An Efficient Method for Simplifying Support Vector Machines

DucDung Nguyen DUNGDUC@JAIST.AC.JP
TuBao Ho BAO@JAIST.AC.JP
Japan Advanced Institute of Science and Technology, 1-1, Asahidai, Tatsunokuchi, Ishikawa 923-1292 Japan

Abstract

In this paper we describe a new method to reduce the complexity of support vector machines by reducing the number of necessary support vectors included in their solutions. The reduction process iteratively selects two nearest support vectors belonging to the same class and replaces them by a newly constructed vector. Through the analysis of relation between vectors in the input and feature spaces, we present the construction of new vectors that requires to find the unique maximum point of a one-variable function on the interval $(0, 1)$, not to minimize a function of many variables with local minimums in former reduced set methods. Experimental results on real life datasets show that the proposed method is effective in reducing number of support vectors and preserving machine's generalization performance.

1. Introduction

Support Vector Machines (SVMs) (Vapnik, 1995; Cristianini & Shawe-Taylor, 2000) have been found to be very robust in many applications, such as optical character recognition (LeCun et al., 1995; Liu et al., 2003), text categorization (Joachims, 1998), face detection in images (Osuna et al., 1997). The high generalization ability of SVMs is ensured by special properties of the optimal hyperplane that maximizes the distance from it to the training patterns in a high dimensional feature space (Cortes & Vapnik, 1995; Boser et al., 1992; Vapnik, 1995). However SVMs are considerably slower in the test phase than other learning methods like decision trees or neural networks (Burges, 1996; Burges, 1998; Burges & Schoelkopf, 1997; LeCun et al., 1995; Liu et al., 2003).

Appearing in *Proceedings of the 22^{nd} International Conference on Machine Learning*, Bonn, Germany, 2005.

The solution of a SVM is parameterized by a set of input vectors called support vectors (SVs) and their corresponding weights. When a new test example is introduced, SVMs compare it with these SVs via kernel calculations; this computation becomes very expensive if the number of SVs is large. To reduce this computational complexity, reduced set methods, e.g., (Burges, 1996; Schoelkopf & Smola, 2002), try to approximate the original SVM by another comprised by a much smaller number of newly constructed vectors, called the reduced vectors set. The former methods described in (Burges, 1996; Schoelkopf et al., 1999; Schoelkopf & Smola, 2002) start from approximating the solution comprised by all original SVs by only one new vector, and then incrementally construct the reduced set by finding vectors that minimize the differences between the original vector expansion and the reduced set expansion in feature space. This approach leads to the construction of each new vector required to solve an unconstrained optimization problem in a space of $d+1$ variables, where d is the dimension of the input space. Hence the computation is very expensive because the search must be repeated many times with different initial points to escape from local minimums (Burges, 1996; Mika et al., 1999; Schoelkopf et al., 1999).

In this paper we describe a bottom-up method to simplify support vector solutions in which the construction of new vectors only requires to find the unique maximum point of a one-variable function on (0,1). Instead of constructing the reduced vectors set incrementally, the two nearest SVs belonging to the same class will be iteratively considered and replaced by a newly constructed vector. This approach leads to a conceptually simpler and computationally less expensive method, the local extremum problem does not exist, and it also makes the vectors in the simplified solution look more meaningful (e.g. character-like in OCR applications). Experimental results on real life datasets show the effectiveness of our proposed method in reducing the number of support vectors and preserving generalization performance. On the US Postal Ser-

vice (USPS) handwritten digit recognition database, a 91.3% (for polynomial kernel) and 90.0% (for Gaussian kernel) reduction rate can be achieved, with a corresponding 0.2% and 0.3% loss in predictive accuracy. On the MNIST database, the numbers are 88.6% reduction rate and 0.1% loss in accuracy. The reduction rates on other four datasets in the StatLog collection are from 70.9% to 94.5% with almost no change in performance.

The remainder of this paper is organized as follows. We briefly describe the SVM simplification problem and review the earlier reduced set methods in section 2. In section 3 we describe our proposed method that constructs a new vector to replace two other support vectors, and the iterative process to simplify support vector solutions. Experiments are described in section 4. Section 5 discusses the results.

