Boosting Support Vector Machines

Elkin García, Fernando Lozano

Departamento de Ingeniería Eléctrica y Electrónica
Universidad de los Andes, Bogotá, Colombia.
{ elkin-ga, flozano } @ uniandes.edu.co

Abstract. This paper presents a classification algorithm based on Support Vector Machines classifiers combined with Boosting techniques. This classifier presents a better performance in training time, a similar generalization and a similar model complexity but the model representation is more compact.

1 Introduction

Support Vector Machines *(SVM)* have been applied successfully in many problems in classification and regression [1]. It has been shown recently that for some of the kernel functions used in practice [2] SVMs are strong learners, in the sense that they can achieve a generalization error arbitrarily close to the Bayes error with a sufficiently large training set.

The main disadvantage of using *SVMs* is that the running time of training algorithms do not scale well with the size of the training set . If m is the size of the training set, training an SVM involves solving a quadratic program (QP) of size m^2, which takes $O(m^3)$ with a general purposed QP solver. Multiple researchers have proposed some methods that improve this running time to $O(m^2)$ [3,4,5,6].

On the other hand, boosting algorithms like Adaboost [7] find a good hypothesis combining appropriately hypothesis produced by a weak (or base) learner. Roughly, a weak learner is a learner that returns a hypothesis that outperform random guessing.

Using a classifier that is already strong (like SVM) as the base learner in Adaboost does not seem to offer significant advantages in terms of generalization error. In fact, Wickramaratna, Holden and Buxton have used *SVM* as the base classifier of Adaboost and have reported that the performance of the classifier decreases as the number of rounds increases [8]. However, as we will show below, a *weakened* version of SVM can still be useful as a base classifier. As shown in [9] the size of the weights assigned by Adaboost to the weak *SVMs*, serve as an indication of which data points are likely to become support vectors in the final model, and hence can be useful in the implementation of editing algorithms.

This paper is organized as follows. Section 2 reviews basic concepts of *classification* as well as the basics of *Boosting* algorithms and the *SVM* algorithm. In Section 3 we present the new algorithm *Boosting Support Vector Machines (BSVM)*. Section 4 presents experiments using the new algorithm on real world data. Finally, section 5 presents the conclusions of this work.

2 Preliminaries

Let $\mathcal{X}$ be an input space, $\mathcal{Y}$ a space of labels and Δ a distribution over $\mathcal{X}x\mathcal{Y}$. Let $\mathcal{S}=\{\langle \mathbf{x}_i, y_i\rangle\}_{i=1}^{m}$ be a set of labeled examples where each pair $\mathbf{x}_i, y_i$ is drawn i.i.d. according to Δ. For a binary classification problem the space of labels is restricted to $\mathcal{Y} = \{-1, +1\}$.

A *classification rule*, also called *hypothesis*, is a function $h : \mathcal{X} \mapsto \mathcal{Y}$. The hypothesis assigns a label to elements of the input space. In the binary classification problem the hypothesis is $h : \mathcal{X} \mapsto [-1, +1]$, where the sign of $h(\mathbf{x})$ is interpreted as the predicted label of $\mathbf{x}$ and the magnitude $|h(\mathbf{x})|$ is interpreted as the "confidence" in this prediction. Let $\mathcal{H}$ denote a class of hypothesis.

The performance of the hypothesis is evaluated using the generalization error R and the empirical error R_{emp}:

$$R(h) = \mathbb{P}_{(\mathbf{x},y\sim\Delta)}\{sgn(h(\mathbf{x})) \neq y\} \tag{1}$$

$$R_{emp}(h, \mathcal{S}, \mathcal{D}) = \sum_{i=1}^{m} D(i)[\![sign(h(\mathbf{x}_i)) \neq y_i]\!] \tag{2}$$

Where $D \in \mathbb{R}^m$ is a discrete distribution over the set $\mathcal{S}$ and $[\![\cdot]\!]$ is the indicator function.

A *learning algorithm* is an efficient algorithm that takes as inputs a set $\mathcal{S}$ of labeled examples and a discrete distribution $D \in \mathbb{R}^m$ over $\mathcal{S}$ and returns a hypothesis $h \in \mathcal{H}$. A *combined classifier* $H(\mathbf{x})$ is a convex combination of severals hypothesis h_i. This is

$$H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x}) \tag{3}$$

Where $\alpha_i \geq 0$ and $\sum_{i=1}^{T} \alpha_i = 1$. Each hypothesis h_i is a *base classifier.*

