A Bottom-Up Method for Simplifying Support Vector Solutions

DucDung Nguyen and TuBao Ho

***Abstract*—The high generalization ability of support vector machines (SVMs) has been shown in many practical applications, however, they are considerably slower in test phase than other learning approaches due to the possibly big number of support vectors comprised in their solution. In this letter, we describe a method to reduce such number of support vectors. The reduction process iteratively selects two nearest support vectors belonging to the same class and replaces them by a newly constructed one. Through the analysis of relation between vectors in input and feature spaces, we present the construction of the new vectors that requires to find the unique maximum point of a one-variable function on (0,1), not to minimize a function of many variables with local minima in previous reduced set methods. Experimental results on real life dataset show that the proposed method is effective in reducing number of support vectors and preserving machine's generalization performance.**

***Index Terms*—Feature space, input space, kernel methods, reduced set method, support vector machines (SVMs).**

I. Introduction

Support vector machines (SVMs) (e.g., [12], [4]) have been found to be very robust in many applications, for example in the field of optical character recognition [7], [8], text categorization [6], and face detection in images [9]. The high generalization ability of SVMs is ensured by special properties of the optimal hyperplane that maximizes the distance to training examples in a high dimensional feature space [1]. However, SVMs are considerably slower in test phase than other learning methods like decision trees or neural networks [2], [3], [7], [8].

The solution of a SVM is parameterized by a set of training examples, called support vectors (SVs), and their corresponding weights. When a new test sample 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., [2], [5], and [11], try to approximate the original solution by another comprised by a much smaller number of newly constructed vectors, called the reduced vectors set. For former methods described in [2], [10], and [11] the construction of each reduced vector requires to solve an unconstrained optimization problem in a space of $d+1$ variables, where d is dimension of input space. Hence the computation is very expensive because the search must be repeated many times with different initial points to escape from local minima [2], [10]. The method described in [5] tries to find exact approximations of SVM solutions based on the linearly dependency of SVs in feature space, but its applicability is very limited due to the fact that feature space's dimensionality is often very large or even infinite.

In this letter, we describe a conceptually simpler and computationally less expensive method to simplify support vector solutions. Instead of constructing reduced vectors set incrementally, two nearest SVs belonging to the same class will be iteratively considered and replaced by a newly constructed vector. The construction of new vectors only requires to find the unique maximum point of a one-variable function on (0, 1). Experimental results show the effectiveness of our proposed method in reducing the number of support vectors and preserving generalization performance. On the United States Postal Service (USPS) handwritten digit recognition database, a 91.3% (for polynomial kernel) and 90.0% (for Gaussian kernel) reduction rate were achieved, with a corresponding 0.2% and 0.3% loss in predictive accuracy.

Manuscript received July 28, 2004; revised July 31, 2005.

The authors are with the Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan (e-mail: dungduc@jaist.ac.jp; bao@jaist.ac.jp).

Digital Object Identifier 10.1109/TNN.2006.873287

II. Reduced SVMs

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 a d-dimensional input space to a possibly high-dimensional feature space [12]. For a two-class classification problem, the decision rule takes the form

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

where α_i are weights of support vectors x_i, x is the input vector needed to classify, b is the bias, and N_S is the number of support vectors.

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

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

The goal of reduced set method is to choose the smallest $N_Z < N_S$, and construct the corresponding reduced set $\{(z_i, \beta_i)\}_{i=1\ldots N_Z}$ such that any resulting loss in generation performance remains acceptable [2].

The method described in [2] 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 $\Psi_m (\Psi_0 = \Psi)$

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

In general, an unconstrained optimization technique is used to find (z_m, β_m). For a particular kind of kernel $K(x,y) = K(\|x-y\|^2)$ a fixed-point iteration scheme can be used to improve the speed of the finding [10], [11]. However, the main drawback is still that they may suffer from numerical instability and get trapped in a local minimum.

