In summary, AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$ perform the stage-wise gradient descent in the new hypothesis space $\widetilde{\mathcal{H}}$ instead of in $\mathcal{H}$. The new hypothesis space, $\widetilde{\mathcal{H}}$, encodes the information on the data distribution skewness with respect to the center distribution. From Eq. (45), it follows that for implementation of the two algorithms it is not necessary to explicitly construct $\widetilde{\mathcal{H}}$.

Considering our discussion in Sec. 3.1, one may also construct a similar new hypothesis space for AdaBoost$_{\mathrm{Reg}}$. However, AdaBoost$_{\mathrm{Reg}}$ cannot be explained as a gradient-descent algorithm in that space, because the direction the algorithm searches for in each iteration is not the one that maximally decreases the associated cost function.

4. Experimental Results

We report the results of a large scale experiment, where the proposed algorithms are compared with AdaBoost$_{\mathrm{Reg}}$, ν-Arc, C-Barrier, RBF (radial basis function), and SVM (RBF kernel). For fairness sake, our experimental setup is the same as the one used for evaluation of AdaBoost$_{\mathrm{Reg}}$ by Rätsch *et al.*[16] We use 13 artificial and real-world data sets originally from the UCI, DELVE and STATLOG benchmark repositions: *banana*, *breast cancer*, *diabetis*, *flare solar*, *german*, *heart*, *image*



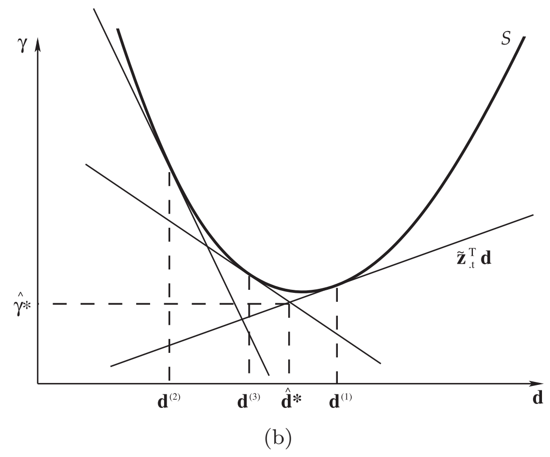


Fig. 4. (a) Illustration of the optimization problem in Eq. (21) in the case of $|\mathcal{H}| = 2$. $(\gamma*, \mathbf{d}^*)$ is the optimum solution; (b) Linear approximation of Eq. (21). $(\hat{\gamma}*, \hat{\mathbf{d}}^*)$ is obtained by solving Eq. (29), which is the approximate solution to the original problem.

ringnorm, *splice*, *thyroid*, *titanic*, *twonorm*, and *waveform*. Each data set has 100 realizations of training and testing data. For each realization, a classifier is trained and the test error is computed. The detailed information about the experimental setup and the benchmark data sets can also be found in Ref. 15.

The RBF net is used as the weak learner. All of the RBF parameters are the same as those used in Ref. 16. To avoid repeating the report on the numerous RBF parameters, for details we refer the reader to Ref. 16. We use cross-validation to estimate the optimal parameter β. The maximum number of iterations, T, is chosen to be 200.

Below, we present several tests, which illustrate the properties of the proposed algorithms. First, we show classification results on *banana* data set, whose samples are characterized by two features. In Fig. 5, we plot the decision boundaries of

