Simpler Core Vector Machines with Enclosing Balls

Ivor W. Tsang IVOR@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

Andras Kocsor KOCSOR@INF.U-SZEGED.HU
Research Group on Artificial Intelligence, Hungarian Academy of Sciences and University of Szeged, Hungary

James T. Kwok JAMESK@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

Abstract

The core vector machine (CVM) is a recent approach for scaling up kernel methods based on the notion of minimum enclosing ball (MEB). Though conceptually simple, an efficient implementation still requires a sophisticated numerical solver. In this paper, we introduce the enclosing ball (EB) problem where the ball's radius is fixed and thus does not have to be minimized. We develop efficient $(1+\epsilon)$-approximation algorithms that are simple to implement and do not require any numerical solver. For the Gaussian kernel in particular, a suitable choice of this (fixed) radius is easy to determine, and the center obtained from the $(1+\epsilon)$-approximation of this EB problem is close to the center of the corresponding MEB. Experimental results show that the proposed algorithm has accuracies comparable to the other large-scale SVM implementations, but can handle very large data sets and is even faster than the CVM in general.

1. Introduction

Large margin methods have been highly successful in supervised learning. In particular, support vector machines (SVM) have obtained outstanding performance in many machine learning problems such as classification, regression, and ranking. Traditionally, SVM training is formulated as a quadratic programming (QP) problem, which is then optimized by some numerical solver. However, a naive implementation takes $O(n^3)$ time and $O(n^2)$ space, where n is the number of training examples, and is thus computationally expensive even on medium-sized data sets. Hence, in practical applications, it is imperative to use more sophisticated techniques and a careful implementation.

A powerful approach to scale up SVM training is by using decomposition methods (Osuna et al., 1997), which break a large QP problem into a series of manageable QP subproblems. In recent years, various other scale-up strategies have been proposed. For example, Vishwanathan et al. (2003), in their SimpleSVM algorithm, use greedy working set selection strategies to identify support vectors for incremental update of the kernel sub-matrix. Bordes et al. (2005) use online SVM learning together with active example selection in their LASVM algorithm. Sonnenburg et al. (2006) exploit special data structures for fast computations of the kernel in their chunking algorithm. Moreover, instead of maximizing the dual problem as is usually done, Chapelle (2007) propose to directly minimize the primal problem.

In this paper, we focus on another recent algorithm called the core vector machine (CVM) (Tsang et al., 2005a), which combines techniques from computational geometry with SVM training. By reformulating SVM's QP as a minimum enclosing ball (MEB) problem, they then apply an efficient $(1+\epsilon)$ approximation algorithm (based on the so-called core-sets) to obtain a close-to-optimal SVM solution. As pointed out in (Har-Peled et al., 2007; Tsang & Kwok, 2007), these core-set algorithms are similar to the cutting-plane algorithm used in (Joachims, 2006) and the column generation method commonly used in large-scale linear programming. Experimentally, the CVM has demonstrated good performance in classification (Asharaf et al., 2006; Tsang et al., 2005a), regression (Tsang et al., 2005b) and semi-supervised learning (Tsang & Kwok, 2007). Besides, this connection between SVM and MEB has also been used for one-class classification with arbitrary Bregman divergence (Nock & Nielsen, 2005), support vector ordinal regression (Shevade & Chu, 2006), and agnostic learning in the presence of outliers (Har-Peled et al., 2007).

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007.

However, though conceptually simple, each iteration of the CVM algorithm involves a QP subproblem defined on the core-set. Thus, this again requires the use of a sophisticated numerical solver for efficient implementation. Moreover, for data sets that are very large or complicated, the size of the core-set can also be large and so this internal optimization problem can become computationally expensive.

Recall that the CVM is closely related to the *minimum* enclosing ball, and thus some optimization appears inevitable. In this paper, we propose to solve instead the simpler *enclosing ball* (EB) problem, where the radius of the ball is fixed. We propose efficient $(1+\epsilon)$-approximation algorithms that are simple to implement and do not require numerical solvers. Moreover, with a convenient choice of the radius, it can be shown that the center obtained from the $(1+\epsilon)$-approximation of this EB problem is close to the center of the corresponding MEB. Thus, the approximate SVM solution obtained is also close to the truly optimal SVM solution, especially as ϵ is typically small in practice. Experimental results show that the proposed algorithm has accuracies comparable to the other large-scale SVM implementations, but can handle very large data sets and is even faster than the CVM in general.

The rest of this paper is organized as follows. Section 2 first gives a short introduction to the CVM. Section 3 then describes the proposed *ball vector machine* (BVM) algorithm. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

2. Core Vector Machine

Let φ be the feature map corresponding to kernel k. Given a set of φ-mapped points[1] $\mathcal{S}_\varphi = \{\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n)\}$, its MEB, denoted $B(\mathbf{c}^*, R^*)$ with center $\mathbf{c}^*$ and radius R^*, is the smallest ball that encloses all these points:

$$(\mathbf{c}^*, R^*) = \arg\min_{\mathbf{c},R} R^2 : \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \le R^2 \ \ \forall i. \quad (1)$$

Its dual is the QP: $\max_{\lambda_i \ge 0} \sum_{i=1}^n \lambda_i k_{ii} - \sum_{i,j=1}^n \lambda_i \lambda_j k_{ij} : \sum_{i=1}^n \lambda_i = 1$, where $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)$. Conversely, any QP of this form can be regarded as a MEB problem. In particular, when k satisfies[2]

$$k(\mathbf{x}, \mathbf{x}) = \kappa, \quad (2)$$

a constant, this form includes the QPs associated with many kernel methods. For example, the two-class L2-SVM using kernel k can be transformed to a MEB problem with the feature map $\tilde{\varphi}((\mathbf{x}_i, y_i)) = [y_i\varphi(\mathbf{x}_i), y_i, \frac{1}{\sqrt{C}}\mathbf{e}_i']'$ (where $\mathbf{e}_i$ is the n-dimensional vector with all zeroes except that the ith position is equal to one), and the corresponding transformed kernel is (Tsang et al., 2005a)