2. Reduced Support Vector Machines

2.1. Support Vector Machines

SVMs work in feature space indirectly via a kernel function $K(x, y) = \Phi(x) \cdot \Phi(y)$ where $\Phi : R^d \rightarrow F$ is a map from the d-dimensional input space to a possibly high-dimensional feature space (Vapnik, 1995). For a two-class classification problem, the decision rule takes the form

$$y = sign\left(\sum_{i=1}^{N_S} \alpha_i K(x, x_i) + b\right) \tag{1}$$

where α_i are weights of support vectors x_i, x is the input vector needed to classify, $K(x, x_i) = \Phi(x) \cdot \Phi(x_i)$ is a kernel function calculating the dot product of two vectors $\Phi(x)$ and $\Phi(x_i)$ in the feature space, b is the bias, and N_S is the number of support vectors. The task of the SVMs training process is to determine all the parameters (x_i, α_i, b, N_S); the resulting x_i, $i = 1...N_S$ are a subset of the training set and are called *support vectors*.

2.2. Reduced Set Methods

The reduced set methods try to approximate the normal vector Ψ of the separating hyperplane

$$\Psi = \sum_{i=1}^{N_S} \alpha_i \Phi(x_i) \tag{2}$$

expanded in images of input vectors $x_i \in R^d$, $\alpha_i \in R$, by a reduced set expansion

$$\Psi' = \sum_{i=1}^{N_Z} \beta_i \Phi(z_i) \tag{3}$$

with $N_Z < N_S$, $z_i \in R^d$, $\beta_i \in R$. To classify a new test point x, calculation (1) is replaced by

$$y = sign\left(\sum_{i=1}^{N_Z} \beta_i K(x, z_i) + b\right) \tag{4}$$

The goal of the reduced set methods is to choose the smallest number $N_Z < N_S$, and the corresponding reduced set $\{(z_i, \beta_i)\}_{i=1...N_Z}$ such that any resulting loss in generation performance remains acceptable.

The method described in (Burges, 1996) starts by replacing the original expansion Ψ with the image of one input vector and its corresponding weight (z_1, β_1), and then iteratively finds (z_{m+1}, β_{m+1}) so that their images approximate the complement vectors Ψ_m ($\Psi_0 = \Psi$)

$$\Psi_m = \sum_{i=1}^{N_S} \alpha_i \Phi(x_i) - \sum_{j=1}^{m} \beta_j \Phi(z_j) \tag{5}$$

Because in many situations it is impossible to find exactly the z_m and β_m that make $\Psi_m = 0$, (e.g. the chosen kernel is a Gaussian RBF), z_m are pre-images that minimize

$$\rho = \|\Psi_{m-1} - \beta_m \Phi(z_m)\|^2 \tag{6}$$

In general, an unconstrained optimization technique is used to find the minimum of ρ. For a particular kind of kernel $K(x, y) = K(\|x - y\|^2)$ the fixed-point iteration method can be used to improve the speed of the finding (Schoelkopf et al., 1999; Schoelkopf & Smola, 2002).

The drawback of the above methods is that they may suffer from numerical instability and get trapped in a local minimum of function ρ; to prevent this circumstance, the finding for each new vector must be repeated many times with different initial values (Burges, 1996; Mika et al., 1999). In the next section we describe a simpler and more efficient method for simplifying support vector solutions.

3. The Bottom-up Method

3.1. Simplification of Two Support Vectors

The solution of SVMs can be analyzed from a mechanical point of view: if each image of support vectors exerts a force $F_i = \alpha_i \hat{\Psi}$ on the decision hyperplane, then the SVMs solution satisfies the conditions of equilibrium: the sum of the forces and the torque all vanish ($\hat{\Psi}$ is the unit vector in the direction Ψ)(Burges, 1998). In an equilibrium system, if we replace two member forces by an equivalent one, then the equilibrium state of the system will not change. In a SVM

solution, if we replace two images $\Phi(x_i)$ and $\Phi(x_j)$ of two support vectors belonging to the same class x_i and x_j by a vector $M = m\Phi(x_i) + (1-m)\Phi(x_j)$, where $m = \alpha_i/(\alpha_i + \alpha_j)$ and weight vector M by $\alpha_m = (\alpha_i + \alpha_j)$, then for any point x in the input space, calculation (1) can be computed through $(N_S - 1)$ vectors:

$$y = sign\left(\sum_{k=1,k\neq i,k\neq j}^{N_S} \alpha_k K(x, x_k) + \alpha_m M \cdot \Phi(x) + b\right) \tag{7}$$

The difficulty is that M can not be used directly; we must use its pre-image, and in many situations, we cannot find the exact pre-image of M. This problem was addressed in (Schoelkopf & Smola, 2002; Kwok & Tsang, 2003) as the pre-image problem in kernel methods.