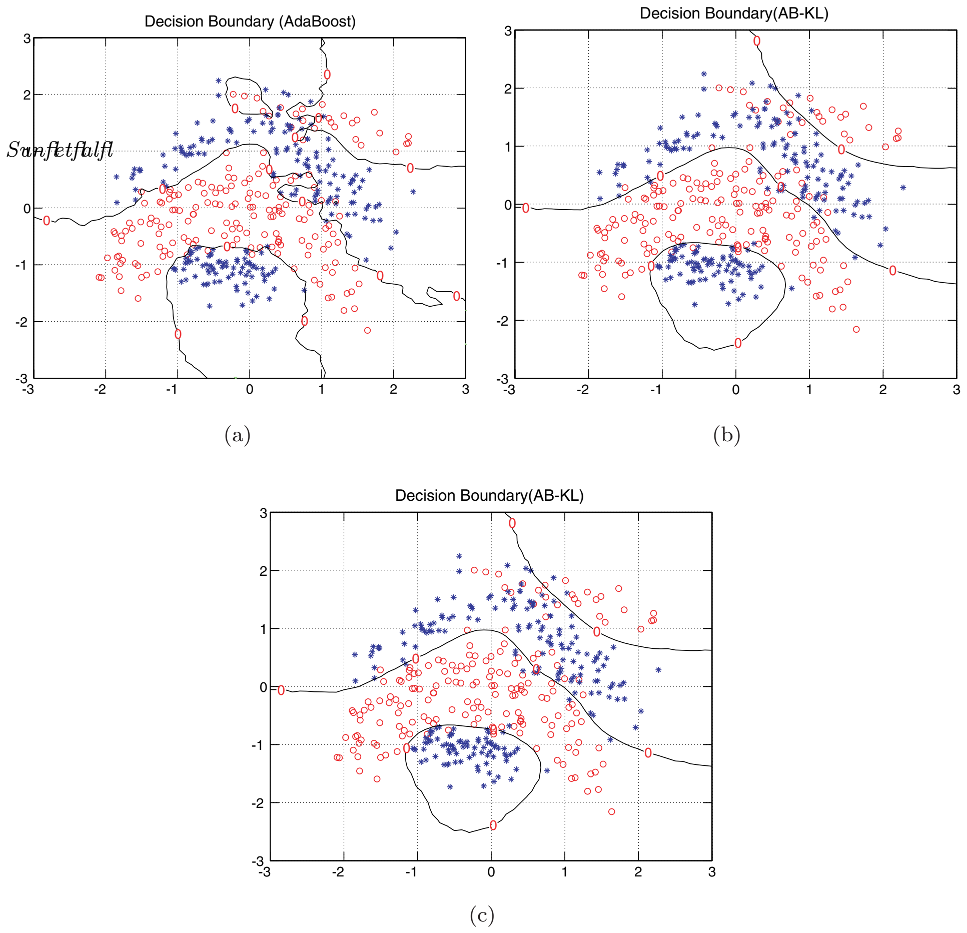


Fig. 5. The decision boundaries of three methods: AdaBoost, AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ based on one realization of the *banana* data. AdaBoost tries to classify each pattern according to its associated label and forms a zigzag decision boundary, which gives a straightforward illustration of the overfitting phenomenon of AdaBoost. Both AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ give smooth and similar decision boundaries.

AdaBoost, AdaBoost$_{\mathrm{norm2}}$, and AdaBoost$_{\mathrm{KL}}$ in the two-dimensional feature space. From the figure, it is obvious that AdaBoost tries to classify each pattern correctly according to its associated label, forming a zigzag shaped decision boundary, which indicates the overfitting of AdaBoost. In contrast, both AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ produce smooth decision boundaries by ignoring some hard-to-learn samples. Note that the boundaries of our algorithms are very similar.

Second, we present the classification results, and margin plots of three methods: AdaBoost, AdaBoost$_{\mathrm{norm2}}$, and AdaBoost$_{\mathrm{KL}}$, on one realization of the *waveform* data. From Figs. 6(a) and 6(b), we observe that AdaBoost tries to maximize the margin, thereby effectively reducing the training error to zero; however, it also

