
An Introduction to Support Vector Machines

Yuh-Jye Lee, Yi-Ren Yeh, and Hsing-Kuo Pao

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology
Taipei, Taiwan 10607
{yuh-jye,D9515009, pao}@mail.ntust.edu.tw

This chapter aims to provide a comprehensive introduction to Support Vector Machines (SVMs). The SVM algorithm was proposed based on the advances of the statistical learning theory and has drawn a lot of research interests in many areas. It has become the state of the art in solving classification and regression problems. It is not only because of its sound theoretical foundation but also because of its good generalization performance in many real applications. We will address the theoretical, algorithmic and computational issues. We also will discuss the implementation problems in real applications such as how to deal with the unbalanced dataset, how to tune the parameters to have a better performance and how to deal with large scale dataset, etc.

1 Introduction

In the last decade, significant advances have been made in support vector machines (SVMs) both theoretically using statistical learning theory, as well as algorithmically based principally on optimization techniques [3, 9, 20, 22, 27, 29]. SVMs have been successfully developed and have become powerful tools for solving data mining problems such as classification, regression and feature selection. In classification problems, SVMs determine an optimal separating hyperplane that classifies data points into different categories. Here, “optimal” is used in the sense that the *separating hyperplane* has the best generalization ability for the unseen data points based on statistical learning theory. This optimal separating hyperplane is generated by solving an underlying optimization problem. SVMs can discriminate between complex data patterns by generating a highly nonlinear separating hyperplane, that is implicitly defined by a nonlinear kernel map. This ability makes SVMs applicable to many important real world problems such as bankruptcy prognosis, face detection, analysis of DNA microarrays and breast cancer diagnosis and prognosis[4, 23].

The goal of this chapter is to provide a comprehensive introduction to SVMs. The materials include the basic idea of SVMs, the formulations of

SVMs, the nonlinear extension of SVMs, the variants of SVMs, the implementation of SVMs, and the some practical issues in SVMs.

2 Support Vector Machine Formulation

In this section, we first introduce the basic idea of SVMs and give the formulation of the linear support vector machine. However, many datasets generated from the real world problems cannot be well separated by a linear separating hyperplane. The boundary between categories is nonlinear. The nature way for dealing this situation might be mapping the dataset into a higher dimensional feature space. Hopefully, the images of these data points will become linear separable in the feature space. We then apply linear SVM in the feature space. The nonlinear extension of SVMs can implement the process without explicitly defined the nonlinear map. It can be achieved by using the “kernel trick”. In this material, we mainly confine ourselves to binary classification, that is classifying points into two classes, \mathbf{A}_+ and \mathbf{A}_- . For the multi-class case, many strategies have been proposed. They either decompose the problem into a series of binary classification or formulate it as a single optimization problem. We will discuss this issue in Section 5. For the binary classification, we are given a dataset consisting of m points in the n -dimensional real space \mathbb{R}^n . Each point in the dataset comes with a class label y , $+1$ or -1 , indicating one of two classes, \mathbf{A}_+ and \mathbf{A}_- , to which the point belongs. We represent these data points by an $m \times n$ matrix \mathbf{A} , where the i^{th} row of the matrix \mathbf{A} , \mathbf{x}_i , corresponds to the i^{th} data point.

2.1 Conventional Support Vector Machine

Structural Risk Minimization

The main goal of the classification problem is to find a classifier that can predict the label of new unseen data points correctly. This can be achieved by learning from the given labeled data. Looking for a model that fit the given data usually is not a good way to do. There always exists a model that can discriminate two classes data points perfectly as long as there is no identical data points that have different labels. There are some bounds governing the relation between the capacity of a learning machine and its performance. It can be used for balancing the *model bias* and *model variance*. The theory grew out of considerations of under what circumstances, and how quickly, the mean of some empirical quantity converges uniformly, as the number of data points increases, to the true mean. The expectation of the test error for a learning model is as follows:

$$R(\alpha) = \int \frac{1}{2} |y - f(\mathbf{x}, \alpha)| dP(\mathbf{x}, y), \quad (1)$$

where \mathbf{x} is an instance and y is the class label of \mathbf{x} from some unknown probability distribution $P(\mathbf{x}, y)$, and $f(\mathbf{x}, \alpha)$ is the learning model with the adjustable parameter α and output values 1 and -1. The error in (1) can represent the true mean error but it needs to know what $P(\mathbf{x}, y)$ is. The quantity $R(\alpha)$, called actual risk, is the quantity that we are interested in. However, estimating $P(\mathbf{x}, y)$ is usually not possible so that (1) is not very useful in practical using. In [32], the authors proposed an upper bound for $R(\alpha)$ with probability $1 - \eta$ as follows:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\lg(2m/h) + 1) - \lg(\eta/4)}{m}}, \quad (2)$$

where η is between 0 and 1, m is the number of instances, h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension, and $R_{emp}(\alpha)$ is the empirical risk denoted as follows:

$$R_{emp}(\alpha) = \frac{1}{2m} \sum_{i=1}^m |y_i - f(\mathbf{x}_i, \alpha)|. \quad (3)$$

The second term on the right hand side in (2) is called the VC confidence. This upper bound gives a principled method for choosing a learning model for a given task. Thus given several different learning models and choosing a fixed, sufficiently small η , by then taking that model which minimizes the right hand side, we are choosing that model which gives the lowest upper bound on the actual risk. Note that the VC confidence is a monotonic increasing function of h . This means that a complicated learning model might also have a high upper bound on the actual risk. In general, for non zero empirical risk, one wants to choose that learning model which minimizes the right hand side of (2). The principle of structural risk minimization (SRM) in [32] is finding the subset of functions which minimizes the bound on the actual risk. This can be achieved by simply training a series of models, one for each subset, where for a given subset the goal of training is simply to minimize the empirical risk. One then can take that learning model whose empirical risk and VC confidence is minimal.