$$\tilde{k}_{ij} = y_i y_j (k_{ij} + 1) + \delta_{ij}/C. \quad (3)$$

After transforming the QP to a MEB problem, the CVM uses an iterative $(1+\epsilon)$-approximation algorithm to obtain a near-optimal solution (Algorithm 1). Its basic idea is to maintain a core-set $\mathcal{S}_t$, which is a subset of $\mathcal{S}$, and its MEB $B(\mathbf{c}_t, R_t)$ at each iteration t. The ball $B(\mathbf{c}_t, R_t)$ is expanded by including a point falling outside $B(\mathbf{c}_t, (1+\epsilon)R_t)$ into the core-set. This is repeated until all the points in $\mathcal{S}$ are covered by $B(\mathbf{c}_t, (1+\epsilon)R_t)$. After obtaining an $(1+\epsilon)$-approximate solution $B(\mathbf{c}, R)$ of the MEB, the primal variables associated with the SVM (i.e., weight $\mathbf{w}$, bias b, and slack errors $\boldsymbol{\xi}$) can be recovered from

$$\mathbf{c} = [\mathbf{w}' \ \ b \ \ \sqrt{C}\boldsymbol{\xi}']', \quad (4)$$

where C is the regularization parameter in the SVM.

Algorithm 1 CVM algorithm.

1: Initialize $\mathcal{S}_0 = \{\varphi(\mathbf{z}_0)\}$, $\mathbf{c}_0 = \varphi(\mathbf{z}_0)$ and $R_0 = 0$.
2: Terminate if no $\varphi(\mathbf{z})$ falls outside $B(\mathbf{c}_t, (1+\epsilon)R_t)$. Otherwise, let $\varphi(\mathbf{z}_t)$ be such a point. Set $\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{\varphi(\mathbf{z}_t)\}$.
3: Find MEB$(\mathcal{S}_{t+1})$
4: Increment t by 1 and go back to Step 2.

3. Ball Vector Machine (BVM)

The CVM is tightly coupled to the MEB problem. Here, instead of finding the *minimum* enclosing ball, we consider the simpler problem of finding an enclosing ball (EB) with its radius given in advance.

Definition 1. *(**Enclosing ball** EB$(\mathcal{S}, r)$) Given the radius $r \ge R^*$, find a ball $B(\mathbf{c}, r)$ that encloses all the points in $\mathcal{S}$, i.e., $\|\mathbf{c} - \varphi(\mathbf{z}_i)\|^2 \le r^2$ for all $\varphi(\mathbf{z}_i)$'s in $\mathcal{S}$.*

In Section 3.1, we first propose an $(1+\epsilon)$-approximation algorithm for solving the EB problem, which is then improved by a more efficient, multi-scale version in Section 3.2. The radius of the EB can also be reduced automatically as shown in Section 3.3. Finally, the issues of how to set the EB's radius and how this EB problem is related to the solving of the SVM are addressed in Section 3.4.

3.1. $(1+\epsilon)$-Approximation Algorithm for EB$(\mathcal{S}, r)$

The proposed iterative algorithm is shown in Algorithm 2. It follows from a similar MEB algorithm in (Panigrahy, 2004), where $\varphi(\mathbf{z}_t)$ in Step 2 is further required to be the point furthest from $B(\mathbf{c}_t, r)$; while here it only has to be

[1] In the sequel, we will simply write $\mathcal{S}_\varphi$ as $\mathcal{S}$ if φ is clear from the context.

[2] Note that condition (2) is satisfied by many kernels, including the commonly used isotropic kernel (e.g., Gaussian kernel) and any normalized kernel.

outside $B(\mathbf{c}_t, (1+\epsilon)r)$). The ball's center is updated such that the new ball just touches $\varphi(\mathbf{z}_t)$. The whole procedure is repeated until no point falls outside $B(\mathbf{c}_{t+1}, (1+\epsilon)r)$. Obviously, this produces an $(1+\epsilon)$-approximation.

Algorithm 2 $(1+\epsilon)$-approximation algorithm for EB$(\mathcal{S}, r)$.

1: Initialize $\mathbf{c}_0 = \varphi(\mathbf{z}_0)$.
2: Terminate if no $\varphi(\mathbf{z})$ falls outside $B(\mathbf{c}_t, (1+\epsilon)r)$. Otherwise, let $\varphi(\mathbf{z}_t)$ be such a point.
3: Find the smallest update to the center such that $B(\mathbf{c}_{t+1}, r)$ touches $\varphi(\mathbf{z}_t)$.
4: Increment t by 1 and go back to Step 2.

3.1.1. Efficient Updating of the Ball Estimate

Unlike the CVM, the update in Step 3 can be performed efficiently without the use of any numerical optimization solver. Mathematically, we want to find the $\mathbf{c}$ such that $\min_{\mathbf{c}} \|\mathbf{c} - \mathbf{c}_t\|^2 \ : \ r^2 \geq \|\mathbf{c} - \varphi(\mathbf{z}_t)\|^2$. The Lagrangian is $\mathcal{L} = \|\mathbf{c} - \mathbf{c}_t\|^2 - \alpha(r^2 - \|\mathbf{c} - \varphi(\mathbf{z}_t)\|^2)$, where α is the Lagrangian multiplier. On setting its derivative to zero, and substituting the result back to the optimality condition $\|\mathbf{c} - \varphi(\mathbf{z}_t)\|^2 = r^2$, we obtain $\alpha = 1/\beta_t - 1$, where

$$\beta_t = r/\|\mathbf{c}_t - \varphi(\mathbf{z}_t)\| \ \ (\geq 0). \tag{5}$$

Thus, the new center is

$$\mathbf{c}_{t+1} = \varphi(\mathbf{z}_t) + \beta_t(\mathbf{c}_t - \varphi(\mathbf{z}_t)), \tag{6}$$

which is a convex combination of $\mathbf{c}_t$ and $\varphi(\mathbf{z}_t)$. Consequently, for any $t > 0$, $\mathbf{c}_t$ is always a combination of $\mathbf{c}_0$ and $\mathcal{S}_t = \{\varphi(\mathbf{z}_i)\}_{i=1}^t$. Note that this step is also similar to the update rule of the passive-aggressive algorithm (Crammer et al., 2006) used for online learning.