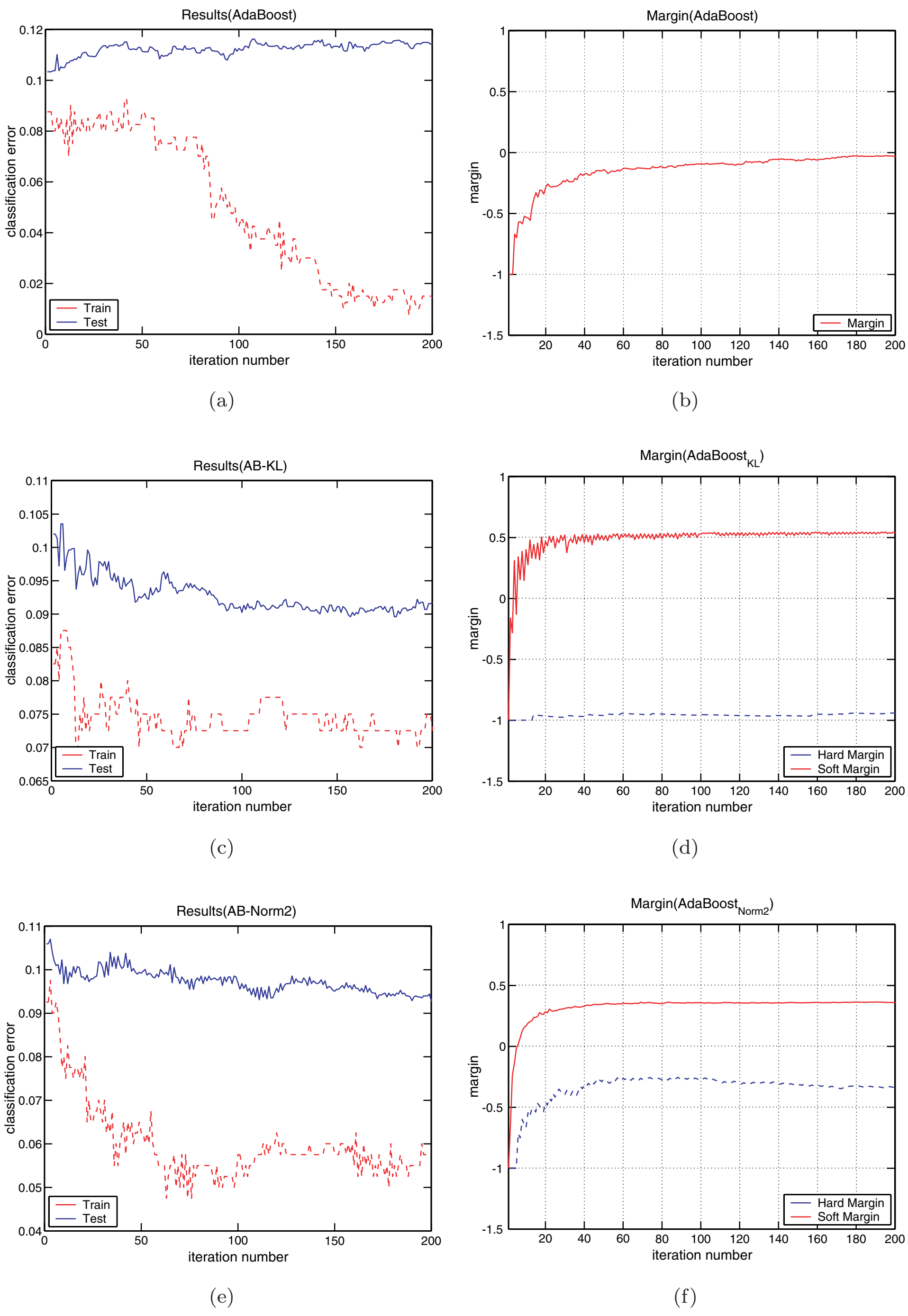


Fig. 6. Training and testing results, and margin plots of three methods: AdaBoost, AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ based on the *waveform* data. AdaBoost quickly leads to overfitting while the regularized methods effectively alleviate this problem.

Table 1. Classification errors and standard deviations of eight algorithms.