III. A Bottom-Up Method for Simplifying SV Solutions

A. Simplification of Two SVs

The solution of SVMs can be analyzed from a mechanical point of view: if each image of support vectors exerts a force on the decision

hyperplane, then the SVMs solution satisfies the conditions of equilibrium [3]. 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 an 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

$$y = \operatorname{sign}\left(\sum_{k=1,k\neq i,k\neq j}^{N_S} \alpha_k K(x,x_k) + \alpha_m M\cdot\Phi(x) + b\right). \quad (6)$$

The difficulty is that M cannot be used directly; we must use its pre-image, and in many situations we cannot find it. Our solution is to approximate M by an 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

$$\min_z \|M-\Phi(z)\|^2. \quad (7)$$

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

$$z = kx_i + (1-k)x_j \quad (8)$$

where k is the maximum point of

$$f(k) = mC_{ij}^{(1-k)^2} + (1-m)C_{ij}^{k^2} \quad (9)$$

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

Proposition 2: For polynomial kernels $K(x,y) = (x\cdot y)^p$, the 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

$$z = \frac{\|M\|^{1/p}}{\|z^*\|}z^* \quad (10)$$

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

$$h(k) = \|M\|u(k)v(k) \quad (11)$$

where

$$u(k) = \frac{1}{\left[x_i^2k^2 + 2(x_i\cdot x_j)k(1-k) + x_j^2(1-k)^2\right]^{p/2}} \quad (12)$$

$$v(k) = m\left[x_i^2k + (x_i\cdot x_j)(1-k)\right]^p + (1-m)\left[(x_i\cdot x_j)k + x_j^2(1-k)\right]^p. \quad (13)$$

The important point here is that $f(k)$ and $h(k)$ are one-variable functions having unique maximum point on (0, 1). The maximum point can be easily reached using common univariate optimization methods.

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

$$\beta = \frac{\alpha_m M\cdot\Phi(z)}{\|\Phi(z)\|^2}. \quad (14)$$

Equation (14) 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 ([10]): The optimal coefficients $\beta = (\beta_1,\ldots,\beta_{N_Z})$ 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 linear independent $\Phi(z_1),\ldots,\Phi(z_{N_Z})$) are given by

$$\beta = (\mathbf{K}^z)^{-1}\mathbf{K}^{zx}\alpha \quad (15)$$

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 [10], (15) always gives optimal coefficients to get a solution that is at least as good as the original one. In our experiments, (15) was used to recompute the final coefficients of all vectors in the reduced set after the iterative simplification process finished.

B. Simplification of SV 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 III-A. 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.

1) Selection Heuristic: The optimal pair of two SVs is the one that produces a minimum value of $d(\beta)$ in (32). However, we cannot use this criterion because it is too expensive. Moreover, we are more concerned about the original solution and the final simplified 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 (7). For Gaussian kernels, we can select x_i and x_j that give a maximum value of $C_{ij} = K(x_i,x_j)$ in (9), or equivalently, select two closest support vectors belonging to the same positive or negative class. Another interpretation for this selection heuristic is that we are trying to approximate two Gaussian RBFs by one Gaussian RBF, and intuitively, the closer pair centers, the better approximation. This selection heuristic can also be reasonably applied to polynomial kernels because the input vector z that maximizes $M\cdot\Phi(z)$ in (26) is linear 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 $M\cdot\Phi(z)$.

2) Stopping Condition: The simplified solution is always different from the original one, so the simplification will possibly cause a degradation in generalization performance. To control this circumstance, we can monitor the difference between the two solutions caused by simplification process, and the 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, 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

$$\mathrm{MD}(\Phi(x)) \stackrel{\mathrm{def}}{=} |d-d'| \quad (16)$$

and the difference between two solutions is defined as

$$\mathrm{MMD} \stackrel{\mathrm{def}}{=} \max_{i=1\ldots N_S}\mathrm{MD}(\Phi(x_i)) \quad (17)$$

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

The MMD uses the differences between two distances from the image of original support vectors to the two discriminant hyperplanes

TABLE I
BOTTOM-UP SIMPLIFICATION ALGORITHM

Input: a set of N_S support vectors $x_1, ..., x_{N_S}$
a threshold θ of *MMD*
Output: a set of N_Z reduced support vectors, $N_Z < N_S$
1. $PairList = \left\{(x_i, x_j) | i = 1...N_S, x_j = \arg\min_k(\|x_i - x_k\|^2), 1 \le k \le N_S, \alpha_i \alpha_k > 0\right\}$
2. Sort $PairList$ incrementally according to the distance between two vectors in pair
3. For each different pair (x_i, x_j) in $PairList$
4. Try to replace x_i and x_j by z found by (8) or (10), weight z by (14)
5. If the replacement does not make MMD greater than θ
6. Then replace x_i and x_j by z, set $N_Z = N_Z - 1$, update $PairList$, and restart the loop from trying with the first pair in $PairList$
7. Otherwise try the next pair, or exist simplification process if (x_i, x_j) is the last pair in $PairList$
8. Recompute coefficients of all support vectors using (15)
9. Optimize the whole reduced set using phase 2 described in Section III-C
10. Return the reduced set

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.