3.1.2. Efficient Distance Computations

After the update, Step 2 in the next iteration requires computing the distance between $\mathbf{c}_{t+1}$ and any pattern $\varphi(\mathbf{z})$. This can also be done efficiently. Note that

$$\begin{aligned}\|\mathbf{c}_{t+1}\|^2 &= \|\beta_t\mathbf{c}_t + (1-\beta_t)\varphi(\mathbf{z}_t)\|^2 \\ &= \beta_t\|\mathbf{c}_t\|^2 + (1-\beta_t)\|\varphi(\mathbf{z}_t)\|^2 + (\beta_t^2 - \beta_t)\|\mathbf{c}_t - \varphi(\mathbf{z}_t)\|^2,\end{aligned}$$

and so can be computed in $O(|\mathcal{S}_{t+1}|)$ time by caching $\|\mathbf{c}_t\|^2$ in the last iteration. Moreover, from (6), the expansion coefficients of $\mathbf{c}_{t+1}$ in terms of the $\varphi(\mathbf{z}_t)$'s can also be computed in $O(|\mathcal{S}_{t+1}|)$ time. Hence, $\|\mathbf{c}_{t+1} - \varphi(\mathbf{z})\|$ (for any $\mathbf{z}$) can be computed in $O(|\mathcal{S}_{t+1}|)$ time.

3.1.3. Convergence

As mentioned in Section 3.1, Algorithm 2 is modified from the MEB algorithm in (Panigrahy, 2004) by relaxing the requirement that $\varphi(\mathbf{z}_t)$ in Step 2 has to be the furthest point. Despite this, convergence of Algorithm 2 still follows from a weaker result in the proof of Theorem 1 in (Panigrahy, 2004). For completeness and later reference in the sequel, the proof is shown here.

Proposition 1. *Algorithm 2 obtains an* $(1+\epsilon)$*-approximation of EB*$(\mathcal{S}, r)$ *in* $O(1/\epsilon^2)$ *iterations.*

Proof. The update of $\mathbf{c}_t$ is shown in Figure 1. From (6), $\mathbf{c}_{t+1}$ must be along $\mathbf{c}_t$ and $\varphi(\mathbf{z}_t)$. As $\|\mathbf{c}_{t+1} - \varphi(\mathbf{z}_t)\| = r$, we have $\|\mathbf{c}_t - \mathbf{c}_{t+1}\| > \epsilon r$. Denote $\|\mathbf{c}_t - \mathbf{c}^*\|$ by δ_t. As $\|\mathbf{c}^* - \varphi(\mathbf{z}_t)\| \leq R^* \leq r$, $\angle\mathbf{c}^*\mathbf{c}_{t+1}\mathbf{c}_t$ is obtuse, and so $\delta_t^2 \geq \delta_{t+1}^2 + \|\mathbf{c}_t - \mathbf{c}_{t+1}\|^2 > \delta_{t+1}^2 + \epsilon^2 r^2$, or

$$\delta_{t+1}^2 < \delta_t^2 - \epsilon^2 r^2. \tag{7}$$

In other words, δ_t^2 decreases by at least $\epsilon^2 r^2$ in each iteration. Thus, the algorithm terminates in a finite number (say, T) of iterations. Summing (7) over the T iterations, we have $\delta_T^2 \leq \delta_0^2 - T\epsilon^2 r^2$ and so

$$T\epsilon^2 r^2 \leq \delta_0^2 \leq r^2. \tag{8}$$

Thus, $T \leq 1/\epsilon^2$. □

Moreover, it can be shown that Algorithm 2 only takes $O(1/\epsilon^4)$ time and $O(1/\epsilon^2)$ space (not including the $O(n)$ space for storing the training patterns). These are thus independent of the number of training examples for a fixed ϵ, and can also be seen to be lower than those of the CVM.

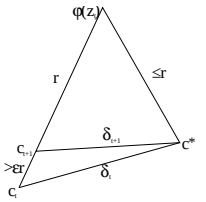


Figure 1. Update of $\mathbf{c}_t$ at the tth iteration.

3.2. Faster, Multi-Scale EB$(\mathcal{S}, r)$ Approximation

If the furthest point were used as $\varphi(\mathbf{z}_t)$ in Step 2 of Algorithm 2, a tighter analysis in (Panigrahy, 2004) shows that Algorithm 2 converges in $O(1/\epsilon)$, rather than $O(1/\epsilon^2)$, iterations. However, computing such a point takes $O(n|\mathcal{S}_t|)$ time and so is computationally expensive for large n.

In this section, we propose a faster version of Algorithm 2 that still only requires $\varphi(\mathbf{z}_t)$ to be any point lying outside $B(\mathbf{c}_t, (1+\epsilon)r)$. Assume that $2^{-1} \leq \epsilon = 2^{-M}$ for some positive integer M. The idea is that instead of using this small ϵ from the very beginning, we start with a much larger $\epsilon' = 2^{-1}$. After an $(1+\epsilon')$-approximation of EB$(\mathcal{S}, r)$ has been obtained by Algorithm 2, ϵ' is reduced by half and the process repeated until $\epsilon' = \epsilon$ (Algorithm 3).

Next, we first show the following proposition:

Proposition 2. *Let* $B(\hat{\mathbf{c}}, (1+\epsilon)r)$ *be any* $(1+\epsilon)$*-approximation of EB*$(\mathcal{S}, r)$*, then*

$$\frac{\|\hat{\mathbf{c}} - \mathbf{c}^*\|}{R^*} \leq \sqrt{\left(1+\sqrt{2}\right)\left((1+\epsilon)\frac{r}{R^*} - 1\right)}. \tag{9}$$

Algorithm 3 Multi-scale $(1+\epsilon)$-approximation algorithm for EB$(\mathcal{S}, r)$.