	RBF	AB	AB_R	AB_{KL}	AB_{norm2}	ν_{Arc}	C-Bar	SVM
waveform	10.7 ± 1.1	10.8 ± 0.6	9.8 ± 0.8	$\mathbf{9.4 \pm 0.6}$	$\mathit{9.5 \pm 0.4}$	10.0 ± 0.7	9.7 ± 0.5	9.9 ± 0.4
thyroid	4.5 ± 2.1	$\mathit{4.4 \pm 2.2}$	4.6 ± 2.2	$\mathbf{4.3 \pm 1.9}$	$\mathit{4.4 \pm 2.2}$	$\mathit{4.4 \pm 2.2}$	4.5 ± 2.2	4.8 ± 2.2
banana	10.8 ± 0.6	12.3 ± 0.7	10.9 ± 0.4	$\mathbf{10.7 \pm 0.4}$	$\mathbf{10.6 \pm 0.4}$	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	$\mathbf{25.8 \pm 4.6}$	$\mathit{25.9 \pm 4.4}$	26.0 ± 4.7
diabetis	24.3 ± 1.9	26.5 ± 2.3	23.8 ± 1.8	$\mathbf{23.5 \pm 1.8}$	$\mathbf{23.6 \pm 1.8}$	23.7 ± 2.0	23.7 ± 1.8	$\mathbf{23.5 \pm 1.7}$
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	$\mathbf{23.6 \pm 2.1}$
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	$\mathbf{16.0 \pm 3.3}$
ringnorm	1.7 ± 0.2	1.9 ± 0.3	1.6 ± 0.1	$\mathbf{1.5 \pm 0.1}$	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	$\mathbf{32.4 \pm 1.8}$
titanic	23.3 ± 1.3	22.6 ± 1.2	22.6 ± 1.2	$\mathit{22.5 \pm 0.9}$	$\mathit{22.5 \pm 1.2}$	23.0 ± 1.4	$\mathbf{22.4 \pm 1.1}$	$\mathbf{22.4 \pm 1.0}$
splice	10.0 ± 1.0	10.1 ± 0.5	9.5 ± 0.7	$\mathbf{9.2 \pm 0.6}$	9.5 ± 0.5	N/A	N/A	10.9 ± 0.7
image	3.3 ± 0.6	$\mathbf{2.7 \pm 0.7}$	$\mathbf{2.7 \pm 0.6}$	$\mathbf{2.7 \pm 0.6}$	$\mathbf{2.7 \pm 0.5}$	N/A	N/A	3.0 ± 0.6
twonorm	2.9 ± 0.3	3.0 ± 0.3	$\mathit{2.7 \pm 0.2}$	$\mathbf{2.6 \pm 0.2}$	$\mathit{2.7 \pm 0.2}$	N/A	N/A	3.0 ± 0.2

Table 2. 90% significant test comparing AdaBoost$_{\mathrm{KL}}$ with the other algorithms.

	AB$_{\mathrm{KL}}$/RBF	AB$_{\mathrm{KL}}$/AB	AB$_{\mathrm{KL}}$/AB_R	AB$_{\mathrm{KL}}$/ν_{Arc}	AB$_{\mathrm{KL}}$/C-Bar	AB$_{\mathrm{KL}}$/SVM
waveform	+	+	+	+	+	+
thyroid	0	0	0	0	0	+
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	−
titanic	+	0	0	0	0	0
splice	+	+	+	N/A	N/A	+
Image	+	0	0	N/A	N/A	+
twonorm	+	+	+	N/A	N/A	+

quickly leads to overfitting. It is reported that the simple *early stopping* method could alleviate the overfitting of AdaBoost. However, in this example (and many other examples on other data sets) we find that the early stopping method is not applicable. In contrast to AdaBoost, AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ try to maximize the soft margin, allowing a few hard-to-learn samples to have a small (even negative) sample margin. The two regularized algorithms effectively overcome the overfitting problem.

For a more comprehensive comparison, in Table 1, we provide the average classification results, with standard deviations, over the 100 realizations of the 13 data sets. The best results are marked in boldface, while the second best, in italics. By analyzing the results in Table 1, we conclude the following:

- AdaBoost performs worse than a single RBF classifier in almost all cases, due to the overfitting of AdaBoost. In ten out of thirteen cases AdaBoost$_{\mathrm{Reg}}$ performs significantly better than AdaBoost, and in ten cases AdaBoost$_{\mathrm{Reg}}$ outperforms a single RBF classifier.
- Except for *heart*, both AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ prove better than AdaBoost$_{\mathrm{Reg}}$.
- In comparison with ν-Arc and C-Barrier, our algorithms also perform better in most cases. This may be explained due to a hard limited penalty function used in the underlying optimization scheme of ν-Arc and C-Barrier.
- In almost all cases, the standard deviations of AdaBoost$_{\mathrm{norm2}}$ and AdaBoost$_{\mathrm{KL}}$ are smaller than those of the single RBF classifier and AdaBoost.
- The results for *ringnorm*, *thyroid* and *twonorm* suggest that the regularized AdaBoost algorithms are effective even in the low noise regime.