2.1 Boosting

Boosting algorithms improve the performance of a learning algorithm by combining several hypothesis in a proper manner. In particular, the Adaboost algorithm constructs a linear combination of hypothesis that are produced by calling a weak learner in a succession of rounds. At each iteration t, Adaboost makes a call to the weak learner *Weak* with input set $\mathcal{S}$ and distribution D_t and returns a hypothesis h_t. The distribution D_t is updated so that the weights of the examples that h_t misclassifies are reduced and the weight of the examples that h_t classifies correctly are increased. In this way, in the next iteration the weak learner is forced to focus its attention in the examples that have been misclassified more often in previous rounds. Figure 1 shows this algorithm as it was presented in [10].

```
**Algorithm:** Adaboost ($\mathcal{S}, D_1, T, Weak$)
**Input**: $\mathcal{S} = \{x_i, y_i\}_{i=1}^m,\ D_1,\ T,\ Weak(\cdot,\cdot)$
**Output**: $H(\cdot)$
**for** $t = 1$ **to** $T$ **do**
    Obtain a weak hypothesis using $D_t$.
    $h_t \leftarrow Weak(\mathcal{S}, D_t)$.
    Select $\alpha_t \in \mathbb{R}$. Usually:
        $\alpha_t = \frac{1}{2} \ln \left( \frac{1 - R_{emp}(h_t, \mathcal{S}, D_t)}{R_{emp}(h_t, \mathcal{S}, D_t)} \right).$   (4)
    Update:
        $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}.$   (5)
    where $Z_t$ is a normalization factor so that $D_{t+1}$ will be a distribution.
**end**
Return final hypothesis:
    $H(x) = sign \left( \sum_{t=1}^{T} \frac{\alpha_t}{\sum_t \alpha_t} h_t(x) \right).$   (6)
```

Fig. 1: Algoritmo Adaboost.

2.2 Support Vector Machines

The aim of Support Vector Machines in the binary classification problem is to find the optimal separating hyperplane (this is the hyperplane that maximizes the geometric margin) in a high dimensional *feature space* $\mathcal{X}'$. This space is related to the input space $\mathcal{X}$ by a nonlinear transformation $\Phi(x)$. The idea of this transformation is to project the data to a space where it is approximately separable by a linear threshold function.

When the data is not linearly separable the problem of finding the linear threshold function with the smallest classification error is NP-Hard [11]. The idea in SVMs is to allow some of the examples to be misclassified while keeping a large margin in the remaining data.

Cortes and Vapnik [12, 13] suggest to solve the following optimization problem. This formulation is known as $C\!-\!SVM$:

$$\begin{aligned} \min_{\substack{\mathbf{w} \in \mathcal{X}', \xi \in \mathbb{R}^m \\ b \in \mathbb{R}}} \quad & \tau(\mathbf{w}, \xi, \rho) = \frac{1}{2} \|\mathbf{w}\|^2_{\mathcal{X}'} + \frac{C}{m} \sum_{i=1}^{m} \xi_i \\ s.t. \quad & y_i(\langle \mathbf{w}, \Phi(x_i) \rangle_{\mathcal{X}'} + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \qquad \text{for} \quad i = 1, \ldots, m \end{aligned} \tag{7}$$

where $C > 0$ is a constant that controls the "trade-off" between minimizing the training error and maximizing the margin. A positive slack variable $\xi_i > 0$ indicates a classification mistake. It is more convenient to look at this problem in its dual form:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\
s.t. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\
& 0 \le \alpha_i \le \frac{C}{m} \qquad \text{for} \quad i = 1, \dots, m
\end{aligned} \tag{8}$$

where $Q_{ij}{=}y_i y_j k(x_i, x_j)$, $k(x_i, x_j){=}\langle \Phi(x_i), \Phi(x_j)\rangle_{\mathcal{X}'}$ is a positive defined kernel and $\mathbf{e}$ is a vector of ones. The solution of this problem has the form:

$$h_{\mathbf{w},b}(x) = sgn\left(\sum_{i=1}^{m} y_i\alpha_i k(x, x_i) + b\right) \tag{9}$$

Note that in the solution in feature space $\mathcal{X}'$ the training examples appear only through the kernel function. Data points for which $\alpha_i \neq 0$, are called support vectors.