1: Initialize $\mathbf{c}_{\text{EB}_0} = \varphi(\mathbf{z}_0)$.
2: For $m = 1$ to M do
3: Set $\epsilon_m = 2^{-m}$. Find $(1+\epsilon_m)$-approximation of EB$(\mathcal{S}, r)$ using Algorithm 2, with $\mathbf{c}_{\text{EB}_{m-1}}$ as warm start.

Proof. Assume $\mathbf{c}^* \neq \hat{\mathbf{c}}$ (otherwise, (9) is trivial). Let $\mathcal{H}$ be the halfspace passing through $\mathbf{c}^*$ that is perpendicular to the line joining $\mathbf{c}^*$ and $\hat{\mathbf{c}}$ but not containing $\hat{\mathbf{c}}$. Using Lemma 2 in (Kumar et al., 2003), there is at least a point $\varphi(\mathbf{z})$ in $\mathcal{H}$ such that its distance to $\mathbf{c}^*$ is R^*. Let $\delta = \|\hat{\mathbf{c}} - \mathbf{c}^*\|$. Using cosine law, $\|\hat{\mathbf{c}} - \varphi(\mathbf{z})\| \geq \sqrt{(R^*)^2 + \delta^2} = R^*\sqrt{1 + \frac{\delta^2}{(R^*)^2}}$. As $B(\hat{\mathbf{c}}, (1+\epsilon)r)$ is an $(1+\epsilon)$-approximation, the distance between $\hat{\mathbf{c}}$ and every $\varphi(\mathbf{z}_i)$ is $\leq (1+\epsilon)r$. Also, $\frac{\delta}{R^*} \leq 1$. Thus, $(1+\epsilon)r \geq \|\hat{\mathbf{c}} - \varphi(\mathbf{z})\| \geq R^*\left(1 + \frac{\delta^2}{(1+\sqrt{2})(R^*)^2}\right)$ and so (9). □

In the special case where $r = R^*$ (i.e., the radius of the MEB is known in advance), (9) reduces to

$$\|\mathbf{c}_T - \mathbf{c}^*\| \leq R^*\sqrt{\left(1+\sqrt{2}\right)\epsilon}. \tag{10}$$

Moreover, on using (4), we have

$$\|\mathbf{c}_T - \mathbf{c}^*\|^2 = \|\mathbf{w}_T - \mathbf{w}^*\|^2 + (b_T - b^*)^2 + C\|\boldsymbol{\xi}_T - \boldsymbol{\xi}^*\|^2. \tag{11}$$

On using (11) and κ in (2), the difference between $f^*(\mathbf{x}) = \mathbf{w}^{*\prime}\varphi(\mathbf{x}) + b^*$ and $f_T(\mathbf{x}) = \mathbf{w}_T'\varphi(\mathbf{x}) + b_T$ is $|f_T(\mathbf{x}) - f^*(\mathbf{x})| \leq \sqrt{\|\mathbf{w}_T - \mathbf{w}^*\|^2 + (b_T - b^*)^2}\sqrt{\kappa+1} = \sqrt{\|\mathbf{c}_T - \mathbf{c}^*\|^2 - C\|\boldsymbol{\xi}_T - \boldsymbol{\xi}^*\|^2}\sqrt{\kappa+1}$. Thus, the bound in (10) guarantees that the prediction $f_T(\mathbf{x})$ is close to $f^*(\mathbf{x})$, particularly when ϵ is small.

Proposition 3. *Algorithm 3 obtains an $(1+\epsilon)$-approximation of EB$(\mathcal{S}, R^*)$ in $O(1/\epsilon)$ iterations.*

Proof. Let $\text{EB}_m = B(\mathbf{c}_{\text{EB}_m}, R^*)$ be an $(1+\epsilon_m)$-approximation of EB$(\mathcal{S}, R^*)$. Then, $\delta_0 \equiv \|\mathbf{c}_0 - \mathbf{c}^*\| = \|\mathbf{c}_{\text{EB}_{m-1}} - \mathbf{c}^*\|$. From (10), $\delta_0 \leq R^*\sqrt{(1+\sqrt{2})\epsilon_{m-1}}$. Using (8), we have

$$T_m\epsilon_m^2 r^2 \leq (1+\sqrt{2})\epsilon_{m-1}(R^*)^2 \leq (1+\sqrt{2})\epsilon_{m-1}r^2$$
$$\Rightarrow \quad T_m \leq (1+\sqrt{2})\epsilon_{m-1}/\epsilon_m^2 = (1+\sqrt{2})2^{m+1}.$$

The total number of iterations is then $\sum_{m=1}^{M} T_m = 2^2 + \sum_{m=2}^{M}(1+\sqrt{2})2^{m+1} = O(2^M) = O(1/\epsilon)$. □

This is thus an improvement over Algorithm 2, which converges in $O(1/\epsilon^2)$ iterations. Note, however, that while Proposition 1 holds for any $r \geq R^*$, Proposition 3 only holds for $r = R^*$. Nevertheless, it is obvious that Algorithm 3 always converges (as m takes finite values and each iteration of Algorithm 2 converges). We conjecture that for $r > R^*$, it converges in $O(1/\epsilon)$ to $O(1/\epsilon^2)$ iterations. We leave a formal analysis to a future occasion.

3.3. Obtaining a Smaller Enclosing Ball

With an initial r, we show in this section how a smaller EB can be obtained. The machinery is similar to that used in (Crammer et al., 2006). First, define $c = \sqrt{r^2 - (R^*)^2}$ and rewrite the distance constraint in the MEB problem (1) as $\|\mathbf{c} - \varphi(\mathbf{z})\|^2 + (r^2 - (R^*)^2) \leq r^2$. This can then be seen as an EB$(\tilde{\mathcal{S}}, r)$ problem $\|\tilde{\mathbf{c}} - \tilde{\varphi}(\mathbf{z}_i)\|^2 \leq r^2$, with $\tilde{\mathcal{S}} = \{\tilde{\varphi}(\mathbf{z}_i)\} = \{[\varphi(\mathbf{z}_i)'\ 0]'\}$ and center $\tilde{\mathbf{c}} = [\mathbf{c}'\ c]'$. After initializing $\tilde{\mathbf{c}}_0 = [\mathbf{c}_0'\ c_0]'$, where $\mathbf{c}_0 = \varphi(\mathbf{z}_0)$ for some $\varphi(\mathbf{z}_0) \in \mathcal{S}$ and $c_0 = r$, either Algorithm 2 or 3 can be used to obtain an $(1+\epsilon)$-approximation of this EB.