For a more rigorous comparison, a 90% significant test is reported in Table 2. In the table, '0' means the test accepts the null hypothesis: "no significant difference in average performance"; '+' denotes the test accepts the alternative hypothesis: "AdaBoost$_{\mathrm{KL}}$ is significantly better"; finally, '−' indicates: "AdaBoost$_{\mathrm{KL}}$

is significantly worse". For some data sets the performance differences between AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Reg}}$ are small (e.g. *titanic*). This is because AdaBoost$_{\mathrm{Reg}}$ is already a good classifier, which has been reported to be slightly better than Support Vector Machine (RBF kernel).[16] Nevertheless, significant improvements are observed for AdaBoost$_{\mathrm{KL}}$ in five datasets out of thirteen (Table 2).

5. Conclusions

In this paper, we have studied strategies to regularize AdaBoost, in order to reduce its overfitting, which has been reported to occur in high-noise regimes. By exploiting the connection between the minimax optimization problem, and the AdaBoost algorithm, we have explained that the impressive generalization capability of AdaBoost in low-noise settings may stem from the fact that the ensemble classifier tries to optimize the performance in the worst case. Due to this very mechanism, we speculate the overfitting of AdaBoost is inevitable in noisy data cases.

We have proposed to alleviate the problem by penalizing the data distribution skewness in the learning process. In this manner, a few outlier samples are prevented from spoiling decision boundaries. More specifically, to control the skewness, we have proposed to add a convex penalty function to the objective of the minimax problem. By means of the generalized minimax theorem, we have shown that the regularization scheme can be pursued equivalently in the dual domain, wherein we have specified the general framework of the proposed regularization. This general framework gives rise to a range of regularized boosting algorithms, differing in a particular specification of the penalty function. Thus, we have pointed out that LP$_{\mathrm{reg}}$-AdaBoost can be derived from the outlined framework if the penalty is defined as a hard-limited function, which represents a novel interpretation of the algorithm.

We have proposed to use two smooth convex penalty functions, one based on the KL divergence and the other on the Euclidean distance between the query and the center data distribution; thereby, we have derived two novel regularized algorithms AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{norm2}}$, respectively. We have proved that the proposed algorithms perform a stage-wise gradient-descent procedure on the cost function of the corresponding soft margin.

We have demonstrated the effectiveness of our algorithms by conducting experiments on a wide variety of data. In comparison with AdaBoost$_{\mathrm{Reg}}$, ν-Arc, and C-Barrier, our AdaBoost$_{\mathrm{KL}}$ and AdaBoost$_{\mathrm{Norm2}}$ achieve at least the same, or better classification performance.

Appendix A. Appendices

Lemma 1. *Suppose in each iteration the learning algorithm can find a hypothesis such that Eq.* (27) *holds. If $\beta \to \infty$, only the first hypothesis h_1 will be kept in AdaBoost$_{\mathrm{KL}}$, i.e. $\alpha_t = 0$, for $t \geq 2$.*

Proof. Suppose AdaBoost$_{\mathrm{KL}}$ found h_1 and the corresponding combination coefficient α_1. Also, suppose it found h_2, as well, and it is about to determine α_2 by a line search. The intermediate cost function is given by:

$$\begin{aligned} G_{\mathrm{KL}}^{(2)} &= \sum_{n=1}^{N} \exp(-F_1(\mathbf{x}_n)y_n) \exp\left\{-\alpha_2 h_2(\mathbf{x}_n)y_n - \alpha_2\beta \ln(d_n^{(2)}N)\right\} \\ &= \sum_{j=1}^{N} \exp\left(-F_1(\mathbf{x}_j)y_j\right) \sum_{n=1}^{N} d_n^{(2)} \exp\left\{-\alpha_2 h_2(\mathbf{x}_n)y_n - \alpha_2\beta \ln(d_n^{(2)}N)\right\}. \end{aligned}$$

α_2 can be computed by taking a derivative of $G_{\mathrm{KL}}^{(2)}$ with respect to α_2, and setting it to zero. For simplicity, we drop the constant terms.