Schölkopf et. al. [14] propose a modification to the optimization problem (8). Here, the parameter C is replaced by the parameter $\nu \in (0, 1]$. This new problem is known as *ν-SVM*

$$\begin{aligned}
\min_{\substack{\mathbf{w}\in\mathcal{X}', \xi\in\mathbb{R}^m \\ \rho, b\in\mathbb{R}}} \quad & \tau(\mathbf{w}, \xi, \rho) = \frac{1}{2}\|\mathbf{w}\|^2_{\mathcal{X}'} - \nu\rho + \frac{1}{m}\sum_{i=1}^{m}\xi_i \\
s.t. \quad & y_i(\langle \mathbf{w}, \Phi(x_i)\rangle_{\mathcal{X}'} + b) \ge \rho - \xi_i \\
& \xi_i \ge 0 \qquad \text{for} \quad i = 1, \dots, m \\
& \rho \ge 0
\end{aligned} \tag{10}$$

In this optimization problem, a new parameter ρ controls the margin between the classes. If $\xi = 0$ then the first restriction of (10) establishes a margin of $2\rho/\|w\|$. An advantage of this formulation is that ν is an upper bound of the training error and a lower bound of the number of support vectors if $\rho > 0$. The dual problem of (10) is

$$\begin{aligned}
\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} \\
s.t. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\
& 0 \le \alpha_i \le \frac{1}{m} \qquad \text{for} \quad i = 1, \dots, m \\
& \mathbf{e}^T\boldsymbol{\alpha} \ge \nu
\end{aligned} \tag{11}$$

Where Q_{ij} and $\mathbf{e}$ are the same as (8) and the final hypothesis is defined in (9).

Even for moderate values of m the quadratic programs (8) and (11) are not easily solvable with generic QP solvers. Several methods have been proposed instead. *Chunking* techniques proposed initially by Vapnik [3], are based in two observations: 1) Removing training examples with $\alpha_i = 0$ do not change the solution of the QP problem. 2) A decomposition of the original problem (8) in smaller problems is easier and more efficient. Some of the algorithms based on these observations are [4,5]. Platt [6] proposes a technique called *Secuential Minimal Optimization (SMO)* where the original problem is chunked in QP problems of two variables, that have analytical solution.

Problem (11) includes an additional inequality constraint. Crisp and Burges [15] and Chang and Lin [16] have shown that $\mathbf{e}^T\boldsymbol{\alpha} \geq \nu$ can be replaced by $\mathbf{e}^T\boldsymbol{\alpha} = \nu$ without changing the solution, and the techniques described before can be adapted.

Joachims [5] employs a combination of *SMO* with *shrinking* and *caching* techniques to solve (8) and (11), that results in a running time that is between quadratic and cubic ,depending on the peculiarities of the problem. The computational complexity is basically dominated by the number of kernel evaluations [17].

3 Boosting Support Vector Machines

The running time of training algorithms for SVMs can be reduced if only a few training examples are involved in the actual computations. This fact can be exploited by Adaboost if at each iteration most of the weight in the distribution passed to the weak learner is assigned to a few data points.

The computation time can be reduced as follows. If the complexity of the original training algorithm is bounded by Am^x with $A \in \mathbb{R}$ then training with a fraction νm is bounded by $A(\nu m)^x$. Thus, training q hypothesis is bounded by $Aq(\nu m)^x$ (assuming the overhead introduced by Adaboost is negligible). Then if $x > 1$, $0 > \nu > 1$ and $q \leq 1/\nu$

$$Aq(\nu m)^x \leq \frac{A}{\nu}(\nu m)^x = A\nu^{x-1}m^x \leq Am^x \tag{12}$$

The complexity of this algorithm is still $O(m^x)$ but the constant is smaller.

For this reason is useful to make a Boosting algorithm similar to Adaboost using *SVM* as a weak learner. Then, it is necessary to modify the optimization problems (8) and (11) in therms of distributions and to guarantee that *SVM* is a weak learner.

3.1 Support Vector Machines for distributions

Let D be a discrete distribution over the set of labeled examples $\mathcal{S}=\{\langle x_i, y_i\rangle\}_{i=1}^m$. We want to incorporate this information in the optimization problem solved by *C-SVM* and *ν-SVM*.

A simple way of doing this is to use *bootstrap* samples of the original set drawn according to D [18]. A serious drawback of this method is that the matrix Q in (8) and (11) may become ill conditioned because examples with large weight in the distribution may appear several times in a bootstrap sample.

