Ensembles of Partially Trained SVMs with Multiplicative Updates

Ivor W. Tsang **James T. Kwok**
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
{ivor,jamesk}@cse.ust.hk

Abstract

The training of support vector machines (SVM) involves a quadratic programming problem, which is often optimized by a complicated numerical solver. In this paper, we propose a much simpler approach based on multiplicative updates. This idea was first explored in [Cristianini *et al.*, 1999], but its convergence is sensitive to a learning rate that has to be fixed manually. Moreover, the update rule only works for the hard-margin SVM, which is known to have poor performance on noisy data. In this paper, we show that the multiplicative update of SVM can be formulated as a Bregman projection problem, and the learning rate can then be adapted automatically. Moreover, because of the connection between boosting and Bregman distance, we show that this multiplicative update for SVM can be regarded as boosting the (weighted) Parzen window classifiers. Motivated by the success of boosting, we then consider the use of an adaptive ensemble of the partially trained SVMs. Extensive experiments show that the proposed multiplicative update rule with an adaptive learning rate leads to faster and more stable convergence. Moreover, the proposed ensemble has efficient training and comparable or even better accuracy than the best-tuned soft-margin SVM.

1 Introduction

Kernel methods, such as the support vector machines (SVMs), have been highly successful in many machine learning problems. Standard SVM training involves a quadratic programming (QP) problem, which is often solved by a complicated numerical solver. Moreover, while a general-purpose QP solver can be used, they are inefficient on this particular type of QPs. Consequently, a lot of specialized optimization techniques have been developed. The most popular approach is by using decomposition methods such as sequential minimization optimization (SMO) [Platt, 1999], which involves sophisticated working set selection strategies, advance caching schemes for the kernel matrix and its gradients, and also heuristics for stepsize prediction.

A much simpler approach is to use multiplicative updates, which was first explored in [Cristianini *et al.*, 1999]. However, its convergence is sensitive to a learning rate that has to be fixed manually. When the learning rate is too large, overshooting and oscillations occur; while when it is too small, convergence is slow.

A second problem with the multiplicative update rule is that it can only work with the hard-margin SVM. However, on noisy data, the hard-margin SVM has poor performance and the soft-margin SVM, which can trade off complexity with training error, is usually preferred. However, the choice of this tradeoff parameter, which can be from zero to infinity, is sometimes critical. Various procedures have been proposed for its determination, such as by using cross-validation or some generalization error bounds [Evgeniou *et al.*, 2004]. However, they are typically very computationally expensive.

Another possibility that avoids parameter tuning completely is by using the ensemble approach. [Kim *et al.*, 2003] combines multiple SVMs with different parameters, and obtains improved performance. However, as a lot of SVMs still have to be trained, this approach is also very expensive.

In this paper, we first address the learning rate problem by formulating the multiplicative update of SVM as a Bregman projection problem [Collins and Schapire, 2002]. It can then be shown that this learning rate can be adapted automatically based on the data. Moreover, because of the connection between boosting and Bregman distance [Collins and Schapire, 2002; Kivinen and Warmuth, 1999], we show that this multiplicative update can be regarded as a boosting algorithm in which the (weighted) Parzen window classifiers are the base hypotheses, and each base hypothesis added to the ensemble is a partially trained hard-margin SVM. Now, in the boosting literature, it is well-known that the whole ensemble performs better than a single base hypothesis. Hence, to address the possibly poor performance of the hard-margin SVM, we consider using the whole ensemble of partially trained SVMs for prediction. Note that this comes at little extra cost in the multiplicative updating procedure. Experimentally, this ensemble is observed to have comparable or even better accuracy than the best-tuned soft-margin SVM.

The rest of this paper is organized as follows. The learning rate problem in multiplicative update is addressed in Section 2. The connection between this update process and boosting, together with the proposed ensemble of partially trained hard-margin SVMs, is then discussed in Section 3. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

2 Multiplicative Updating Rule of SVM

In this paper, we focus on a particular variant of the hard-margin SVM, called ρ-SVM in the sequel, which will be introduced in Section 2.1. We then develop in Sections 2.2 and 2.3 an iterative multiplicative updating procedure based on the Bregman projection [Collins and Schapire, 2002; Kivinen and Warmuth, 1999]. The convergence of this iterative procedure is shown in Section 2.4.