Rather than trying to find the exact pre-image, we will approximate M by the image $\Phi(z)$ of some input vector z. The optimal approximation can be made if we choose a vector z that gives a minimum value of $\|M - \Phi(z)\|^2$, or in other words, we have to solve the optimization problem:

$$\min_z \|M - \Phi(z)\|^2 \tag{8}$$

The following propositions will give us the way to find vector z efficiently. All that is required is to find the unique maximum point of a one-variable function on (0,1). The coefficient of z then can be calculated analytically.

Proposition 1 *For Gaussian RBF kernels $K(x, y) = \exp(-\gamma \|x - y\|^2)$, the 2-norm optimal approximation of $M = m\Phi(x_i) + (1-m)\Phi(x_j)$, $m = \alpha_i/(\alpha_i + \alpha_j)$, $\alpha_i\alpha_j > 0$, is the image of input vector z determined by*

$$z = kx_i + (1-k)x_j \tag{9}$$

where k is the maximum point of

$$f(k) = mC_{ij}^{(1-k)^2} + (1-m)C_{ij}^{k^2} \tag{10}$$

with $C_{ij} = K(x_i, x_j)$

Proof: For Gaussian RBF kernels, Φ maps each input vector onto the surface of the unit hypersphere in the feature space, so we have $\|\Phi(z)\| = 1$ for every z, $\|M\|$ is a constant and can be calculated via $\Phi(x_i)$ and $\Phi(x_j)$. (8) is equivalent to

$$\max_z M \cdot \Phi(z) \tag{11}$$

For the extremum, we have $0 = \nabla_z(M \cdot \Phi(z))$. To get the gradient in terms of K, we substitute $M = m\Phi(x_i) + (1-m)\Phi(x_j)$ and $K(x, y) = \exp(-\gamma \|x - y\|^2)$ to get the sufficient condition

$$\begin{aligned} 0 &= \nabla_z(M \cdot \Phi(z)) \\ &= 2m\exp(-\gamma\|x_i - z\|^2)(x_i - z) \\ &\quad +2(1-m)\exp(-\gamma\|x_j - z\|^2)(x_j - z) \end{aligned} \tag{12}$$

leading to

$$z = \frac{\sum_{s=i,j} \alpha_s \exp(-\gamma\|x_s - z\|^2)x_s}{\sum_{s=i,j} \alpha_s \exp(-\gamma\|x_s - z\|^2)} \tag{13}$$

or

$$z = kx_i + (1-k)x_j \tag{14}$$

where

$$k = \frac{\alpha_i \exp(-\gamma\|x_i - z\|^2)}{\sum_{s=i,j} \alpha_s \exp(-\gamma\|x_s - z\|^2)} \tag{15}$$

Because $\alpha_i\alpha_j > 0$ (or x_i and x_j belong to the same positive or negative class) then $0 < k < 1$. (14) means that z always lies on the segment connecting x_i and x_j. To ease the finding of z we define $f(k) = M \cdot \Phi(z)$ and search for the maximum point of $f(k)$

$$\begin{aligned} f(k) &= M \cdot \Phi(z) \\ &= M \cdot \Phi(kx_i + (1-k)x_j) \\ &= m\exp(-\gamma\|x_i - x_j\|^2(1-k)^2) \\ &\quad +(1-m)\exp(-\gamma\|x_i - x_j\|^2 k^2) \\ &= mC_{ij}^{(1-k)^2} + (1-m)C_{ij}^{k^2} \end{aligned} \tag{16}$$

where $C_{ij} = \exp(-\gamma\|x_i - x_j\|^2) = K(x_i, x_j)$

$f(k)$ is a one-variable function and has a unique maximum point on $(0, 1)$ (as illustrated in Figure 1). The maximum point can be easily reached using common univariate parameter optimization methods. In our experiments, the inverse parabolic interpolation method (Press et al., 2002) was used with three starting points

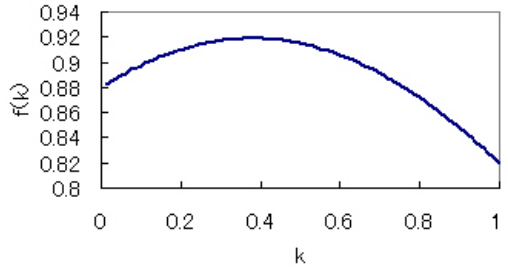

Figure 1. $f(k) = mC_{ij}^{(1-k)^2} + (1-m)C_{ij}^{k^2}$ with $m = 0.4, C_{ij} = 0.7$.