The Formulation of Conventional Support Vector Machine

Let us start with a strictly linearly separable case, *i.e.* there exists a hyper-plane which can separate the data \mathbf{A}_+ and \mathbf{A}_- . In this case we can separate the two classes by a pair of parallel *bounding planes*:

$$\begin{aligned} \mathbf{w}^\top \mathbf{x} + b &= +1 \\ \mathbf{w}^\top \mathbf{x} + b &= -1, \end{aligned} \quad (4)$$

where \mathbf{w} is the normal vector to these planes and b determines their location relative to the origin. The first plane of (4) bounds the class \mathbf{A}_+ and the second plane bounds the class \mathbf{A}_- . That is,

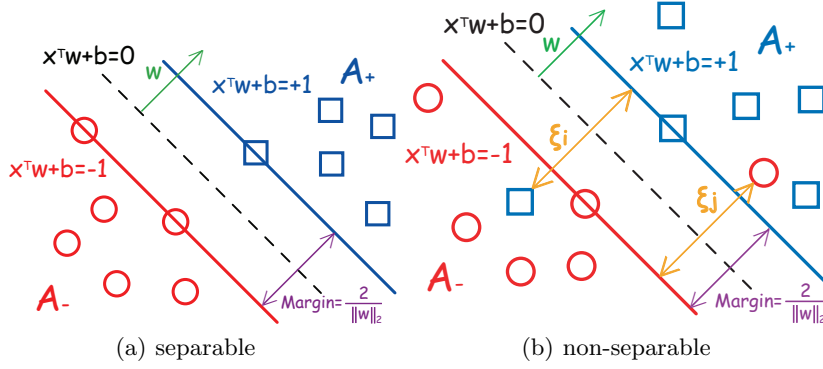


Fig. 1. The illustration of linearly separable and non-separable SVM

$$\begin{aligned} \mathbf{w}^\top \mathbf{x} + b &\geq +1, \text{ for } \mathbf{x} \in \mathbf{A}_+, \\ \mathbf{w}^\top \mathbf{x} + b &\leq -1, \text{ for } \mathbf{x} \in \mathbf{A}_-. \end{aligned} \quad (5)$$

According to the statistical learning theory [32], SVM achieves a better prediction ability via maximizing the margin between two bounding planes. Hence, SVM searches for a separating hyperplane by maximizing $\frac{2}{\|\mathbf{w}\|_2}$. It can be done by means of minimizing $\frac{1}{2} \|\mathbf{w}\|_2^2$ and leads to a quadratic program, as follows:

$$\begin{aligned} \min_{(\mathbf{w}, b) \in \mathbb{R}^{n+1}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \text{ for } i = 1, 2, \dots, m. \end{aligned} \quad (6)$$

The linear separating hyperplane is the plane

$$\mathbf{w}^\top \mathbf{x} + b = 0, \quad (7)$$

midway between the bounding planes (4) as shown in Figure 1(a). For the linearly separable case, the feasible region of the above minimization problem (6) is nonempty and the objective function is quadratic convex function, hence there exists an optimal solution (\mathbf{w}^*, b^*) . The data points on the bounding planes, $\mathbf{w}^{*\top} \mathbf{x} + b^* = \pm 1$, are called support vectors. If we remove any point which is not a support vector, the training result will not be changed. This is a very nice character of SVMs learning algorithms. Once we have the training result, all we need to keep in our databases are the support vectors.

If the classes are linearly inseparable then the two planes bound the two classes with a “soft margin” determined by a nonnegative slack vector variable ξ , that is:

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_i + b + \xi_i &\geq +1, \text{ for } \mathbf{x}_i^\top \in \mathbf{A}_+ \\ \mathbf{w}^\top \mathbf{x}_i + b - \xi_i &\leq -1, \text{ for } \mathbf{x}_i^\top \in \mathbf{A}_-. \end{aligned} \quad (8)$$

The 1-norm of the slack variable ξ , $\sum_{i=1}^m \xi_i$, is called the penalty term. We are going to determine a separating hyperplane that not only correctly classifies the training data, but also performs well on a testing set. This idea is equivalent to minimizing the upper bound of the actual risk in (2). We depict this geometric property in Figure 1(b). Hence, we can extend equation (6) and produce the conventional SVM [32] as following formulation:

$$\begin{aligned} \min_{(\mathbf{w}, b, \xi) \in \mathbb{R}^{n+1+m}} \quad & C \sum_{i=1}^m \xi_i + \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \dots, m. \end{aligned} \quad (9)$$