When it terminates at, say, the Tth iteration, all the points are inside $B(\tilde{\mathbf{c}}_T, (1+\epsilon)r)$. In other words, $\|\mathbf{c}_T - \varphi(\mathbf{z}_i)\|^2 \leq (1+\epsilon)^2r^2 - c_T^2$ for all $\varphi(\mathbf{z}_i)$'s. Note, on the other hand, that the original approximate EB problem only guarantees $\|\mathbf{c}_{T'} - \varphi(\mathbf{z}_i)\|^2 \leq (1+\epsilon)^2r^2$ on termination. As $c_T \geq 0$, we may now obtain a smaller ball of radius $r' = \sqrt{(1+\epsilon)^2r^2 - c_T^2}$. Indeed, if this $r' < r$, we can repeat the procedure and thus find an even smaller EB$(\mathcal{S}, r')$; otherwise, we should have $r \leq r'$ or $c_T^2 \leq (2\epsilon + \epsilon^2)r^2$ and we stop. The whole procedure is shown in Algorithm 4.

Algorithm 4 Obtaining a smaller enclosing ball.

1: Initialize $\tilde{\mathbf{c}}_0 = [\mathbf{c}_0'\ c_0]'$ where $\mathbf{c}_0 = \varphi(\mathbf{z}_0)$ and $c_0 = r$.
2: Use Algorithm 2 or 3 to obtain $B(\tilde{\mathbf{c}}_T, (1+\epsilon)r)$.
3: Terminate when $c_T^2 \leq (2\epsilon + \epsilon^2)r^2$.
4: Set $r = \sqrt{(1+\epsilon)^2r^2 - c_T^2}$, and go to Step 1.

3.4. Linking EB with SVM

We first address the question of how to set the EB's radius in the context of finding a SVM solution. Recall from Section 2 that the two-class L2-SVM using kernel k corresponds to a MEB problem with kernel $\tilde{k}$ defined in (3). As in (Tsang et al., 2005a), we assume that the kernel k satisfies condition (2). It is easy to see that $\tilde{k}_{ii} = \kappa + \frac{1}{C} \equiv \tilde{\kappa}$ is also a constant. In the corresponding EB$(\mathcal{S}_{\tilde{\varphi}}, r)$ problem, we can then set $r = \sqrt{\tilde{\kappa}}$ because of the following lemma.

Lemma 1. *$\sqrt{\kappa} \geq R^*$, where R^* is the radius of MEB($\mathcal{S}_{\tilde{\varphi}}$).*

Proof. From the KKT conditions of (1),

$$(R^*)^2 = \tilde{\kappa} - \sum_{i,j=1}^{n} \lambda_i\lambda_j k_{ij} \leq \tilde{\kappa} \tag{12}$$

as the kernel $\tilde{k}$ is psd. □

Noting that $\lambda_i^* \geq 0$ and $\sum_{i=1} \lambda_i^* = 1$, we have from (12) that $(R^*)^2 \geq \tilde{\kappa} - \frac{1}{n^2}\sum_{ij}\tilde{k}_{ij} \geq \tilde{\kappa} - \frac{1}{nC} - \frac{\ell^2}{n^2} - \frac{1}{n^2}\sum_{ij} k_{ij}$, where $\ell = \sum_i y_i$. Assume that the data $\mathbf{x} \in \mathbb{R}^D$ is generated from the normal distribution $N(\mathbf{0}, \sigma^2\mathbf{I})$. With the use of the Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$, we replace $\frac{1}{n^2}\sum_{ij} k_{ij}$ by $E[k(\mathbf{x}, \mathbf{x}')] = \frac{1}{3^{D/2}}$. Substituting back into the previous equation, we obtain $(R^*)^2 \geq \tilde{\kappa} - \frac{1}{nC} - \frac{\ell^2}{n^2} - \frac{1}{3^{D/2}}$. As $y_i \in \{\pm 1\}$, so $n \gg \ell$ unless the two classes are very imbalanced. When $n \to \infty$ and $D \to \infty$, we obtain $R^* \geq \sqrt{\tilde{\kappa}}$. Finally, on combining with Lemma 1, we have $R^* = \sqrt{\tilde{\kappa}}$. Thus, this justifies the choice of $r = \sqrt{\tilde{\kappa}}$. In the sequel, this approach of solving the SVM via the EB($\mathcal{S}_{\tilde{\varphi}}, \sqrt{\tilde{\kappa}}$) problem will be called the *ball vector machine* (BVM) procedure.

Moreover, with $r = \sqrt{\tilde{\kappa}} = R^*$, the bound in (9) reduces to that in (10). In other words, when ϵ is small, the center obtained from the $(1+\epsilon)$-approximation of this EB($\mathcal{S}_{\tilde{\varphi}}, \sqrt{\tilde{\kappa}}$) problem is close to the center of MEB($\mathcal{S}_{\tilde{\varphi}}$). Recall that one can use (4) to recover SVM's weight and bias from the ball's center. Thus, the obtained BVM will also be close to the truly optimal SVM solution and has comparable performance, especially as ϵ is typically small. This will be verified experimentally in Section 4.

4. Experiments

Experiments are performed on ten data sets[3] (Table 1). The proposed BVM (using Algorithm 4)[4] is compared with:[5] 1. CVM; 2. LIBSVM; 3. LASVM; 4. SimpleSVM. All are implemented in C++, except for SimpleSVM which is in Matlab. Hence, some measurements reported here may not be directly comparable with the SimpleSVM. Moreover, unless otherwise specified, SVM's C parameter is always set to 1. We use the Gaussian kernel $\exp(-\|\mathbf{x}-\mathbf{z}\|^2/\beta)$, where β is the average squared distance between training patterns. All experiments are performed on an AMD Athlon 4400+ PC with 4GB of RAM running Windows XP.