A better option is to modify the original optimization problem to incorporate directly the distribution. We present two alternatives of solution. The first alternative is to penalize the slack variables ξ_i proportional to D_i in the objective function. The optimization problem in *C-SVM* becomes:

$$\begin{aligned} \min_{\substack{\mathbf{w}\in\mathcal{X}',\xi\in\mathbb{R}^m \\ b\in\mathbb{R}}} \quad & \tau(\mathbf{w},\xi,\rho) = \frac{1}{2}\left\|\mathbf{w}\right\|^2_{\mathcal{X}'} + \frac{C}{m}\sum_{i=1}^{m} D_i\xi_i \\ s.t. \quad & y_i(\langle \mathbf{w}, \Phi(x_i)\rangle_{\mathcal{X}'} + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \qquad \text{for} \quad i = 1,\ldots,m \end{aligned} \tag{13}$$

And the optimization problem in *ν-SVM* is:

$$\begin{aligned} \min_{\substack{\mathbf{w}\in\mathcal{X}',\xi\in\mathbb{R}^m \\ \rho,b\in\mathbb{R}}} \quad & \tau(\mathbf{w},\xi,\rho) = \frac{1}{2}\left\|\mathbf{w}\right\|^2_{\mathcal{X}'} - \nu\rho + \frac{1}{m}\sum_{i=1}^{m} D_i\xi_i \\ s.t. \quad & y_i(\langle \mathbf{w}, \Phi(x_i)\rangle_{\mathcal{X}'} + b) \geq \rho - \xi_i \\ & \xi_i \geq 0 \qquad \text{for} \quad i = 1,\ldots,m \\ & \rho \geq 0 \end{aligned} \tag{14}$$

The dual problems of (13) and (14) are respectively

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ s.t. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq C D_i \qquad \text{for} \quad i = 1,\ldots,m \end{aligned} \tag{15}$$

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\ s.t. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq D_i \qquad \text{for} \quad i = 1,\ldots,m \\ & \mathbf{e}^T\boldsymbol{\alpha} \geq \nu \end{aligned} \tag{16}$$

The second alternative is to include the distribution in the objetive function but also force examples with larger weights to have large margin. The modified *C-SVM* and *ν-SVM* formulations are:

$$\begin{aligned} \min_{\substack{\mathbf{w}\in\mathcal{X}',\xi\in\mathbb{R}^m\\ b\in\mathbb{R}}} \quad & \tau(\mathbf{w},\xi,\rho) = \frac{1}{2}\|\mathbf{w}\|^2_{\mathcal{X}'} + \frac{C}{m}\sum_{i=1}^{m} D_i\xi_i \\ s.t. \quad & y_i(\langle \mathbf{w}, \Phi(x_i)\rangle_{\mathcal{X}'} + b) \geq mD_i(1-\xi_i) \\ & \xi_i \geq 0 \qquad \text{for} \quad i=1,\dots,m \end{aligned} \tag{17}$$

$$\begin{aligned} \min_{\substack{\mathbf{w}\in\mathcal{X}',\xi\in\mathbb{R}^m\\ \rho,b\in\mathbb{R}}} \quad & \tau(\mathbf{w},\xi,\rho) = \frac{1}{2}\|\mathbf{w}\|^2_{\mathcal{X}'} - \nu\rho + \frac{1}{m}\sum_{i=1}^{m} D_i\xi_i \\ s.t. \quad & y_i(\langle \mathbf{w}, \Phi(x_i)\rangle_{\mathcal{X}'} + b) \geq mD_i(\rho-\xi_i) \\ & \xi_i \geq 0 \qquad \text{for} \quad i=1,\dots,m \\ & \rho \geq 0 \end{aligned} \tag{18}$$

The dual problems of (17) and (18) are respectively:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - m\mathbf{D}^T\boldsymbol{\alpha} \\ s.t. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq \frac{C}{m} \qquad \text{for} \quad i=1,\dots,m \end{aligned} \tag{19}$$

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} \\ s.t. \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq \frac{1}{m} \qquad \text{for} \quad i=1,\dots,m \\ & m\mathbf{D}^T\boldsymbol{\alpha} \geq \nu \end{aligned} \tag{20}$$

Where $Q_{ij}{=}y_iy_jk(x_i,x_j)$ and $\mathbf{D}$ is the distribution vector.

3.2 Support Vector Machines like a Weak Learner

As mentioned before, a strong learner like SVM does not work well as the base learner of Adaboost. However, a version of SVM that has been weakened can be useful.

The idea is to discard a percentage μ of the examples in the original data set, and solve the optimization problem with a smaller training set. This will work as long as the training error on the whole data set remains under 50%. Thus, the idea is to discard the less representative examples in the data set, namely the examples with small weight under the current distribution. Then, the subset $\mathcal{J} \subseteq \mathcal{S}$ is defined by $\sum_{j\in\mathcal{J}} D_j \leq (1-\mu)$ where $\mathcal{J}$ has minimum cardinality.

```
**Algorithm:** WSVM($\mathcal{S}$,$D$,$SVM$,kernel, C or $\nu$, $\mu$)
**Input**: $\mathcal{S} = \{x_i, y_i\}_{i=1}^m$, $D$, $SVM(\cdot,\cdot)$, kernel, C or $\nu$, $\mu$
**Output**: $h(\cdot)$
Select $\mathcal{J}$ so that $\sum_{j\in\mathcal{J}} D_j \leq (1-\mu)$ and $\mathcal{J}$ has minimum cardinality
Select $\mathcal{S}^* = \{\langle x_j, y_j\rangle\}_{\mathcal{J}}$ and $D^* = D_j$
Obtain a weak hypothesis using $D^*$
    $h \leftarrow SVM(\mathcal{S}^*, D^*, \text{kernel, C o } \nu) = \sum_{i=1}^{m} y_i\beta_i k(x, x_i) + b$
```

Fig. 2: Weakened *SVM* Algorithm.