$$\begin{aligned} &\partial G_{\mathrm{KL}}^{(2)}/\partial\alpha_2 \\ &\quad = \sum_{n=1}^{N} d_n^{(2)} \exp\left\{-\alpha_2 h_2(\mathbf{x}_n)y_n - \alpha_2\beta \ln(d_n^{(2)}N)\right\} \left\{-h_2(\mathbf{x}_n)y_n - \beta\ln(d_n^{(2)}N)\right\} \\ &\quad = \sum_{n=1}^{N} d_n^{(2)} \exp(-\alpha_2 h_2(\mathbf{x}_n)y_n)\,(d_n^{(2)}N)^{-\alpha_2\beta} \left\{-h_2(\mathbf{x}_n)y_n - \beta\ln(d_n^{(2)}N)\right\} = 0. \end{aligned}$$

By setting $\alpha_2 = 1/\beta$ and letting $\beta \to \infty$, we have:

$$\lim_{\beta\to\infty} \partial G_{\mathrm{KL}}^{(2)}/\partial\alpha_2 = \lim_{\beta\to\infty} \sum_{n=1}^{N} \frac{1}{N} \exp\left(-\frac{h_2(\mathbf{x}_n)y_n}{\beta}\right) \left\{-h_2(\mathbf{x}_n)y_n - \beta\ln(d_n^{(2)}N)\right\} > 0.$$

The last inequality follows from the fact that $\sum_{n=1}^{N} \frac{1}{N} \ln(d_n^{(2)}N) < 0$ for $\mathbf{d}^{(2)} \neq \mathbf{d}_0$. By setting $\alpha_2 = 1/\beta^2$ and letting $\beta \to \infty$, we have:

$$\begin{aligned} &\lim_{\beta\to\infty} \partial G_{\mathrm{KL}}^{(2)}/\partial\alpha_2 \\ &\quad = \lim_{\beta\to\infty} \sum_{n=1}^{N} (d_n^{(2)})^{1-1/\beta} N^{-1/\beta} \exp\left(-\frac{h_2(\mathbf{x}_n)y_n}{\beta^2}\right) \left\{-h_2(\mathbf{x}_n)y_n - \beta\ln(d_n^{(2)}N)\right\} < 0. \end{aligned}$$

The last inequality follows from the fact that $\sum_{n=1}^{N} d_n^{(2)} \ln(d_n^{(2)}N) > 0$ for $\mathbf{d}^{(2)} \neq \mathbf{d}_0$. Therefore, when $\beta \to \infty$, $\alpha_2 \in (\frac{1}{\beta^2}, \frac{1}{\beta}) \to 0$. □

References

1. L. Breiman, Prediction games and arcing algorithms, *Neural Comput.* **11** (1999) 1493–1517.
2. C. Cortes and V. Vapnik, Support vector networks, *Mach. Learn.* **20** (1995) 273–297.
3. T. M. Cover and J. A. Thomas, *Elements of Information Theory* (Wiley Interscience Press New York, 1991).
4. A. Demiriz, K. P. Bennett and J. Shawe-Taylor, Linear programming boosting via column generation, *Mach. Learn.* **46** (2002) 225–254.