2.1 Hard-Margin ρ-SVM

Given a training set $\{\mathbf{x}_i, y_i\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the ρ-SVM finds the plane $f(\mathbf{x}) = \mathbf{w}'\varphi(\mathbf{x})$ in the kernel-induced feature space (with feature map φ) that separates the two classes with maximum margin $\rho/\|\mathbf{w}\|$:

$$\max_{\mathbf{w},\rho} 2\rho - \|\mathbf{w}\|^2 \quad : \quad y_i\mathbf{w}'\varphi(\mathbf{x}_i) \geq \rho, \; i = 1, \ldots, m. \tag{1}$$

The corresponding dual is:

$$\min_{\boldsymbol{\alpha}} \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \quad : \quad \boldsymbol{\alpha} \geq \mathbf{0}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1, \tag{2}$$

where $\mathbf{0}$ and $\mathbf{1}$ are vectors of zero and one respectively, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]'$ is the vector of Lagrange multipliers, and $\tilde{\mathbf{K}} = [y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)]$. It can be easily shown that

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \varphi(\mathbf{x}_i), \text{ and } f(\mathbf{x}) = \sum_{i=1}^m \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}). \tag{3}$$

Because of the zero duality gap, the objectives in (1) and (2) are equal at optimality[1], i.e.,

$$\rho^* = \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^*. \tag{4}$$

Denote

$$u(\boldsymbol{\alpha}, \bar{\rho}) = \frac{1}{2}\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} - \bar{\rho}(\boldsymbol{\alpha}'\mathbf{1} - 1),$$

where $\bar{\rho}$ is the Lagrangian multiplier for the dual constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$. Moreover, the Karush-Kuhn-Tucker (KKT) conditions lead to:

$$\mathbf{0} \leq \boldsymbol{\alpha}^* \perp \left.\frac{\partial u(\boldsymbol{\alpha}, \bar{\rho}^*)}{\partial \boldsymbol{\alpha}}\right|_{\boldsymbol{\alpha}^*} \geq \mathbf{0}, \tag{5}$$

where $\perp$ denotes that the two vectors are orthogonal, and

$$\frac{\partial u(\boldsymbol{\alpha}, \bar{\rho})}{\partial \boldsymbol{\alpha}} = \tilde{\mathbf{K}}\boldsymbol{\alpha} - \bar{\rho}\mathbf{1} = [y_1 f(\mathbf{x}_1) - \bar{\rho}, \ldots, y_m f(\mathbf{x}_m) - \bar{\rho}]', \tag{6}$$

on using (3). Moreover, the optimal $\bar{\rho}$ (i.e., the dual variable of the dual) is indeed equal to the optimal primal variable ρ:

$$\bar{\rho}^* = \rho^*,$$

as $\bar{\rho}^* = \bar{\rho}^*\boldsymbol{\alpha}^{*\prime}\mathbf{1} = \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^* = \rho^*$ on using (4), (5) and (6).

[1]Here, variables at optimality are denoted by the superscript *.

2.2 Bregman Projection

In this section, we assume that we have access to the optimal value of ρ^*. We will return to the issue of determining ρ^* in Section 2.3.

As $\boldsymbol{\alpha} \in P_m = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha} \geq \mathbf{0}, \; \boldsymbol{\alpha}'\mathbf{1} = 1\}$, the m-dimensional probability simplex, a natural choice of the Bregman distance is the (unnormalized) relative entropy. At the t-th iteration, we minimize the Bregman distance between the new $\boldsymbol{\alpha}$ and the current estimate $\boldsymbol{\alpha}_t = [\alpha_1^{(t)}, \ldots, \alpha_m^{(t)}]'$, while requiring $\boldsymbol{\alpha}$ to satisfy $\boldsymbol{\alpha}'\frac{\partial u(\boldsymbol{\alpha}_t, \rho^*)}{\partial \boldsymbol{\alpha}} = \boldsymbol{\alpha}'(\tilde{\mathbf{K}}\boldsymbol{\alpha}_t - \rho^*\mathbf{1}) = 0$, which is similar to the constraint in (5). We then have the following entropy projection problem:

$$\min_{\boldsymbol{\alpha} \in P_m} \sum_{i=1}^m \left(\alpha_i \ln \frac{\alpha_i}{\alpha_i^{(t)}} - \alpha_i + \alpha_i^{(t)}\right) \; : \; \rho^* = \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t. \tag{7}$$

Introducing Lagrange multipliers for the constraints $\rho^* = \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t$ and $\boldsymbol{\alpha}'\mathbf{1} = 1$ (it will be shown that the remaining constraint $\boldsymbol{\alpha} \geq \mathbf{0}$ is inactive and so no Lagrange multiplier is needed) and on setting the derivative of the Lagrangian $\sum_{i=1}^m \alpha_i \ln \frac{\alpha_i}{\alpha_i^{(t)}} - \alpha_i + \alpha_i^{(t)} - \eta_t \left(\rho^* - \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t\right) - \lambda(\boldsymbol{\alpha}'\mathbf{1} - 1)$ w.r.t. $\boldsymbol{\alpha}$ to zero, we have $\alpha_i = \alpha_i^{(t)} \exp(-\eta_t y_i f_t(\mathbf{x}_i) + \lambda)$ on using (3). Substituting this back into $\boldsymbol{\alpha}'\mathbf{1} = 1$, we then obtain $\lambda = -\ln Z_t(\eta_t)$, where $Z_t(\eta_t) = \sum_{i=1}^m \alpha_i^{(t)} \exp(-\eta_t y_i f_t(\mathbf{x}_i))$, and thus