Notice that as D becomes more askew in later rounds of Adaboost, more and more data points are discarded, reducing dramatically the training time of the base SVM learners. The new algorithm is presented in 2.

If we use an universal kernel (a kernel is universal if every set of points in feature space is separable using this kernel) we can obtains a bound on the percentage of rejected data μ. This universal kernel is a strong learner with respect to subset $\mathcal{J}$. Then, we can obtain a classifier that has an error of at most ϵ (for small epsilon) and that in the worst case errs on all the elements of $\mathcal{J}'$. Provided that SVM is a weak learner we have that $(1-\mu)\epsilon + \mu < 1/2$ where

$$\mu < \frac{1/2 - \epsilon}{1 - \epsilon} \tag{21}$$

Thus, μ is close to 50%. However the bound is not tight because the elements of $\mathcal{J}$ and $\mathcal{J}'$ are correlated and hence the classification rule obtained with $\mathcal{J}$ does not have a large error on the set $\mathcal{J}'$ as long as the number of elements in $\mathcal{J}$ is large enough. For this reason, μ is left as a parameter of the algorithm. Notice that an appropriate value for this parameter can be determined using some model selection strategy.

3.3 Boosting Support Vector Machines

The weakened SVM can be incorporated directly into Adaboost, obtaining a combined classifier of the form:

$$H_T(x) = sgn\left(\sum_{i=1}^{T} \frac{\alpha_t}{\sum_t \alpha_t} sgn\left(\sum_{i=1}^{m} y_i\alpha_i k(x, x_i) + b\right)\right) \tag{22}$$

Notice however that 22 is not a hyperplane in feature space and that the complexity of the hypothesis $H_T(x)$ is larger than the complexity of the *SVM* in (9).

```
**Algorithm:** BSVM $(\mathcal{S}, D_1, T, WSVM, \mu, \text{kernel}, \text{C or } \nu)$
**Input**: $\mathcal{S} = \{x_i, y_i\}_{i=1}^m$, $D_1$, $T$, $WSVM$,$\mu$,kernel,C or $\nu$
**Output**: $H(\cdot)$
    $H(x) \leftarrow WSVM(\mathcal{S}, D_1, \text{kernel}, \text{C or } \nu) = \sum_{i=1}^{m} y_i \beta_{1,i} k(x, x_i) + b_1$
for $t = 2$ to $T$ or stopping condition do
    $D_t = \frac{D_{t-1}(i) \left( \frac{1 - R_{emp}(h_{t-1}, \mathcal{S}, D_{t-1})}{R_{emp}(h_{t-1}, \mathcal{S}, D_{t-1})} \right)^{-\frac{1}{2} y_t h_t}}{Z_{t-1}}$
    Where $Z_{t-1}$ is a nomalization factor so that $D_t$ will be a distribution.
    $h_t(x) \leftarrow WSVM(\mathcal{S}, D_t, \text{kernel}, \text{C or } \nu) = \sum_{i=1}^{m} y_i \beta_{t,i} k(x, x_i) + b_t$
    $H(x, \alpha) = \alpha h_t(x) + (1 - \alpha) H(x)$
    Choose $\alpha_t \in [0, 1]$ so that $\alpha_t = \arg\min \left( R_{emp} \left( H(\alpha, x), \mathcal{S}, D_1 \right) \right)$
    $H(x) = H(x, \alpha_t)$
end
Return final hypothesis:   $H(x) = sgn\left(H(x)\right)$
```

Fig. 3: Algorithm *BSVM*.

For this reason, we modify the Adaboost algorithm shown in figure 1 so that the combined classifier has the form 9. This new algorithm that we will refer to as *BSVM* is detailed on figure 3.

Algorithm *BSVM* keeps the same general structure of Adaboost. The main difference is that the final hypothesis is expressed as a hyperplane in feature space. The computation of the coefficients α_t of each hypothesis is also different: At each iteration, we compute the value of α_t that minimizes the training error over the convex combination of the current hypothesis $h_t(x)$ and the combined hypothesis of the previous iteration $H(x)$. This optimization problem can be easily solved by numeric methods such as golden search or cubic interpolation.