5. T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization, *Mach. Learn.* **40** (2000) 139–157.
6. I. Ekeland and R. Temam, *Convex Analysis and Variational Problems* (North-Holland Pub. Co., Amsterdam, 1976)
7. Y. Freund and R. E. Schapire, Game theory, on-line prediction and boosting, *Proc. Ninth Annual Conf. Computational Learning Theory* (1996).
8. Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in *Eurp. Conf. Computational Learning Theory* (1995), pp. 23–37.
9. J. Friedman, Greedy function approximation: a gradient boosting machine, *Ann. Statis.* **29**(5) (2001) 1189–1232.
10. A. J. Grove and D. Schuurmans, Boosting in the limit: maximizing the margin of learned ensembles, *AAAI/IAAI* (1998), pp. 692–699.
11. L. Mason, J. Bartlett, P. Baxter and M. Frean, Functional gradient techniques for combining hypotheses, in *Advances in Large Margin Classifiers*, eds. B. Scholkopf, A. Smola, P. Bartlett and D. Schuurmans (2000).
12. R. Meir and G. Rätsch, An introduction to boosting and leveraging, in *Advanced Lectures on Machine Learning*, eds. S. Mendelson and A. Smola (Springer, 2003), pp. 119–184.
13. S. G. Nash and A. Sofer, *Linear and Nonlinear Programming* (McGraw-Hill, NY, USA, 1996), Chap. 7.4.
14. G. Rätsch, *Robust Boosting via Convex Optimization: Theory and Application*, Ph.D. thesis, University of Potsdam, Germany (October 2001).
15. G. Rätsch, IDA benchmark repository, World Wide Web, http://ida.first.fhg.de/projects/bench/benchmarks.htm (2001).
16. G. Rätsch, T. Onoda and K.-R. Müller, Soft margins for AdaBoost, *Mach. Learn.* **42** (2001) 287–320.
17. G. Rätsch, B. Scholkopf, A. Smola, S. Mika, T. Onoda and K.-R. Muller, Robust ensemble learning, in *Advances in Large Margin Classifiers*, eds. B. Scholkopf, A. Smola, P. Bartlett and D. Schuurmans (2000), pp. 207–220.
18. C. Rudin I. Daubechies and R. E. Schapire, The dynamics of adaboost: cyclic behavior and convergence of margins, *J. Mach. Learn. Res.* **5** (2004) 1557–1595.
19. 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 Int. Conf. Machine Learning* (1997), pp. 322–330.
20. R. E. Schapire, Y. Freund, P. Bartlett and W. S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, *Ann. Statis.* **26**(5)(1998) 1651–1686.
21. R. Schapire and Y. Singer, Improved boosting algorithms using confidence-rated predictions, *Mach. Learn.* **37**(3) (1999) 297–336.
22. V. Vapnik, *Statistical Learning Theory* (John Wiley and Sons Inc., New York, 1998).
23. J. von Neumann, Zur theorie der gesellschaftsspiele, *Math. Ann.* **100** (1928) 295–320.

Yijun Sun received the B.S. degrees in both electrical and mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1995, and the M.S. and Ph.D. degrees in electrical engineering from the University of Florida, Gainesville, USA, in 2003 and 2004, respectively. Since 2000, he has been a research assistant at the Department of Electrical and Computer Engineering at the University of Florida. Currently, he is a research scientist at the Interdisciplinary Center for Biotechnology Research at the University of Florida. He also holds a position of visiting assistant professor at the Department of Electrical and Computer Engineering at the same university.

His research interests include pattern recognition, machine learning, statistical signal processing, and their applications to target recognition and bioinformatics.

Sinisa Todorovic received his B.S. degree in electrical engineering at the University of Belgrade, Serbia, in 1994. From 1994 until 2001, he worked as a software engineer in the communications industry. He earned his M.S. and Ph.D. degrees at the University of Florida, in 2002, and 2005, respectively. Currently, he holds the position of postdoctoral research associate at Beckman Institute, University of Illinois at Urbana-Champaign.

His primary research interests encompass statistical image modeling for object recognition and image segmentation, machine learning, and multi-resolution image processing. He has published approximately 20 journal and refereed conference papers.

Jian Li received the M.Sc. and Ph.D. degrees in electrical engineering from the Ohio State University, Columbus, in 1987 and 1991, respectively.

From April 1991 to June 1991, she was an Adjunct Assistant Professor with the Department of Electrical Engineering, the Ohio State University, Columbus. From July 1991 to June 1993, she was an Assistant Professor with the Department of Electrical Engineering, University of Kentucky, Lexington. Since August 1993, she has been with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, where she is currently a Professor.

Her current research interests include spectral estimation, statistical and array signal processing, and their applications.

Dr. Li is a Fellow of IEEE and a Fellow of IEE. She is a member of Sigma Xi and Phi Kappa Phi. She received the 1994 National Science Foundation Young Investigator Award and the 1996 Office of Naval Research Young Investigator Award. She was an Executive Committee Member of the 2002 International Conference on Acoustics, Speech, and Signal Processing, Orlando, Florida, May 2002. She was an Associate Editor of the *IEEE Transactions on Signal Processing* from 1999 to 2005 and an Associate Editor of the *IEEE Signal Processing Magazine* from 2003 to 2005. She has been a member of the Editorial Board of Signal Processing, a publication of the European Association for Signal Processing (EURASIP), since 2005. She is presently a member of two of the IEEE Signal Processing Society technical committees: the Signal Processing Theory and Methods (SPTM) Technical Committee and the Sensor Array and Multichannel (SAM) Technical Committee.