$k = 0, k = m$ and $k = 1$, and the optimization process converges quickly after several iterations (if $m = 1/2$ then $k = m = 1/2$ is exactly the maximum point of $f(k)$).

Proposition 2 *For polynomial kernels $K(x, y) = (x \cdot y)^p$, the 2-norm optimal approximation of $M = m\Phi(x_i) + (1-m)\Phi(x_j)$, $m = \alpha_i/(\alpha_i + \alpha_j)$, $\alpha_i\alpha_j > 0$, is the image of input vector z determined by*

$$z = \frac{\|M\|^{1/p}}{\|z^*\|} z^* \tag{17}$$

where $z^* = kx_i + (1-k)x_j$ and k is the maximum point of $h(k)$

$$h(k) = \|M\|\, u(k)v(k), \tag{18}$$

where

$$u(k) = \frac{1}{\left[x_i^2 k^2 + 2(x_i \cdot x_j)k(1-k) + x_j^2(1-k)^2\right]^{p/2}} \tag{19}$$

$$\begin{aligned} v(k) &= m\left[x_i^2 k + (x_i \cdot x_j)(1-k)\right]^p \\ &\quad + (1-m)\left[(x_i \cdot x_j)k + x_j^2(1-k)\right]^p \end{aligned} \tag{20}$$

Proof: For polynomial kernels, Φ maps each input vector x lying on the surface of a hypersphere of radius r ($\|x\| = r$) onto the surface of a hypersphere of radius r^{2p} in the feature space ($(r^2+1)^p$ for inhomogeneous kernel $K(x, y) = (x \cdot y + 1)^p$). To approximate M by $\Phi(z)$ we can constrain $\Phi(z)$ to lay on the surface of the same hypersphere with M in feature space without any lost in generality. This is equivalent to constraining z to lie on the surface of the hypersphere of radius $\|M\|^{1/p}$ in the input space, and (8) becomes

$$\max_z M \cdot \Phi(z) \tag{21}$$

subject to

$$\|z\| = \|M\|^{1/p} \tag{22}$$

The following lemma shows that the (vector) solution of (21), x_i, and x_j are linearly dependent

Lemma 1 *The input vector z that maximizes $M \cdot \Phi(z)$ in (21) is linearly dependent with x_i and x_j.*

Proof: Replace $M = m\Phi(x_i) + (1-m)\Phi(x_j)$ into (21) we have

$$\begin{aligned} M \cdot \Phi(z) &= (m\Phi(x_i) + (1-m)\Phi(x_j)) \cdot \Phi(z) \\ &= m(x_i \cdot z)^p + (1-m)(x_j \cdot z)^p \end{aligned} \tag{23}$$

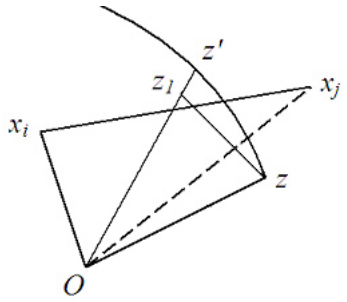

Figure 2. Projection of vector z on the plane (x_i, x_j) in the input space.

Suppose that z is an input vector satisfying constraint (22) and z_1 is the orthogonal projection of z on the plane determined by x_i and x_j (as described in Figure 2). Let's consider input vector z'

$$z' = \frac{\|z\|}{\|z_1\|} z_1 \tag{24}$$

We have: z' satisfies constraint (22) and $x_i \cdot z' \geq x_i \cdot z$, $x_j \cdot z' \geq x_j \cdot z$, or $M \cdot \Phi(z') \geq M \cdot \Phi(z)$. This means that the optimal vector z_{opt} for maximizing $M \cdot \Phi(z)$ lies on the plane (x_i, x_j), or z_{opt} is linearly dependent with x_i and x_j.