4.1. Varying ϵ

We compare BVM and CVM at different values of ϵ on a large (letter) and a very large (usps) data set. Figure 2 shows that both have high accuracies for $\epsilon \in [10^{-8} : 10^{-3}]$. When ϵ is increased further, their performance begins to deteriorate. Moreover, the training time and number of support vectors for both algorithms become stable for $\epsilon \leq 10^{-4}$. On both data sets, BVM and CVM have comparable numbers of support vectors, but training a BVM is faster than CVM (by almost an order of magnitude).

4.2. Varying C

We perform experiments on two medium-sized data sets (w3a and reuters) and a large data set (usps), with C in the range $[10^{-1} : 10^4]$. Results are shown in Figure 3. In general, BVM attains almost the same accuracies as LIBSVM for the different C's, except that it deteriorates on the w3a data when the relatively large $\epsilon = 10^{-3}$ is used. Moreover, the training time and number of support vectors obtained by BVM (with $\epsilon \geq 10^{-4}$) are comparable with those of LIBSVM, even though these data sets are only of medium size. On the other hand, LASVM also has comparable performance as LIBSVM for most of the C's, though it seems to deteriorate with large C's.

4.3. Varying the Training Set Size

We perform experiments on the web data with the standard partitioning (w1a, w2a, ..., w8a). Here, we show the results with $\epsilon = 10^{-3}$ and 10^{-4} (Figure 4). At $\epsilon = 10^{-4}$, both BVM and CVM have comparable testing accuracies as the other implementations; whereas a larger $\epsilon = 10^{-3}$ leads to poor performance (as in Sections 4.1 and 4.2).

As the web data is not very large, almost all implementations scale well except for the SimpleSVM which has to be terminated early. We speculate that this might be partly because a Matlab implementation is used.

4.4. Comparisons on all Data Sets

We perform experiments on all data sets in Table 1. By default, we set $C = 1$ which yields good performance except for the intrusion data where we have $C = 10^6$ instead. As the previous sections have indicated that $\epsilon = 10^{-3}$ is not appropriate for the BVM, we only report ϵ values of $10^{-4}, 10^{-5}$, and 10^{-6} here. Methods that cannot be run (either because of not enough memory and/or the training time is longer than 10,000 seconds) are indicated by "–".

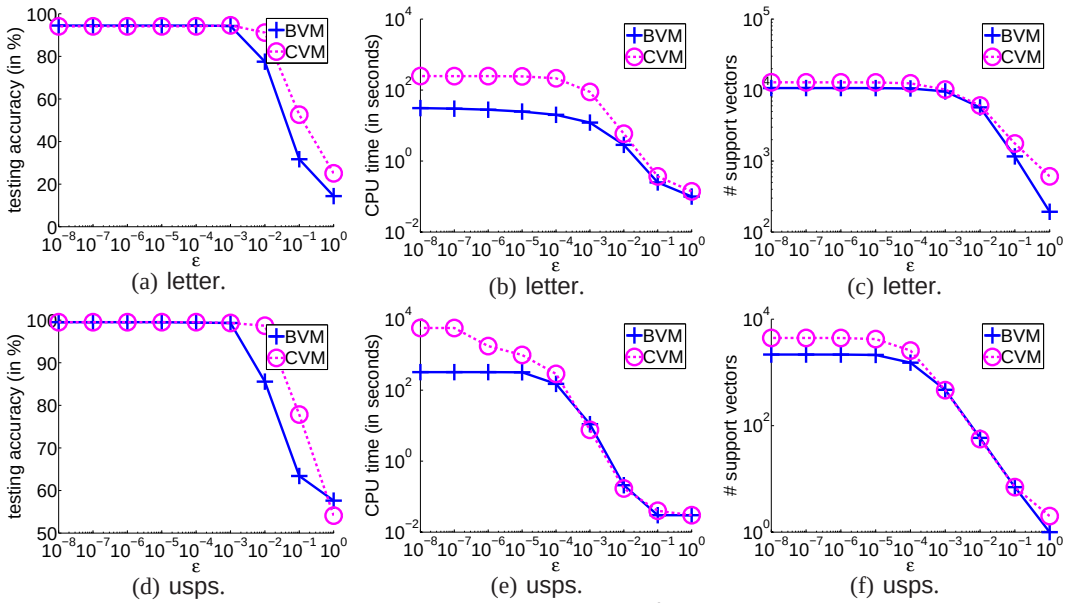
Table 2 shows that BVM and CVM have accuracies comparable with the other SVM implementations[6]. In terms of the training speed, Table 3 shows that BVM is usually faster than CVM for the same value of ϵ, and is faster / comparable with the other implementations. On usps (the second largest data set), BVM is faster than LIBSVM and LASVM

[3] optdigits, satimage, pendigits are from UCI machine learning repository; reuters from http://www.daviddlewis.com/resources/testcollections/reuters21578/; w3a, letters, web (obtained by combining w1a to w8a), ijcnn1 are from http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/; usps and intrusion are from http://www.cse.ust.hk/∼ivor/cvm.html.

[4] For simplicity, there is only one iteration of Algorithm 4 in the current implementation.

[5] CVM (v1.1) is from http://www.cse.ust.hk/∼ivor/cvm.html, LIBSVM (v2.83) from http://www.csie.ntu.edu.tw/∼cjlin/libsvm/, LASVM (v1.0) from http://leon.bottou.org/projects/lasvm, and SimpleSVM (v2.31) from http://asi.insa-rouen.fr/∼gloosli/simpleSVM.html.

[6] The LASVM implementation does not support multi-class classification, and so the corresponding entries in Tables 2 – 4 are marked "N/A".



(a) letter. (b) letter. (c) letter.

(d) usps. (e) usps. (f) usps.