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} \exp(-\eta_t y_i f_t(\mathbf{x}_i))/Z_t(\eta_t). \tag{8}$$

Such an update is commonly known as multiplicative update [Cristianini *et al.*, 1999] or exponentiated gradient (EG) [Kivinen and Warmuth, 1997], with η_t playing the role of the learning rate. Notice that by initializing[2] $\boldsymbol{\alpha}_1 > \mathbf{0}$, then $\boldsymbol{\alpha}_t > \mathbf{0}$ in all subsequent iterations. Hence, as mentioned earlier, $\boldsymbol{\alpha} \geq \mathbf{0}$ in (7) is an inactive constraint. Finally, it can be shown that η_t can be obtained from the dual of (7), as:

$$\max_{\eta_t} \; -\ln(Z_t(\eta_t)\exp(\rho^*\eta_t)). \tag{9}$$

As mentioned in Section 1, a similar multiplicative updating algorithm for the standard hard-margin SVM is also explored in [Cristianini *et al.*, 1999]. However, the learning rate η there is not adaptive and the performance is sensitive to the manual choice of η. On the other hand, for our ρ-SVM variant, the value of η_t can be automatically determined from (9). The improved convergence properties of our adaptive scheme will be experimentally demonstrated in Section 4.1.

2.3 Estimating the Value of ρ^*

We now return to the question of estimating ρ^*. Recall that we initialize $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha}_1 = \frac{1}{m}\mathbf{1}$. From (2) and (4), ρ^* is equal to the optimal dual objective (which is to be minimized), and so the optimal $\boldsymbol{\alpha}^*$ should yield a smaller objective than that by $\boldsymbol{\alpha}_1$ (which is equal to $\bar{\rho}_1 = \frac{1}{m^2}\mathbf{1}'\tilde{\mathbf{K}}\mathbf{1}$). Moreover, from (4), we have $\rho^* \geq 0$ as $\mathbf{K} \succeq 0$. Hence, $\rho^* \in [0, \bar{\rho}_1]$.

In the following, we consider using a decreasing sequence of ρ_t's such that $\bar{\rho}_1 \geq \rho_t \geq \rho^*$. Consequently, the constraint

[2]In this paper, we initialize as $\boldsymbol{\alpha}_1 = \frac{1}{m}\mathbf{1}$.

in (7) has to be replaced by $\rho_t \geq \rho^* = \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t$, and the entropy projection problem becomes

$$\min_{\boldsymbol{\alpha}\in P_m} \sum_{i=1}^{m} \left(\alpha_i \ln \frac{\alpha_i}{\alpha_i^{(t)}} - \alpha_i + \alpha_i^{(t)}\right) \quad : \quad \rho_t \geq \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t. \quad (10)$$

Let $\bar{\rho}_t \equiv \boldsymbol{\alpha}_t'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t$. We first consider the feasibility of (10).

Proposition 1. *For any $1 > \epsilon_t > 0$, if $\bar{\rho}_t$ satisfies*

$$\bar{\rho}_t \frac{1-\epsilon_t}{1+\epsilon_t} \geq \rho^* > 0, \quad (11)$$

then there exists an $\boldsymbol{\alpha} \in P_m$ such that $\frac{\bar{\rho}_t}{1+\epsilon_t} \geq \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t$.

Proof. Since $\tilde{\mathbf{K}} \succeq 0$, we have $\mathbf{v}'\tilde{\mathbf{K}}\mathbf{v} \geq 0$ for any vector $\mathbf{v}$. Put $\mathbf{v} = \boldsymbol{\alpha}_t - \boldsymbol{\alpha}^*$, then

$$\boldsymbol{\alpha}_t'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t \geq 2\boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}_t - \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^*. \quad (12)$$

From (4), the condition

$$\bar{\rho}_t \frac{1-\epsilon_t}{1+\epsilon_t} \geq \rho^* \quad \Rightarrow \quad \boldsymbol{\alpha}_t'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t \frac{1-\epsilon_t}{1+\epsilon_t} \geq \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^*. \quad (13)$$

Summing (12) and (13), we have

$$\frac{\bar{\rho}_t}{1+\epsilon_t} \geq \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}_t, \quad (14)$$

and thus $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*$ satisfies the condition. $\square$

Hence, when $\bar{\rho}_t \frac{1-\epsilon_t}{1+\epsilon_t} \geq \rho^*$, a feasible solution of (10) exists. We set $\boldsymbol{\alpha}_{t+1}$ as the optimal $\boldsymbol{\alpha}$ of (10). Notice that (10) differs from (7) only in that the equality constraint in (7) now becomes an inequality constraint. Hence, optimization of (10) is almost exactly the same as that in Section 2.2, except that optimization of η_t now has an extra constraint $\eta \geq 0$.