Because the solution of (21), called z_{opt}, lies on the plane (x_i, x_j) and $\|z_{opt}\| = \|M\|^{1/p}$, there exits a vector z^* and a scalar k such that

$$z^* = kx_i + (1-k)x_j \tag{25}$$

and

$$z_{opt} = \frac{\|M\|^{1/p}}{\|z^*\|} z^* \tag{26}$$

Call $g(z) = M \cdot \Phi(z)$, we have

$$\begin{aligned} g(z_{opt}) &= M \cdot \Phi(z_{opt}) \\ &= m(x_i \cdot z_{opt})^p + (1-m)(x_j \cdot z_{opt})^p \\ &= m(\|x_i\|\,\|z_{opt}\|\cos(x_i, z_{opt}))^p \\ &\quad + (1-m)(\|x_j\|\,\|z_{opt}\|\cos(x_j, z_{opt}))^p \\ &= \|z_{opt}\|^p \left[m\|x_i\|^p \left(\frac{x_i \cdot z^*}{\|x_i\|\,\|z^*\|} \right)^p \right. \\ &\quad \left. + (1-m)\|x_j\|^p \left(\frac{x_j \cdot z^*}{\|x_j\|\,\|z^*\|} \right)^p \right] \\ &= \frac{\|z_{opt}\|^p}{\|z^*\|^p} \left[m(x_i \cdot z^*)^p + (1-m)(x_j \cdot z^*)^p \right] \end{aligned} \tag{27}$$

Because z_{opt} satisfies (22) then $\|z_{opt}\|^p = \|M\|$. Re-

placing $z^* = kx_i + (1-k)x_j$ into (27) leads to

$$h(k) = \|M\| \, u(k)v(k) \tag{28}$$

where $u(k)$ and $v(k)$ are defined in (19) and (20).

$h(k)$ is also an one-variable function and has unique maximum point on $(0, 1)$ (corresponding to the unique vector z' in (24)). This means that the finding of each new vector in the reduced set is much easier and cheaper than that in former methods (in the space of $d+1$ variables with local minimums).

Proposition 3 *The optimal coefficient β for approximating $\alpha_m M = \alpha_i \Phi(x_i) + \alpha_j \Phi(x_j)$ by $\beta\Phi(z)$ is*

$$\beta = \frac{\alpha_m M \cdot \Phi(z)}{\|\Phi(z)\|^2} \tag{29}$$

Proof: Once we replace x_i and x_j by z, or approximate M by $\Phi(z)$ in the feature space, the difference between two solutions will be, for every input vector x

$$\begin{aligned} d(\beta) &= |\alpha_m M \cdot \Phi(x) - \beta\Phi(z) \cdot \Phi(x)| \\ &= |(\alpha_m M - \beta\Phi(z)) \cdot \Phi(x)| \end{aligned} \tag{30}$$

The difference will be minimized when $d(\beta)$ gets the minimum value. In (30) $d(\beta)$ can be minimized by minimizing $d_1(\beta) = \|\alpha_m M - \beta\Phi(z)\|$. Because $d_1(\beta)$ is a quadratic function of β, its minimum point is at

$$\beta = \frac{\alpha_m M \cdot \Phi(z)}{\|\Phi(z)\|^2} \tag{31}$$

Equation (29) is used to find the coefficient for one newly constructed vector. For the whole reduced vectors set, the following proposition is used to recompute all the coefficients to get a better approximation.

Proposition 4 ((Schoelkopf et al., 1999)) *The optimal coefficients $\beta = (\beta_1, ..., \beta_{N_Z})$ for approximating $\Psi = \sum_{i=1}^{N_S} \alpha_i \Phi(x_i)$ by $\Psi' = \sum_{j=1}^{N_Z} \beta_j \Phi(z_j)$ (for linearly independent $\Phi(z_1), ..., \Phi(z_{N_Z})$) are given by*

$$\beta = (\mathbf{K}^z)^{-1} \mathbf{K}^{zx} \alpha \tag{32}$$

where $\mathbf{K}^z_{ij} = \Phi(z_i) \cdot \Phi(z_j)$ and $\mathbf{K}^{zx}_{ij} = \Phi(z_i) \cdot \Phi(x_j)$

As mentioned in (Schoelkopf et al., 1999), (32) always gives the optimal coefficients to get a solution that is at least as good as the original one. In our experiments, (32) was used to recompute the final coefficients of all vectors in the reduced set after the iterative simplification process finished.

3.2. Simplification of Support Vector Solution

The simplification procedure iteratively replaces two support vectors (including newly created vectors) x_i and x_j by a new vector z using the method described in section 3.1. This process can be viewed as a bottom-up hierarchical clustering procedure, and there are two problems we have to take into consideration. First, how to select a good pair of support vectors to simplify, and second, when the simplification process will stop.