Figure 2. Performance at different ϵ's on letter and usps.

even with $\epsilon = 10^{-6}$. Moreover, only BVM and CVM (but neither LIBSVM nor LASVM) can work on intrusion, the largest data set used in the experiment. In contrast, SimpleSVM can work on medium-sized data sets only, as its rank-one update involves the inverse of a kernel sub-matrix defined on all the support vectors. This causes problems when there are a lot of support vectors. Note that the accuracy of BVM is quite insensitive to the value of ϵ (in our range $[10^{-6} : 10^{-4}]$), but both its training time and number of support vectors increase as ϵ gets smaller. This suggests $\epsilon = 10^{-4}$ is a reasonable choice in practice.

Table 1. Data sets used in the experiments.

data sets	#class	dim	#train patns.	#test patns
optdigits	10	64	3,823	1,797
satimage	6	36	4,435	2,000
w3a	2	300	4,912	44,837
pendigits	10	16	7,494	3,498
reuters	2	8,315	7,770	3,299
letter	26	16	15,000	5,000
web	2	300	49,749	14,951
ijcnn1	2	22	49,990	91,701
usps	2	676	266,079	75,383
intrusion	2	127	4,898,431	311,029

The numbers of support vectors obtained[7] are shown in Table 4. As can be seen, all of them obtain comparable numbers of support vectors. In particular, LIBSVM, which uses second order information for working set selection, can identify a smaller number of support vectors on most medium-sized data sets. On the large data sets (such as reuters, web, ijcnn1, usps and intrusion), CVM and, even better, BVM can have fewer support vectors.

[7]For multi-class problems, the same pattern may appear as support vectors in different binary classifiers in the current SimpleSVM implementation. To avoid confusion, the number of support vectors obtained by SimpleSVM is not reported here.

5. Conclusion

Motivated by the minimum enclosing ball (MEB) problem, we introduce in this paper the easier enclosing ball (EB) problem where the ball's radius is fixed. Three efficient $(1 + \epsilon)$-approximation algorithms are developed that are simple to implement and do not require any numerical solver. For the Gaussian kernel in particular, a suitable choice of this (fixed) radius is easy to determine, and the center obtained from the $(1 + \epsilon)$-approximation of this EB problem is expected to be close to the center of the corresponding MEB. Experimental results show that the proposed algorithm has accuracies comparable to the other large-scale SVM implementations, but can handle very large data sets and is even faster than the CVM in general.

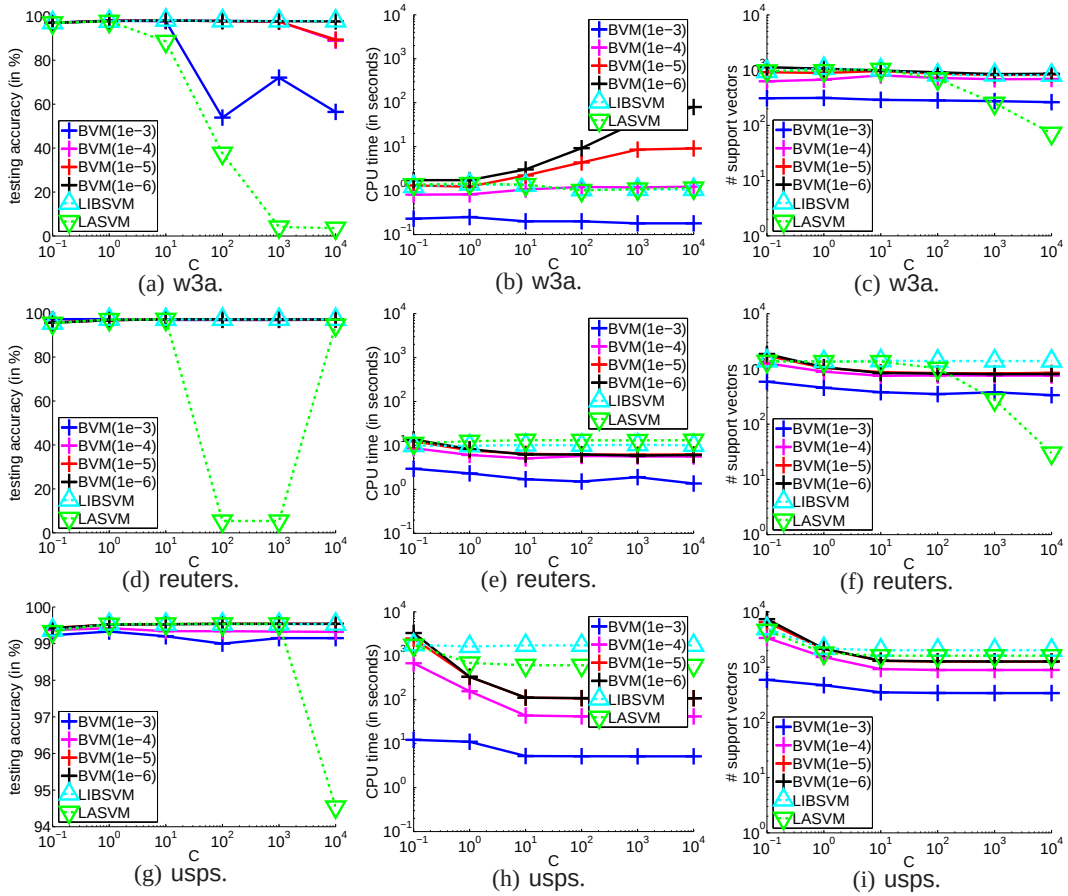
Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant 615005.