Subsequently, we set $\bar{\rho}_{t+1} = \boldsymbol{\alpha}_{t+1}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_{t+1}$ and ρ_{t+1} according to

$$\rho_t = \frac{\bar{\rho}_t}{1+\epsilon_t}. \quad (15)$$

From the KKT condition of (10), $\eta_t(\boldsymbol{\alpha}_{t+1}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t - \rho_t) = 0$. This, together with[3] $\eta_t > 0$, leads to

$$\boldsymbol{\alpha}_{t+1}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t = \rho_t. \quad (16)$$

Using (3), then

$$\cos\omega_{t,t+1} = \frac{\boldsymbol{\alpha}_{t+1}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t}{\sqrt{\bar{\rho}_t}\sqrt{\bar{\rho}_{t+1}}} = \frac{1}{1+\epsilon_t}\sqrt{\frac{\bar{\rho}_t}{\bar{\rho}_{t+1}}}, \quad (17)$$

where $\omega_{t,t+1}$ is the angle between $\mathbf{w}_t$ and $\mathbf{w}_{t+1}$. Recall that this entropy projection procedure is used to solve the dual in (2). When (2) starts to converge, the angle $\omega_{t,t+1}$ tends to zero, and $\cos\omega_{t,t+1} \to 1$. As $\epsilon_t > 0$, if $\bar{\rho}_{t+1} > \bar{\rho}_t$, from (17), the optimization progress becomes deteriorated. This implies that the current ϵ_t is too large, and we have overshot and this solution is discarded. This is also the case when (10) is infeasible. Instead, we can relax the constraint in (10) by setting $\epsilon_t \leftarrow \epsilon_t/2$ to obtain a larger ρ_t. This is repeated until $\bar{\rho}_{t+1} \leq \bar{\rho}_t$. Note that we always have $\rho^* \leq \cdots \leq \bar{\rho}_{t+1} \leq \bar{\rho}_t$ as $\rho^* = \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^*$ and $\boldsymbol{\alpha}^*$ is optimal. Optimization of (10) then resumes with the new $\boldsymbol{\alpha}_{t+1}$ and ρ_{t+1}. The whole process iterates until ϵ_t is smaller than some termination threshold ϵ. The complete algorithm[4] is shown in Algorithm 1.

[3]When $\eta_t = 0$, no process can be made as we achieve the global optimal. Hence, we terminate the algorithm and $\boldsymbol{\alpha}_t = \boldsymbol{\alpha}^*$.

[4]In the experiments, we use $\epsilon = 0.005$ and initialize $\epsilon_1 = 0.1$.

Algorithm 1 Training the ρ-SVM.

```
1: Input: $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$, tolerance $\epsilon_1$ and $\epsilon$.
2: Initialize $t = 1$: $\alpha_i^{(1)} = 1/m$ for all $i = 1, \ldots, m$, and $\bar{\rho}_1 = \boldsymbol{\alpha}_1'\tilde{\mathbf{K}}\boldsymbol{\alpha}_1$.
3: Use (3) to compute $f_t(\mathbf{x}_i)$.
4: Set $\rho_t = \bar{\rho}_t/(1+\epsilon_t)$.
5: Set $\eta_t = \arg\min_{\eta\geq 0} Z_t(\eta)\exp(\rho_t\eta)$. If $\eta_t < 0$, then $\epsilon_t = \epsilon_t/2$, and goto Step 8 to test whether $\epsilon_t$ becomes too small; else if $\eta_t = 0$, then set $T = t - 1$ and terminate.
6: Update the Lagrangian multipliers $\alpha_i$'s using (8), as $\alpha_i^{(t+1)} = \alpha_i^{(t)} \exp(-\eta_t y_i f_t(\mathbf{x}_i))/Z_t(\eta_t)$, and compute $\bar{\rho}_{t+1} = \boldsymbol{\alpha}_{t+1}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_{t+1}$.
7: If $\bar{\rho}_t \geq \bar{\rho}_{t+1}$, then $\epsilon_{t+1} = \epsilon_t$ and $t \leftarrow t+1$ and goto Step 3; else $\epsilon_t = \epsilon_t/2$ and check whether $\epsilon_t < \epsilon$.
8: If $\epsilon_t \geq \epsilon$, then goto Step 4; else set $T = t - 1$ and terminate.
9: Output: $f_{\mathrm{SVM}}(\mathbf{x}) = f_{T+1}(\mathbf{x})$.
```

2.4 Convergence

In this section, we will design an auxiliary function, which is then used to lower bound the amount that the loss decreases at each iteration [Collins and Schapire, 2002]. Denote the relative entropy (which is our Bregman distance here) by $\Delta(\cdot,\cdot)$. From (14), the optimal $\boldsymbol{\alpha}^*$ solution of (2) is in the intersection of all the hyperplanes defined by the linear constraints $\rho_t \geq \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t$ for all t's. Using (14), (15) and (16), we have

$$\boldsymbol{\alpha}_{t+1}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t \geq \boldsymbol{\alpha}^{*\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}_t. \quad (18)$$