The stopping condition of *BSVM* can be set so that the algorithms halts when it can no longer improve, i.e. when the error of the combined classifier can not be decreased further. This can be accomplished by stopping when $\alpha_t \approx 0$ or when $R_{emp}(H(x), \mathcal{S}, D_1) = 0$.

From the generalization point of view it may be better to stop the algorithm when it starts over fitting the data. This can be done by checking the behavior of the empirical error on an independent cross validation set. Another alternative is to look at the slope of the training error and stop the algorithm when it becomes (approximately) flat. We can compute:

$$\frac{R_{emp.train}[t-1] - R_{emp.train}[t]}{R_{emp.train}[t-1]} \leq f[t] \tag{23}$$

where $f[t]$ is non increasing.

4 Experiments

In this section, we present some experiments demonstrating the performance of our algorithm on synthetic and real world data sets.

The data sets we utilize are the following:

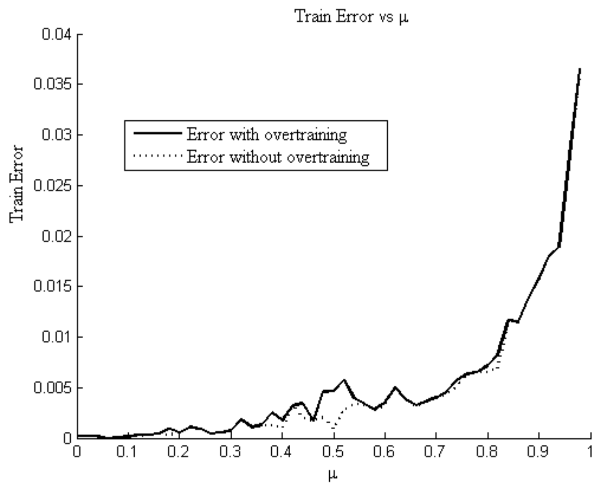
- The *MNIST* data set of handwritten digits [19]. To restrict the problem to binary classification, we consider the separation between digits 3 and 8. This is a relatively difficult task, with high dimensionality and moderately large number of examples.
- The synthetic data set *Four-norm* is a binary classification problem in 20 dimensions. In this problem, examples from one class are drawn from one of two normal distributions with equal probability. The normal distributions have identity covariance matrix and means $(a, a, ..., a)$ and $(-a, -a, ..., -a)$. Similarly, examples from the second class are drawn from one of two normal distributions with equal probability, identity covariance matrix and means $(a, -a, ..., a, -a)$ and $(-a, a, ..., -a, a)$. In this case we set $a = \sqrt{2}$.
- Data sets *Breast cancer*, *diabetes* and *australian* from the *UCI* repository. [20].

For data sets that do not have test examples we set apart 10% of the data for testing. In all experiments we utilize gaussian kernels $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\sigma)$. A summary of the data sets and the settings of the parameters of the gaussian kernel is shown in table 1.

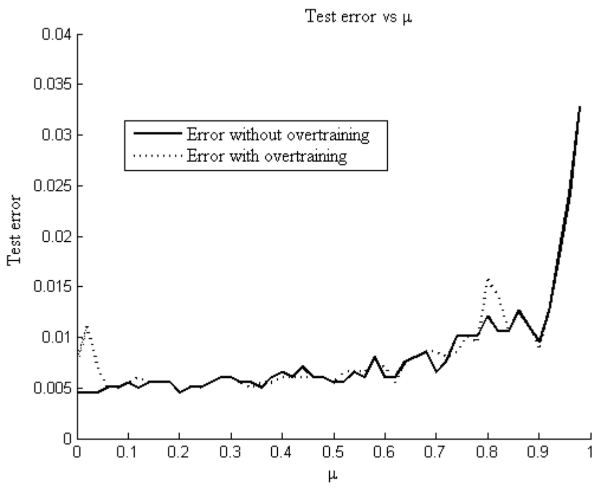
Table 1: Data sets description

Database	**# Training Elements**	**# Testing Elements**	**Dimension**	$\boldsymbol{\sigma}$
MNIST	11982	1984	784	4M
Four norm	1000	10000	20	1k
Breast Cancer	615	68	9	100k
Diabetes	692	76	8	40
Australian	621	69	14	10

To train the *SVM* with distributions (i.e. solve problems (15) and (16)), we follow a strategy similar to the used by the *SMO* algorithm [6] and the improvements suggested by others authors [17, 16].



(a) Relation between training error and percentage of rejected data μ.



(b) Relation between test error and percentage of rejected data μ.