References

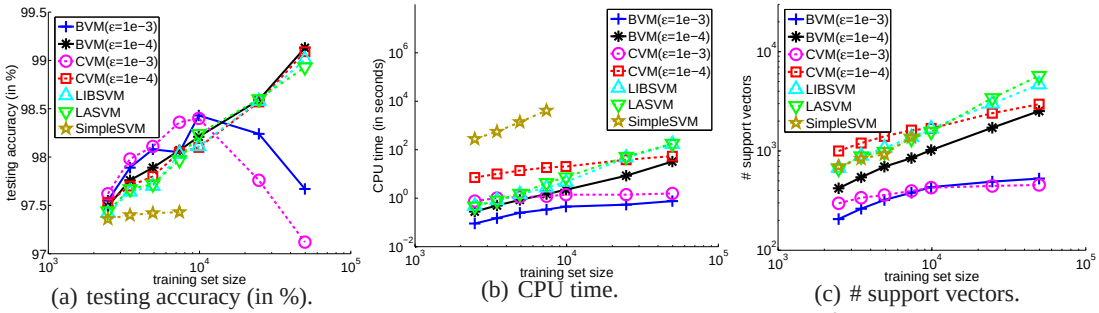
Asharaf, S., Murty, M., & Shevade, S. (2006). Cluster based core vector machine. *Proceedings of the Sixth International Conference on Data Mining* (pp. 1038–1042).

Bordes, A., Ertekin, S., Weston, J., & Bottou, L. (2005).



(a) w3a. (b) w3a. (c) w3a.
(d) reuters. (e) reuters. (f) reuters.
(g) usps. (h) usps. (i) usps.

Figure 3. Performance at different C's on w3a, reuters and usps.



(a) testing accuracy (in %). (b) CPU time. (c) # support vectors.

Figure 4. Performance with different training set sizes on web.

Table 2. Testing accuracies (in %) of the various SVM implementations.

data	BVM			CVM			LIBSVM	LASVM	SimpleSVM
	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$			
optdigits	96.38	96.38	96.38	96.38	96.38	96.38	96.77	N/A	**96.88**
satimage	89.35	89.60	89.60	89.55	89.55	89.60	89.55	N/A	**89.65**
w3a	**97.89**	97.88	97.88	97.80	97.81	97.82	97.70	97.72	97.42
pendigits	**97.97**	97.94	97.91	97.85	97.85	97.85	97.91	N/A	**97.97**
reuters	96.75	96.78	96.75	96.96	96.96	96.96	**97.15**	97.09	–
letter	**94.47**	**94.47**	**94.47**	94.12	94.10	94.10	94.25	N/A	94.23
web	**99.13**	**99.13**	99.08	99.09	99.07	99.08	99.01	98.93	–
ijcnn1	97.58	98.38	98.25	**98.67**	98.11	98.23	98.19	98.42	94.10
usps	99.42	99.52	99.52	99.52	99.52	99.51	**99.53**	**99.53**	–
intrusion	91.97	91.97	91.97	81.68	**92.44**	**92.44**	–	–	–

Table 3. CPU time (in seconds) used in SVM training.

data	BVM			CVM			LIBSVM	LASVM	SimpleSVM
	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$			
optdigits	**1.65**	1.76	1.87	24.86	26.07	26.25	1.79	N/A	81.15
satimage	1.82	2.67	3.54	14.81	19.15	19.67	**1.06**	N/A	221.01
w3a	**0.85**	1.31	1.75	13.82	30.71	35.12	1.46	1.54	1384.34
pendigits	1.31	1.40	1.45	12.10	13.31	13.43	**0.82**	N/A	41.22
reuters	**6.32**	7.78	8.25	63.51	136.04	162.89	9.76	13.81	–
letter	19.87	24.84	28.40	215.73	244.25	250.43	**10.85**	N/A	1290.55
web	**32.59**	235.26	576.95	54.46	670.28	1235.32	168.84	178.73	–
ijcnn1	99.95	1019.58	2385.14	62.78	433.35	784.78	**57.96**	140.25	2201.35
usps	**150.46**	319.20	324.17	288.96	998.96	1753.12	1578.27	753.09	–
intrusion	0.73	0.73	0.73	**0.51**	0.84	0.70	–	–	–

Table 4. Numbers of support vectors obtained by the various SVM implementations.

data	BVM			CVM			LIBSVM	LASVM
	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$		
optdigits	1583	1594	1595	2154	2191	2197	**1306**	N/A
satimage	1956	2048	2058	2333	2581	2611	**1433**	N/A
w3a	**694**	952	1060	1402	2118	2269	1072	979
pendigits	1990	2006	2008	2827	2925	2926	**1206**	N/A
reuters	**925**	1059	1076	1496	2217	2389	1356	1359
letter	10536	10658	10673	12440	12820	12843	**8436**	N/A
web	**2522**	4786	6263	2960	9986	12984	4674	5718
ijcnn1	4006	7409	7981	**3637**	9041	11097	5700	5525
usps	**1524**	2128	2163	2576	4224	4429	2178	1803
intrusion	99	100	100	**26**	44	51	–	–

Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, *6*, 1579–1619.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, *19*, 1155–1178.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Har-Peled, S., Roth, D., & Zimak, D. (2007). Maximum margin coresets for active and noise tolerant learning. *Proceedings of International Joint Conference on Artificial Intelligence* (pp. 836–841).

Joachims, T. (2006). Training linear SVMs in linear time. *Proceedings of the International Conference on Knowledge Discovery and Data Mining* (pp. 217–226).

Kumar, P., Mitchell, J., & Yildirim, A. (2003). Approximate minimum enclosing balls in high dimensions using coresets. *ACM Journal of Experimental Algorithmics*, *8*.

Nock, R., & Nielsen, F. (2005). Fitting the smallest enclosing Bregman ball. *Proceedings of the 16th European Conference on Machine Learning* (pp. 649–656).

Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithm for support vector machines. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing* (pp. 276–285).

Panigrahy, R. (2004). Minimum enclosing polytope in high dimensions. *CoRR cs.CG/0407020*.

Shevade, S., & Chu, W. (2006). Minimum enclosing spheres formulations for support vector ordinal regression. *Proceedings of the Sixth International Conference on Data Mining* (pp. 1054–1058).

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.

Tsang, I. W., & Kwok, J. T. (2007). Large-scale sparsified manifold regularization. *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press.

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005a). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, *6*, 363–392.

Tsang, I. W., Kwok, J. T., & Lai, K. T. (2005b). Core vector regression for very large regression problems. *Proceedings of the Twenty-Second International Conference on Machine Learning* (pp. 913–920).

Vishwanathan, S., Smola, A., & Murty, M. (2003). SimpleSVM. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 760–767).