3.2.1. Selection heuristic

In general, a pair of two support vectors that gives a minimum value of $d(\beta)$ in (30) will produce the minimum difference between two solutions (solutions at two consecutive steps). However, the cost of using this criterion is rather expensive because we have to try all possible pairs of support vectors and then evaluate (30) for each of them. Moreover, we are more concerned about the original solution and the final one, so the strictly good approximation of the solutions at every intermediate steps is not necessary. The alternative heuristic is based on the difference between two vectors $M = m\Phi(x_i) + (1-m)\Phi(x_j)$ and $\Phi(z)$ in (8). For Gaussian RBF kernels, we can select x_i and x_j that give a maximum value of $C_{ij} = K(x_i, x_j)$ in (10) because the bigger the C_{ij}, the bigger the maximum value of $f(k)$, or smaller difference between M and $\Phi(z)$ ($f(k) = 1$ gives zero difference). This is equivalent to selecting two closest support vectors belonging to the same positive or negative class. Another interpretation for this selection heuristic is that we will try to approximate two Gaussian RBFs by one Gaussian RBF, and intuitively, the closer pair centers, the better approximation. This selection heuristic also be reasonably applied to polynomial kernels because the input vector z that maximizes $M \cdot \Phi(z)$ in (23) is linearly dependent with x_i and x_j and the closer two vectors x_i and x_j (or smaller angle between two vectors x_i and x_j) will give a bigger maximum value of $M \cdot \Phi(z)$ (given fixed values of m, $\|x_i\|$, and $\|x_i\|$).

3.2.2. Stopping condition

The simplified solution is mostly different from the original one, so the simplification of support vector solutions will possibly cause a degradation in generalization performance. In the formed methods there is no specific way to manage this possibility (DeCoste & Mazzoni, 2003); instead, the size of the reduced set is first specified, and the resulting accuracy loss is determined experimentally (Burges, 1996).

To control this circumstance, we can monitor the difference between the two solutions caused by the sim-

plification process, and the simplification process will stop when any replacement of two SVs by a new one makes the difference exceed a given threshold. In the following we define a quantity called *Maximum Marginal Difference* (*MMD*) to estimate the difference between two support vector solutions.

Definition 1 *Suppose that the distance from a point* $\Phi(x)$ *to the original optimal hyperplane is* d *(*d *is 1 when* x *is a non-bounded support vector), and to the new hyperplane determined by the simplified solution is* d'*. The Marginal Difference (MD) on* $\Phi(x)$ *regarding to the two solutions is*

$$MD(\Phi(x)) \stackrel{def}{=} |d - d'| \tag{33}$$

and the difference between two solutions is defined as

$$MMD \stackrel{def}{=} \max_{i=1...N_S} MD(\Phi(x_i)) \tag{34}$$

where $x_1, ..., x_{N_S}$ *are original support vectors.*

The *MMD* uses the differences between two distances from the image of the original support vectors to the two discriminant hyperplanes to estimate the difference between two support vector solutions. The reason for not using the difference between two normal vectors of the two hyperplanes $||\Psi - \Psi'||$ is that this quantity depends too much on $||\Psi||$ and $||\Psi'||$. For complicated problems ($||\Psi||$ is large), a small difference between two hyperplanes may cause a big difference $||\Psi - \Psi'||$, while for easy cases, a small $||\Psi - \Psi'||$ corresponds to a big difference between hyperplanes, so there is a big difference between the two solutions. One note on the implementation of calculating $MD(\Phi(x_i))$ is that whenever two support vectors (v_1, α_1) and (v_2, α_2) are replaced by a vector (v, α), the marginal difference on $\Phi(x_i)$ will change an amount of $\alpha_1 K(v_1, x_i) + \alpha_2 K(v_2, x_i) - \alpha K(v, x_i)$(ref. (7)); therefore, during the simplification process the marginal differences on the original support vectors can be calculated accumulatively using only three vectors.

3.3. Pursuing a Better Approximation

A better approximate solution can be achieved by applying the unconstrained optimization process to minimize $F = \|\Psi - \Psi'\|$ with respect to all z_j and β_j together (phase 2 in (Burges, 1996)). In spite of high cost (working in a space of $(d+1)N_Z$ variables), we also found that this process can bring an effective reduction in the objective function F, or effective improvement of the simplified solution.