3) The Algorithm: The algorithm for simplifying support vector solution is described in Table I. It iteratively selects two support vectors and tries to replace them by a newly created vector. The process will stop when there is no replacement success, and finally all coefficients and reduced vectors are recomputed to get a better approximation.

C. 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 [2]). Though the cost is high (working in a space of $(d+1)N_Z$ variables), this process can bring an effective reduction in the objective function F, or effective improvement of the simplified solution.

IV. EXPERIMENT

To assess its effectiveness, we applied proposed method to simplify ten binary classifiers trained to distinguish one digit from others in the USPS handwritten digit recognition database. The dataset contains normalized gray scale images of handwritten digits taken from US zip codes; the size of each image is 16×16 pixels, and the data set is divided into a training set of 7291 images and a test set of 2007 images. For each binary classifier trained by a Gaussian 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 loss in generalization performance. The first column in Table II displays different values of threshold MMD ($\mathrm{MMD} = 0.0$ for original machines). The columns titled "# of SVs" display the total number of SVs in all ten binary classifiers. There are two kinds of errors. The first, named "Phase 1 Errors," were produced by the simplified classifiers using the simplification process described in Section III-B (phase 1), and the second, named "Phase 2 Errors," were produced by those using the optimization process described in Section III-C (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.

TABLE II
REDUCTION IN NUMBER OF SVS AND THE CORRESPONDING LOSS IN GENERALIZATION PERFORMANCE WITH DIFFERENT VALUES OF MMD

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

V. DISCUSSION

We have described a method to reduce the computational complexity of support vector machines by reducing number of support vectors comprised in their solution. Our method has several advantages compared to earlier reduced set methods. First, 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. The second advantage lies in the uniqueness of the result in finding 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 search from the same initial point and use the same strategy. All the results described in this letter 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 [10], 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, $\mathrm{MMD} = 1.0$), and 91.3% with a 0.2% lost (polynomial kernel, $\mathrm{MMD} = 1.2$). The corresponding numbers reported in [10] are (for Gaussian kernel) 90% reduction rate with 0.3% loss.

Appendix A
Proof of Proposition 1

Proof: For Gaussian RBF kernels, Φ maps each input vector onto the surface of the unit hypersphere in feature space, so we have $\|\Phi(z)\| = 1$ for every z, $\|M\|$ is a constant and can be calculated via $\Phi(x_i)$ and $\Phi(x_j)$. (7) is equivalent to

$$\max_z M \cdot \Phi(z). \tag{18}$$

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

$$\begin{aligned} 0 &= \nabla_z(M \cdot \Phi(z)) \\ &= 2m\exp(-\gamma\|x_i - z\|^2)(x_i - z) \\ &\quad + 2(1-m)\exp(-\gamma\|x_j - z\|^2)(x_j - z) \end{aligned} \tag{19}$$

leading to

$$z = \frac{\sum_{s=i,j}\alpha_s \exp(-\gamma\|x_s - z\|^2)x_s}{\sum_{s=i,j}\alpha_s \exp(-\gamma\|x_s - z\|^2)} \tag{20}$$

or

$$z = kx_i + (1-k)x_j \tag{21}$$

where

$$k = \frac{\alpha_i \exp(-\gamma\|x_i - z\|^2)}{\sum_{s=i,j}\alpha_s \exp(-\gamma\|x_s - z\|^2).} \tag{22}$$

Because $\alpha_i\alpha_j > 0$ (or x_i and x_j belong to the same positive or negative class) then $0 < k < 1$. (21) 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 $f(k)$

$$\begin{aligned} f(k) &= M \cdot \Phi(kx_i + (1-k)x_j) \\ &= m\exp(-\gamma\|x_i - x_j\|^2(1-k)^2) \\ &\quad + (1-m)\exp(-\gamma\|x_i - x_j\|^2k^2) \\ &= mC_{ij}^{(1-k)^2} + (1-m)C_{ij}^{k^2} \end{aligned} \tag{23}$$

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

Appendix B
Proof of Proposition 2

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. 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 (7) becomes

$$\max_z M \cdot \Phi(z) \tag{24}$$

subject to

$$\|z\| = \|M\|^{1/p}. \tag{25}$$

The following lemma shows that the (vector) solution of (24), x_i, and x_j are linearly dependent.