Fig. 4: Relation between error and percentage of rejected data μ.

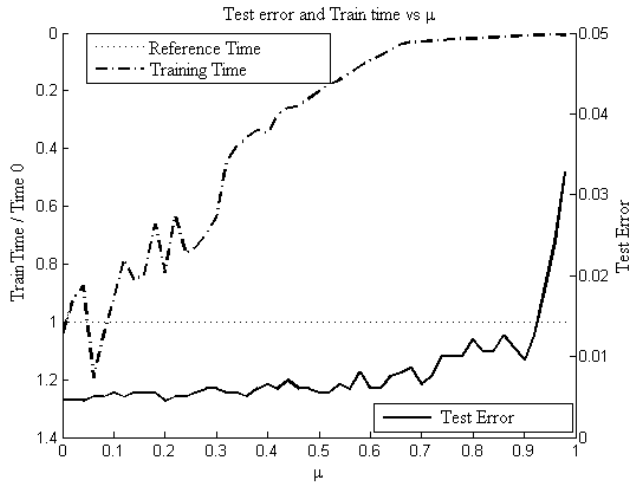


Fig. 5: Test error and train time vs μ.

MNIST is used for a general analysis of the algorithm because it has a large number of examples. Figure 4a shows the training error as a function of the percentage of rejected data μ. It is clear that the training error can not be reduced to the original value when more examples are rejected. We show also the different outcomes obtained by including different stopping condition (described in section 3.3). We use $f[t] = 0.25$ and $g[t] = 1/t^2$. Although the training error does not diminish significantly, figure 4b shows the improved performance of our algorithm in terms of generalization error, and reduced computation time.

Figure 5 shows the relation between the test error (without over fitting) and the training time when a percentage of examples is rejected. Although the generalization of the model with rejected data is similar to the generalization error of the original model (using all the data), the training time decreases dramatically almost to $\frac{1}{10}$ of original time. As expected, there is a breaking point where generalization gets worse when a lot of data is rejected.

Regarding the number of support vectors, figure 6 shows that if the percentage of rejected examples increases then the number of support vectors decreases (using the stopping conditions) independently of the training error and the test error. However, we have observed that if stopping condition are not applied, an increase in the number of support vectors corresponds to an increase in the generalization error. This suggest that it is possible to use the number of support vectors as a stopping criterion.

Table 2 summarizes the results of *C-SVM* in the different data sets and table 3 summarizes the results of *ν-SVM*. *BSVM* obtained a similar generalization error with respect to the original algorithm in a few rounds. It demonstrates the performance and rapid convergence of our algorithm. Additionally, the training time is lower in most cases, in particular with *MNIST* and *australian* which are

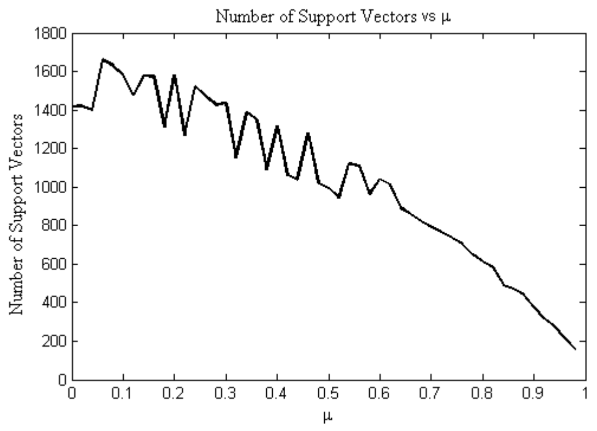


Fig. 6: Number of Support Vectors vs μ.

Table 2: Results of *C-SVM*

Database	**C**	**Test Error** $\boldsymbol{SVM}$**(%)**	$\boldsymbol{\mu}$	**#** **Iterations**	**Test Error** $\boldsymbol{BSVM}$**(%)**	**Time**$\boldsymbol{BSVM}$ **/Time**$\boldsymbol{SVM}$
MNIST	10	0.45	0.64	3	0.60	0.0762
Four norm	50	23.50	0.60	4	20.77	0.5212
	200	19.68	0.60	4	18.48	0.5617
B. Cancer	100	0.00	0.7	3	0.00	0.9286
Diabetes	50	19.74	0.7	3	22.37	0.3636
Australian	100	18.84	0.7	3	15.94	0.135

Table 3: Results of ν-*SVM*

Database	$\boldsymbol{\nu}$	**Test Error** $\boldsymbol{SVM}$**(%)**	$\boldsymbol{\mu}$	**#** **Iterations**	**Test Error** $\boldsymbol{BSVM}$**(%)**	**Time**$\boldsymbol{BSVM}$ **/Time**$\boldsymbol{SVM}$
MNIST	0.15	4.83	0.60	3	4.54	0.6554
Four norm	0.3	21.05	0.60	4	26.06	0.8532
B. Cancer	0.4	1.47	0.80	3	0.00	1.52
Diabetes	0.2	26.32	0.80	3	26.32	1.13
Australian	0.4	15.94	0.80	3	13.04	0.76

large sets of high dimensionality. Finally, in general, algorithm *C-BSVM* is more efficient than ν*-BSVM*.