4. Experiments

To assess its effectiveness on real world applications, we first used the proposed method to simplify ten binary classifiers trained to distinguish one digit from others in the US Postal Service (USPS) handwritten digit recognition database [1]. The dataset contains normalized grayscale images of handwritten digits taken from US zip codes; the size of each image is 16x16 pixels, and the data set is divided into a training set of 7,291 images and a test set of 2,007 images. For each binary classifier (using the one-versus-rest strategy) trained by a Gaussian RBF kernel or by a polynomial kernel, different values of *MMD* were used to give a different reduction rate in number of SVs as well as different levels of the loss in generalization performance. The results are reported in Table 1. The first column displays different values of threshold MMD ($MMD = 0.0$ for original machines). The second column displays the total number of SVs in all ten binary classifiers. There are two kinds of errors. The first, named "Phase 1 Err.", were produced by the simplified classifiers using the simplification process described in section 3.2 (phase 1), and the second, named "Phase 2 Err.", were produced by those using the optimization process described in 3.3 (phase 2) after phase 1 finished. For both kernels we could reduce more than 90% of SVs with only a minor loss in generalization performance.

Note that the achieved reduction rate depends on the "*complexity*" of the solution, or the difficulty of the problem. To judge this argument we conducted experiments on five other datasets: the MNIST database of handwritten digits [2], four datasets DNA, Letter Recognition, Satimage, and Shuttle in the StatLog collection [3]. These datasets are summarized in Table 2. Parameter settings for these datasets were polynomial kernel of degree five for the MNIST, Gaussian kernels with the width of $\frac{1}{0.6Variance}$ (Liu et al., 2003) for the StatLog datasets; the parameter MMD was fixed at 1.0. Experimental results in Table 2 show that with almost no change in generalization performance the achieved reduction rates could vary from 70.9% to 94.5% (corresponding to a speed-up rate from 3.4 to 18.2 times) depending on application.

5. Discussion

We have described a method to reduce the computational complexity of support vector machines by reduc-

[1] Available at http://www.kernel-machines.org/
[2] Available at http://yann.lecun.com/exdb/mnist/
[3] Available at http://www.liacc.up.pt/ML/

Table 1. Reduction in number of support vectors and the corresponding loss in generalization performance with different values of *MMD* on the USPS dataset.

	RBF machines: *gamma* = 0.0078, $C = 10$			Polynomial machines: *degree* = 3, $C = 10$		
MMD	# of SVs	Phase 1 Err.	Phase 2 Err.	# of SVs	Phase 1 Err.	Phase 2 Err.
0.0	5041	88(4.4%)	88(4.4%)	4538	88(4.4%)	88(4.4%)
0.1	3476	85(4.2%)	88(4.4%)	3024	88(4.4%)	88(4.4%)
0.2	2588	88(4.4%)	87(4.3%)	2269	91(4.5%)	88(4.4%)
0.5	1285	91(4.5%)	90(4.5%)	1114	93(4.6%)	89(4.4%)
0.7	864	97(4.8%)	94(4.7%)	795	104(5.2%)	89(4.4%)
1.0	502	108(5.4%)	95(4.7%)	522	110(5.5%)	91(4.5%)
1.2	343	124(6.2%)	97(4.8%)	397	116(5.8%)	93(4.6%)
1.5	246	144(7.2%)	101(5.0%)	270	147(7.3%)	95(4.7%)

Table 2. Experimental results on various applications.

		# of	Size		Original Machines		Simplified Machines	
Dataset	Dimensions	Classes	Train	Test	# of SVs	Error (%)	# of SVs	Error (%)
MNIST	784	10	60,000	10,000	22,294	1.5	2,538	1.6
Dna	180	3	2,000	1,186	1,686	6.0	93	6.4
Letter	16	26	15,000	5,000	10,284	5.0	2,993	5.2
Satimage	34	6	4,435	2,000	2,494	10.9	354	10.9
Shuttle	9	9	43,500	14,500	1,131	0.2	124	0.2

ing number of support vectors comprised in their solution. Our method has several advantages compared to earlier reduced set methods. Firstly, the reduced vectors are constructed in a more *"natural"* way, leading to a more "meaningful" reduced set. Each vector in the reduced set could be considered as representative of several closed original SVs belonging to the same class. In Figure 3 we display the whole reduced vector set of 10 simplified classifiers. Each reduced vector is constructed from the same positive or negative closed original SVs, so the original shape of these SVs is preserved. This is quite different from the former reduced set methods that construct each new vector from all original and newly constructed SVs. From Figure 3 we can also see that different machine requires a different number of reduced SVs. For example, the machine distinguishing character '1' from the other consists of only 6 SVs, while this number is 43 SVs for machine '8'. This indicates that it is unreasonable to decide the same number of reduced SVs for all machines. The second difference lies in the uniqueness of the result in finding the reduced set. With our proposed method, each reduced vector corresponds to the unique maximum point of a one-variable function on $(0, 1)$, and the result of the finding (for both two phases) is unique because we start from the same initial point and use the same search strategy. All the results described in this paper can be reproduced easily with a one-run test. Reproduction is difficult and very expensive, if not impossible, for the former methods because for each reduced vector they have to solve a multivariate parameter optimization problem, and the search has to restart many times with different initial points. For the second phase optimization, as noted in (Schoelkopf et al., 1999), the optimization also must be restarted to make sure that the global minimum of the cost function is actually found. The third advantage is its competitive SVs reduction rate while preserving well machine's performance. Experiments on the USPS dataset show that a reduction rate of 90.0% can be achieved with only a 0.3% loss in predictive accuracy (Gaussian kernel, $MMD = 1.0$), and 91.3% with a 0.2% lost (polynomial kernel, $MMD = 1.2$). The corresponding numbers reported in (Schoelkopf et al., 1999) are (for Gaussian kernel) 90% reduction rate with 0.3% loss (we report the reduction rate, not the number of SVs reduced because we did experiments on non-processed datasets and the total number of SVs were different).