Now, the decrease in loss at two consecutive iterations is:

$$\begin{aligned} & \Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_t) - \Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_{t+1}) \\ = & \sum_{i=1}^{m} \alpha_i^* \left(\ln\alpha_i^{(t+1)} - \ln\alpha_i^{(t)}\right) \quad (\text{as } \boldsymbol{\alpha}_t'\mathbf{1} = \boldsymbol{\alpha}_{t+1}'\mathbf{1} = 1) \\ = & -\eta_t \sum_{i=1}^{m} \alpha_i^* y_i f_t(\mathbf{x}_i) - \ln Z_t(\eta_t) \quad (\text{using (8)}) \\ \geq & -\eta_t \sum_{i=1}^{m} \alpha_i^{(t+1)} y_i f_t(\mathbf{x}_i) - \ln Z_t(\eta_t) \quad (\text{using (18)}) \\ = & \sum_{i=1}^{m} \alpha_i^{(t+1)} \left(\ln\alpha_i^{(t+1)} - \ln\alpha_i^{(t)}\right), \quad (\text{using (8)}) \end{aligned}$$

or, for all t,

$$\Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_t) - \Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_{t+1}) \geq \Delta(\boldsymbol{\alpha}_{t+1}, \boldsymbol{\alpha}_t) \equiv A(\boldsymbol{\alpha}_t), \quad (19)$$

where $A(\cdot)$ is the auxiliary function. As $A(\cdot)$ never increases and is bounded below by zero (as $\Delta(\cdot,\cdot) \geq 0$), the sequence of $A(\boldsymbol{\alpha}_t)$'s converges to zero. Since $\boldsymbol{\alpha} \in P_m$ is compact, by the continuity of A and the uniqueness of $\boldsymbol{\alpha}^*$, the limit of this sequence of minimizers converges to the global optimum $\boldsymbol{\alpha}^*$.

Moreover, using (15) and (16), we have $\boldsymbol{\alpha}^{t+1\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^t = \frac{\bar{\rho}_t}{1+\epsilon_t}$. As $\boldsymbol{\alpha}^{t\prime}\tilde{\mathbf{K}}\boldsymbol{\alpha}^t = \bar{\rho}_t$, then $|(\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t+1})'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t| = \frac{\epsilon_t\bar{\rho}_t}{1+\epsilon_t}$. In the special case where $f_t(\mathbf{x}_i) \in [-1, 1]$, $\|\tilde{\mathbf{K}}\boldsymbol{\alpha}_t\|_\infty \leq 1$. Using the Hölder's inequality, we have $\|\boldsymbol{\alpha}_{t+1} - \boldsymbol{\alpha}_t\|_1 \geq \frac{\epsilon_t\bar{\rho}_t}{1+\epsilon_t}$.

We then apply the Pinsker's inequality [Fedotov *et al.*, 2001] and obtain

$$\Delta(\boldsymbol{\alpha}_{t+1}, \boldsymbol{\alpha}_t) > \frac{\epsilon_t^2 \bar{\rho}_t^2}{2(1+\epsilon_t)^2}.$$

Let $\nu = \min \frac{\epsilon_t \bar{\rho}_t}{1+\epsilon_t}$, and summing up (19) at each step, then

$$\begin{aligned} \Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_1) - \Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_{T+1}) &\geq \sum_{t=1}^{T} \Delta(\boldsymbol{\alpha}_{t+1}, \boldsymbol{\alpha}_t), \\ \Rightarrow \ln m \geq \Delta(\boldsymbol{\alpha}^*, \boldsymbol{\alpha}_1) &> T\nu^2/2, \end{aligned}$$

and so the number of iterations required is at most $T = \lceil \frac{2 \ln m}{\nu^2} \rceil$ for a fixed ϵ. This gives a faster finite convergence result than the linear asymptotic convergence of the SMO algorithm [Lin, 2001].

Assuming that (11) is satisfied for all t and following the convergence proof here[5], the solution of the ρ-SVM's dual, $\boldsymbol{\alpha}_t \to \boldsymbol{\alpha}^*$ in at most T iterations such that $\boldsymbol{\alpha}'_{T+1}\tilde{\mathbf{K}}\boldsymbol{\alpha}_{T+1} \to \rho^*$. Thus, the final solution is the desired ρ-SVM classifier.

3 Ensemble of Partially Trained SVMs

There is a close resemblance between the update rules of the ρ-SVM with variants of the AdaBoost algorithm. In particular, we will show in Section 3.1 that the algorithm in Section 2.2 (with ρ^* assumed to be known) is similar to the AdaBoost$_\varrho$ algorithm (Algorithm 2); while that in Section 2.3 (with ρ^* unknown) is similar to the AdaBoost$^*_\nu$ algorithm [Rätsch and Warmuth, 2005]. Inspired by this connection, in Section 3.2, we consider using the output of the boosting ensemble for improved performance. The time complexity will be considered in Section 3.3.