Table 4: Number Support Vectors Comparison

Database	**C**	**# S. Vectors** *CSVM*	**# S. Vectors** *CBSVM*	$\boldsymbol{\nu}$	**# S. Vectors** $\boldsymbol{\nu}$ *SVM*	**# S. Vectors** $\boldsymbol{\nu}$ *BSVM*
MNIST	10	1417	1012	0.15	1817	1175
Four norm	200	688	454	0.3	340	395
Breast Cancer	100	78	46	0.4	216	58
Diabetes	50	327	147	0.2	121	50
Australian	100	202	104	0.4	245	93

The relation between the number of support vectors is shown in table 4. *BSVM* obtains a reduced number of support vectors while maintaining the same generalization error. There is a large reduction in no separable classes with large Bayes error, *four-norm* and *diabetes* are examples of this reduction.

5 Conclusions

The proposed algorithm *BSVM* combined efficiently *SVM* classifiers using Boosting techniques. This new algorithm does not increase the complexity of the final hypothesis while decreasing the training time specially when the training set is large or the dimensionality of the data is high.

The models produced by our algorithms are more compact because they have less support vectors.

The proposed strategies for over fitting are effective because *BSVM* presents similar values of generalization with respect to the original implementation.

Problems for future research are finding tighter theoretical bounds on the percentage of rejected examples and finding more exact stopping conditions.

References

1. Schölkopf, B., Smola, A.: Learning With Kernels. MIT Press, Cambridge, MA (2002)
2. Steinwart, I.: Sparness of support vector machines –some asymptotically sharp bounds. In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems. Volume 16., Cambridge, MA, MIT Press (2004) 169–184
3. Vapnik, V.: Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow (1979) (English translation: Springer Verlag, New York, 1982).

4. Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In Principe, J., Gile, L., Morgan, N., Wilson, E., eds.: Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop, New York, IEEE (1997) 276 – 285
5. Joachims, T.: Making large–scale SVM learning practical. In Schölkopf, B., Burges, C.J.C., Smola, A.J., eds.: Advances in Kernel Methods — Support Vector Learning, Cambridge, MA, MIT Press (1999) 169–184
6. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C.J.C., Smola, A.J., eds.: Advances in Kernel Methods — Support Vector Learning, Cambridge, MA, MIT Press (1999) 185–208
7. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55** (1997) 119–139
8. Wickramaratna, J., Holden, S., Buxton, B.: Performance degradation in boosting. In Kittler, J., Roli, F., eds.: Proceedings of the 2nd International Workshop on Multiple Classifier Systems MCS2001. Volume 2096 of LNCS. Springer (2001) 11–21
9. Rangel, P., Lozano, F., García, E.: Boosting of support vector machines with application to editing. In: Proceedings of the 4nd International Conference of Machine Learning and Applications ICMLA'05. Springer (2005)
10. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning **37** (1999) 297–336
11. Johnson, D., Preparata, F.: The densest hemisphere problem. Theorical Computer Science (1978) 93–107
12. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning **20** (1995) 273 – 297
13. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998) forthcoming.
14. Schölkopf, B., Smola, A., Williamson, R., Bartlett, P.: New support vector algorithms. NeuroCOLT Technical Report NC-TR-98-031, Royal Holloway College, University of London, UK (1998)
15. Crisp, D.J., Burges, C.J.C.: A geometric interpretation of ν-svm classifiers. In Solla, S.A., Leen, T.K., Müller, K.R., eds.: NIPS, The MIT Press (1999) 244–250
16. Chang, C., Lin, C.: Training ν-support vector classifiers: Theory and algorithms. Neural Computation (2001) 2119–2147
17. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
18. Pavlov, D., Mao, J., Dom, D.: Scaling-up support vector machines using boosting algorithm. 15th International Conference on Pattern Recognition **2** (2000) 219–222
19. LeCun, Y.: (MNIST handwritten digit database) Available as http://www.research.att.com/∼yann/ocr/mnist/.
20. D.J. Newman, S. Hettich, C.B., Merz, C.: UCI repository of machine learning databases (1998)