The proposed method is applicable for common kernels like Gaussian RBFs and polynomial, and for both support vector classification and regression machines. For a further speed-up, other approximation methods, e.g., (DeCoste & Mazzoni, 2003), can be applied together with the reduced set methods to accelerate the test phase of support vector machines.

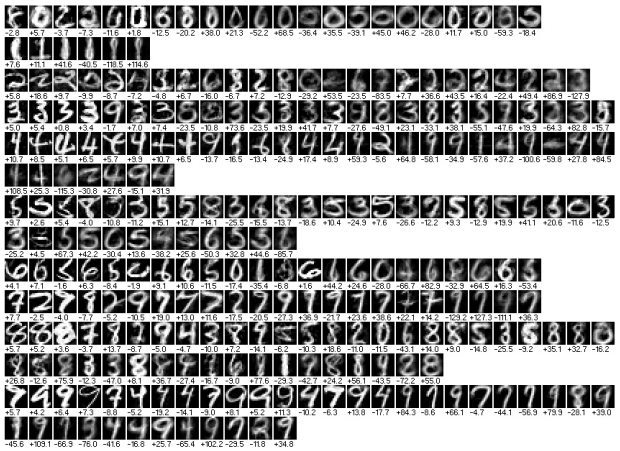

Figure 3. Display of vectors in the simplified solutions. The original 10 classifiers trained with polynomial kernel of degree three and the cost $C = 10$ consist of 4538 SVs and produce 88 errors (on 2007 testing data). The simplified 10 classifiers consist of 270 vectors and produce 95 errors.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments on the submission version of this paper.

References

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144–152). Pittsburgh, Pennsylvania, United States.

Burges, C. J. C. (1996). Simplified support vector decision rules. *Proc. 13th International Conference on Machine Learning* (pp. 71–77). San Mateo, CA.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Burges, C. J. C., & Schoelkopf, B. (1997). Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan and T. Petsche (Eds.), *Advances in neural information processing systems 9*, 375–381. Cambridge, MA: MIT Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Cristianini, C., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.

DeCoste, D., & Mazzoni, D. (2003). Fast query-optimized kernel machine classification via incremental approximate nearest support vectors. *Proceedings International Conference on Machine Learning (ICML-03)* (pp. 115–122).

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the European Conference on Machine Learning* (pp. 137–142). Berlin: Springer.

Kwok, J., & Tsang, I. (2003). The pre-image problem in kernel methods. *In Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)* (pp. 408–415). Washington, D.C., USA.

LeCun, Y., Botou, L., Jackel, L., Drucker, H., Cortes, C., Denker, J., Guyon, I., Muller, U., Sackinger, E., Simard, P., & Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks*, 261–276.

Liu, C., Nakashima, K., Sako, H., & Fujisawa, H. (2003). Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, *36*, 2271–2285.

Mika, S., Schoelkopf, B., Smola, A., Muller, K.-R., Scholz, M., & Ratsch, G. (1999). Kernel pca and de-noising in feature spaces. In M. S. Kearns, S. A. Solla and D. A. Cohn (Eds.), *Advances in neural information processing systems 11*, 536–542. Cambridge, MA: MIT Press.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines:an application to face detection. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 130–136). Puerto Rico.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2002). *Numerical recipes in c++ : the art of scientific computing*. Cambridge University Press.

Schoelkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Muller, K., Ratsch, G., & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Trans. Neural Networks*, *10*, 1000–1017.

Schoelkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Vapnik, V. (1995). *The nature of statistical learning theory*. N.Y.: Springer.