Algorithm 2 AdaBoost$_\varrho$ [Rätsch and Warmuth, 2005]

```
1: Input: $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,m}$; Number of iteration $T$.
2: Initialize: $d_i^{(1)} = 1/m$ for all $i = 1, \ldots, m$.
3: for $t = 1, \ldots, T$ do
4:     Train a weak classifier w.r.t. the distribution $d_i$ for each
       pattern $\mathbf{x}_i$ to obtain a hypothesis $h_t \in [-1, 1]$.
5:     Set
           $c_t = \arg\min_{c \geq 0} N_t(c) \exp(\varrho c),$        (20)
       where $N_t(c) = \sum_{i=1}^m d_i^{(t)} \exp(-c y_i h_t(\mathbf{x}_i))$.
6:     Update distributions $d_i$:
           $d_i^{(t+1)} = d_i^{(t)} \exp(-c_t y_i h_t(\mathbf{x}_i)) / N_t(c_t).$  (21)
7: end for
8: Output: $f(\mathbf{x}) = \sum_{t=1}^T \frac{c_t}{\sum_{r=1}^T c_r} h_t(\mathbf{x})$.
```

3.1 ρ-SVM Training vs Boosting

By comparing the update rules of $\alpha_i^{(t)}$ and η_t in Section 2.2 with those of $d_i^{(t)}$ and c_t in AdaBoost$_\varrho$, we can see that they are identical, with $\alpha_i^{(t)}$ playing the role of $d_i^{(t)}$ (compare (8) and (21)), and η_t the role of c_t (compare (9) and (20)). This is a consequence of the fact that boosting can also be regarded as entropy projection [Kivinen and Warmuth, 1999]. From this boosting perspective, we are effectively using base hypothesis of the form (3), with $\boldsymbol{\alpha} \in P_m$ (not necessarily equal to $\boldsymbol{\alpha}^*$). This base hypothesis can be regarded as a variant of the Parzen window classifier, with the patterns weighted by $\boldsymbol{\alpha}$. Note that the edge[6] of the base hypothesis at the t-th iteration is $\sum_{i=1}^m \alpha_i y_i \left(\sum_{j=1}^m \alpha_i^{(t)} y_i k(\mathbf{x}_i, \mathbf{x}_j) \right) = \boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha}_t$. Thus, the constraint in (7) is the same as requiring that the edge of the base hypothesis w.r.t. the α_i distribution to be ρ^*.

The same correspondence also holds between the ρ-SVM algorithm (Algorithm 1) and the AdaBoost$^*_\nu$ algorithm[7], with additionally that ρ_t in ρ-SVM is analogous to ϱ_t in AdaBoost$^*_\nu$. Moreover, ϵ in the ρ-SVM is similar to ν in AdaBoost$^*_\nu$ in controlling the approximation quality of the maximum margin. Note, however, that ν is quite difficult to set in practice. On the other hand, our ϵ_t can be set adaptively.

There have been some other well-known connections between SVM and boosting (e.g., [Rätsch, 2001]). However, typically these are interested in relating the combined hypothesis with the SVM, and the fact that boosting uses the L1 norm while SVM uses the L2 norm. Here, on the other hand, our focus is on showing that by using the weighted Parzen window classifier as the base hypothesis, then the last hypothesis is indeed a SVM.

3.2 Using the Whole Ensemble for Prediction

As mentioned in Section 1, the hard-margin SVM (which is the same as the last hypothesis) may have poor performance on noisy data. Inspired by its connection with boosting above, we will use the boosting ensemble's output for prediction:

$$\sum_{t=1}^T \frac{\eta_t}{\sum_{r=1}^T \eta_r} f_t(\mathbf{x}),$$

which is a convex combination of all the partially trained SVMs. Note that this takes little extra cost. As evidenced in the boosting literature, the use of such an ensemble can lead to better performance, and this will also be experimentally demonstrated in Section 4.

3.3 Computational Issue

In (8), each step of the EG update takes $O(m^2)$ time, which can be excessive on large data sets. This can be alleviated by performing low-rank approximations on the kernel matrix $\mathbf{K}$ [Fine and Scheinberg, 2001], which takes $O(mr^2)$ time (r is the rank of $\mathbf{K}$). Then, the EG update and computation of $\bar{\rho}_t$ both take $O(mr)$, instead of $O(m^2)$, time.

4 Experiments

In this section, experiments are performed on five small (breast cancer, diabetis, german, titanic and waveform)[8] and

[5]If $\bar{\rho}_t \frac{1-\epsilon_t}{1+\epsilon_t} < \rho^*$, then $\rho^* \leq \boldsymbol{\alpha}'_t\tilde{\mathbf{K}}\boldsymbol{\alpha}_t < \rho^*(1 + \frac{2\epsilon_t}{1-\epsilon_t})$ which satisfies the loose KKT condition for the ρ-SVM.

[6]The edge of the hypothesis h_t is $\gamma_t = \sum_{i=1}^m d_i^{(t)} y_i h_t(\mathbf{x}_i)$.

[7]In the AdaBoost$^*_\nu$ algorithm, ϱ in Algorithm 2 is replaced by $\varrho_t = \min_{r=1,\ldots,t} \gamma_r - \nu$, where γ_t is the edge of h_t.