Lemma 1: The input vector z that maximizes $M \cdot \Phi(z)$ in (24) is linearly dependent with x_i and x_j.

Proof: Replacing $M = m\Phi(x_i) + (1-m)\Phi(x_j)$ into (24) we have

$$\begin{aligned} M \cdot \Phi(z) &= (m\Phi(x_i) + (1-m)\Phi(x_j)) \cdot \Phi(z) \\ &= m(x_i \cdot z)^p + (1-m)(x_j \cdot z)^p. \end{aligned} \tag{26}$$

Suppose that z is an input vector satisfying constraint (25) and z_1 is the orthogonal projection of z on the plane determined by x_i and x_j. Let's consider input vector z'

$$z' = \frac{\|z\|}{\|z_1\|}z_1. \tag{27}$$

We have z' satisfying constraint (25) 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 linear dependent with x_i and x_j. □

Because the solution of (24), called z_{opt}, lies on the plane (x_i, x_j) and $\|z_{\mathrm{opt}}\| = \|M\|^{1/p}$, there exits a vector z^* and a scalar k such that

$$z^* = kx_i + (1-k)x_j \tag{28}$$

and

$$z_{\mathrm{opt}} = \frac{\|M\|^{1/p}}{\|z^*\|}z^*. \tag{29}$$

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

$$\begin{aligned} g(z_{\mathrm{opt}}) &= M \cdot \Phi(z_{\mathrm{opt}}) \\ &= m(x_i \cdot z_{\mathrm{opt}})^p + (1-m)(x_j \cdot z_{\mathrm{opt}})^p \\ &= \frac{\|z_{\mathrm{opt}}\|^p}{\|z^*\|^p}\left[m(x_i \cdot z^*)^p + (1-m)(x_j \cdot z^*)^p\right]. \end{aligned} \tag{30}$$

Because z_{opt} satisfies (25) then $\|z_{\mathrm{opt}}\|^p = \|M\|$. Replacing $z^* = kx_i + (1-k)x_j$ into (30) leads to

$$h(k) = \|M\|u(k)v(k) \tag{31}$$

where $u(k)$ and $v(k)$ are defined in (12) and (13). ■

Appendix C
Proof of Proposition 3

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

$$\begin{aligned} d(\beta) &= |\alpha_m M \cdot \Phi(x) - \beta\Phi(z) \cdot \Phi(x)| \\ &= |(\alpha_m M - \beta\Phi(z)) \cdot \Phi(x)|. \end{aligned} \tag{32}$$

This difference will be minimized when $d(\beta)$ gets the minimum value. In (32) $d(\beta)$ can be minimized by minimizing $d_1(\beta) = \|\alpha_m M - \beta\Phi(z)\|$, and its minimum point is at

$$\beta = \frac{\alpha_m M \cdot \Phi(z)}{\|\Phi(z)\|^2} \tag{33}$$

■

REFERENCES

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Ann. Workshop on Computat. Learn. Theory*, 1992, pp. 144–152.

[2] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. 13th Int. Conf. Mach. Learn.*, San Mateo, CA, 1996, pp. 71–77.

[3] ——, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Disc.*, vol. 2, no. 2, pp. 121–167, 1998.

[4] C. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[5] T. Downs, K. Gates, and A. Masters1, "Exact simplification of support vector solutions," *J. Mach. Learn. Res.*, vol. 2, pp. 293–297, Dec. 2001.

[6] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. Europ. Conf. Mach. Learn.*, C. Nedellec and C. Rouveirol, Eds. Berlin, Germany, 1998, pp. 137–142.

[7] Y. LeCun, L. Botou, L. Jackel, H. Drucker, C. Cortes, J. Denker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Netw.*, pp. 261–276, 1995.

[8] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: Bench-marking of state-of-the-art techniques," *Pattern Recognit.*, vol. 36, pp. 2271–2285, 2003.

[9] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," *IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 130–136, Jan. 1997.

[10] B. Schoelkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. Muller, G. Ratsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1000–1017, 1999.

[11] B. Schoelkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[12] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer, 1995.