[8]http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

seven medium-to-large real-world data sets (astro-particles, adult1, adult2, adult3, adult, web1 and web2)[9] (Table 1).

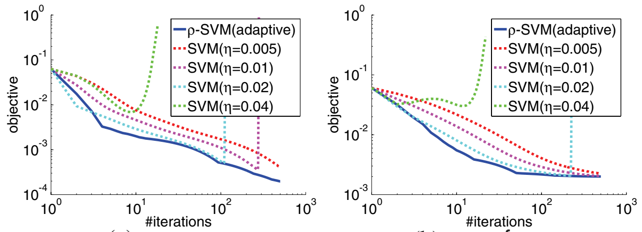
Table 1: A summary of the data sets used.

data set	dim	# training patterns	# test patterns
cancer	9	200	77
diabetis	8	468	300
german	20	700	300
titanic	3	150	2,051
waveform	21	200	4,600
astro	4	3,089	4,000
adult1	123	1,605	30,956
adult2	123	2,265	30,296
adult3	123	3,185	29,376
adult	123	32,561	16,281
web1	300	2,477	47,272
web2	300	3,470	46,279

4.1 Adaptive Learning Rate

We first demonstrate the advantages of the adaptive learning rate scheme (in Section 2.2). Because of the lack of space, we only report results on breast cancer and waveform. Similar behavior is observed on the other data sets. For comparison, we also train the ρ-SVM using the multiplicative updating rule (with fixed η) in [Cristianini *et al.*, 1999].

As can be seen from Figure 1, the performance of [Cristianini *et al.*, 1999] is sensitive to the fixed choice of η. When η is small (e.g., $\eta = 0.01$ or smaller), the objective value decreases gradually. However, when η is slightly larger (say, at 0.02), the estimate will finally over-shoot (as indicated by the vertical lines) and convergence can no longer be obtained. On the other hand, our proposed adaptive scheme always converges much faster.



(a) cancer (b) waveform

Figure 1: Comparing the EG update (with fixed η) with our adaptive scheme.

4.2 Accuracy and Speed

Next, we show that the proposed ensemble of partially trained hard-margin SVMs (Section 3.2) has comparable performance as the best-tuned soft-margin SVM (C-SVM). We use the Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\beta)$, with $\beta = \frac{1}{m}\sum_{i=1}^m \left\|\mathbf{x}_i - \frac{1}{m}\sum_{j=1}^m \mathbf{x}_j\right\|^2$. Each small data set in Table 1 comes with 100 realizations, and the performance is obtained by averaging over all these realizations. However, for each large data set, only one realization is available.

The ensemble is compared with the following:

1. SVM (BEST): The best-performing SVM among all the C-SVMs obtained by the regularization path algorithm[10] [Hastie *et al.*, 2005];
2. SVM (LOO): The C-SVM which has the best leave-one-out-error bound [Evgeniou *et al.*, 2004] among those obtained by the regularization path algorithm;
3. the Parzen window classifier, which is the first hypothesis added to the ensemble;
4. the (hard-margin) ρ-SVM, which is the last hypothesis;
5. the proposed ensemble.

All these are implemented in Matlab[11] and run on a 3.2GHz Pentium–4 machine with 1GB RAM.

Table 2: Testing errors (in %) on the different data sets.

data set	SVM (BEST)	SVM (LOO)	Parzen window	ρ-SVM	proposed ensemble
cancer	**23.1**	26.1	27.4	33.5	28.2
diabetis	**24.8**	28.9	33.4	34.5	**24.8**
german	**22.6**	24.9	29.8	29.3	23.7
titanic	22.7	25.5	23.3	45.2	**22.5**
waveform	11.2	18.8	22.0	11.2	**10.0**
astro	5.8	9.3	7.0	22.1	**3.4**
adult1	16.3	17.7	24.1	19.4	**15.9**
adult2	16.0	17.4	24.0	22.8	**15.8**
adult3	16.1	16.2	24.1	19.7	**15.9**
web1	**2.3**	2.8	3.0	14.4	2.8
web2	**2.1**	2.2	3.0	3.4	**2.1**

As can be seen from Table 2, the ensemble performs much better than the (hard-margin) ρ-SVM. Indeed, its accuracy is comparable to or sometimes even better that of SVM(BEST). Table 3 compares the time[12] for obtaining the ensemble with that of running the regularization path algorithm. As can be seen, obtaining the ensemble is much faster, and takes a small number of iterations. To illustrate the performance on large data sets, we experiment on the full adult data set in Table 1. The regularization path algorithm cannot be run on this large data. So, we use instead a ν-SVM (using LIBSVM[13]) for comparison, where ν (unlike the C parameter in the C-SVM)

[9]astro-particles is downloaded from http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/, while others are from http://research.microsoft.com/users/jplatt/smo.html.

[10]Alternatively, one can find the best-tuned SVM by performing a grid search on the soft-margin parameter. However, as each grid point then requires a complete re-training of the SVM, the regularization path algorithm is usually more efficient [Hastie *et al.*, 2005].

[11]The inner QP in the regularization path algorithm is solved by using the optimization package Mosek (http://www.mosek.com).

[12]This includes the time for computing the kernel matrix, which can be expensive on large data sets. Moreover, time for computing the leave-one-out bound is not included in the timing of SVM(LOO).

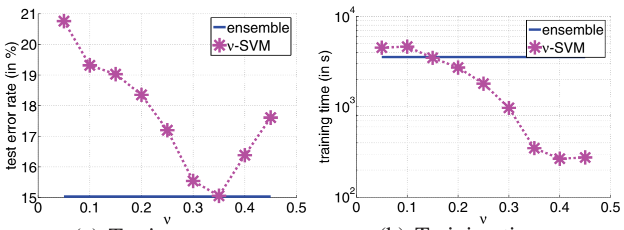
[13]Version 2.8.1 (C++ implementation) from http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

Table 3: Training time (in seconds) and number of iterations.

	regularization path	proposed ensemble	
data set	time	time	#iterations
cancer	0.4	**0.03**	62.3
diabetis	3.1	**0.1**	65.5
german	22.2	**0.3**	44.9
titanic	**0.08**	**0.08**	434.4
waveform	2.7	**0.1**	42.4
astro	180.3	**9.5**	310
adult1	243.3	**10.7**	42
adult2	632.7	**23.1**	40
adult3	957.3	**49.1**	104
web1	2,853.7	**64.3**	33
web2	6,978.7	**129.9**	125

can only be in $[0, 1]$. In the experiments, we further restrict ν to $[0.05, 0.45]$ as the optimization becomes infeasible when ν falls outside this range. Moreover, low-rank approximation of the kernel matrix as mentioned in Section 3.3 is used.

As can be seen from Figure 2, training of the single best-tuned ν-SVM takes 348.9s, while training the whole ensemble takes 3,555.1s, which is thus around 10 times slower. However, recall that ours is implemented in Matlab while LIBSVM is in C++. Moreover, as shown in Figure 2(b), SVM's training time is highly dependent on the value of ν. Hence, in practice, one has to perform SVM training together with ν parameter selection, which can be much more expensive than ensemble training.



(a) Testing error. (b) Training time.

Figure 2: Training of the ensemble vs training of a single ν-SVM on the adult data.

5 Conclusion

In this paper, we show that the hard-margin SVM can be trained by a simple multiplicative updating rule. By casting its training as a Bregman projection problem, the optimal learning rate can be adaptively obtained from the data, and a finite convergence result can also be derived. Moreover, inspired from a connection between boosting and Bregman projection, we propose an ensemble classifier consisting of the partially trained hard-margin SVMs. Though its base hypothesis are hard-margin SVMs, experimental results show that this ensemble is resistant to overfitting and has comparable performance with the best-tuned soft-margin SVM. Moreover, training the whole ensemble (which in Matlab) is only 10 times slower than that of training the best-tuned SVM (in C++). Thus, we can obtain an accurate classifier while avoiding tedious tuning of the soft-margin parameter.

One problem with the ensemble is that its solution is no longer sparse. In the future, we will investigate the use of reducd set methods to alleviate this.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region.

References

[Collins and Schapire, 2002] M. Collins and R.E. Schapire. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

[Cristianini *et al.*, 1999] N. Cristianini, C. Campbell, and J. Shawe-Taylor. Multiplicative updatings for support vector machines. In *Proceeding of European Symposium on Artificial Neural Networks*, Bruges, Belgium, 1999.

[Evgeniou *et al.*, 2004] T. Evgeniou, M. Pontil, and A. Elisseeff. Leave one out error, stability, and generalization of voting combinations of classifiers. *Machine Learning*, 55(1):71–97, April 2004.

[Fedotov *et al.*, 2001] A.A. Fedotov, P. Harremoes, and F. Topsoe. Vajda's tight lower bound and refinements of Pinsker's inequality. In *Proceedings of the International Symposium on Information Theory*, San Mateo, CA, 2001.

[Fine and Scheinberg, 2001] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, December 2001.

[Hastie *et al.*, 2005] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 561–568. MIT Press, Cambridge, MA, 2005.

[Kim *et al.*, 2003] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757–2767, 2003.

[Kivinen and Warmuth, 1997] J. Kivinen and M.K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, 1997.

[Kivinen and Warmuth, 1999] J. Kivinen and M. K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 134 – 144, Santa Cruz, CA, USA, 1999.

[Lin, 2001] C.-J. Lin. Linear convergence of a decomposition method for support vector machines. Technical report, National Taiwan University, 2001.

[Platt, 1999] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

[Rätsch and Warmuth, 2005] G Rätsch and M Warmuth. Efficient margin maximization with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

[Rätsch, 2001] G. Rätsch. *Robust Boosting via Convex Optimization: Theory and Applications*. PhD thesis, University of Potsdam, October 2001.