Here $C > 0$ is a positive parameter which balances the weights of the penalty term $\sum_{i=1}^m \xi_i$ versus the margin maximization term $\frac{1}{2} \|\mathbf{w}\|_2^2$. The objective function of (9) can be interpreted as the Structural Risk Minimization (SRM) inductive principle [3, 32]. Basically, SRM defines a trade-off between the quality of the separating hyperplane on the training data and the complexity of the separating hyperplane. Higher complexity of the separating hyperplane may cause overfitting leading to poor generalization. The positive parameter C which can be determined by a *tuning procedure* (where a surrogate testing set is extracted from the training set), plays the role of balancing this trade-off. We will discuss this further in a later section.

2.2 Nonlinear Extension of SVMs via Kernel Trick

Many datasets cannot be well separated by a linear separating hyperplane, but could be linearly separated if mapped into a higher or much higher dimensional space by using a nonlinear map. For example, consider the classical Exclusive-Or (XOR) problem, $\mathbf{A}_+ = \{(1, 1), (-1, -1)\}$ and $\mathbf{A}_- = \{(1, -1), (-1, 1)\}$. A nice feature of SVM is that we do not need even to know the nonlinear map explicitly but can still apply a linear algorithm to the classification problem in the higher dimensional space. In order to do so we need to investigate the dual problem of (9) and the “kernel trick”.

Dual Form of SVMs

The conventional support vector machine formulation (9) is a standard convex quadratic program [2, 21, 25]. The Wolfe dual problem of (9) is as follows:

$$\begin{aligned} \max_{\mathbf{u} \in \mathbb{R}^m} \quad & \sum_{i=1}^m u_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j u_i u_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{s.t.} \quad & \sum_{i=1}^m y_i u_i = 0 \\ & 0 \leq u_i \leq C \quad \text{for } i = 1, 2, \dots, m, \end{aligned} \quad (10)$$

where $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the inner product of \mathbf{x}_i and \mathbf{x}_j . The primal variable \mathbf{w} is given by:

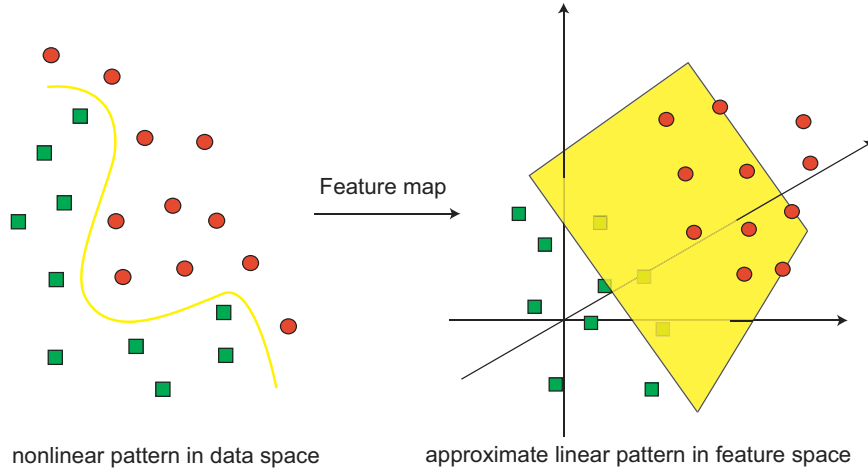


Fig. 2. The illustration of nonlinear SVM

$$\mathbf{w} = \sum_{\{i|u_i>0\}}^m y_i u_i \mathbf{x}_i . \quad (11)$$

The dual variable u_i corresponds to a training point \mathbf{x}_i . The normal vector \mathbf{w} can be expressed in terms of a subset of training data points (called support vectors) which have a corresponding dual variable u_i that is positive. By the Karush-Kuhn-Tucker complementarity conditions [2, 21]:

$$\begin{aligned} 0 \leq u_i \perp y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i - 1 &\geq 0 \\ 0 \leq C - u_i \perp \xi_i \geq 0 \end{aligned} , \text{ for } i = 1, 2, \dots, m, \quad (12)$$

we can determine b simply by taking any training point, x_i such that $i \in I := \{k | 0 < u_k < C\}$ and obtain:

$$b = y_i - \mathbf{w}^\top \mathbf{x}_i = y_i - \sum_{j=1}^m (y_j u_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle) . \quad (13)$$

Kernel Trick

From the dual SVM formulation (10), we know that all we need to know about the training data is $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. That is just the *dot product* between training data vectors. This is a very crucial point. Let us map the training data points from the input space \mathbb{R}^n to a higher dimensional *feature* space \mathcal{F} by a nonlinear map Φ . The training data x in \mathcal{F} will become $\Phi(\mathbf{x}) \in \mathbb{R}^\ell$ where ℓ is the dimensionality of the feature space \mathcal{F} . By the above observation if we know the *dot product*, $\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$ for all $i, j = 1, 2, \dots, m$ then we can perform the linear SVM algorithm in the feature space \mathcal{F} . The separating hyperplane will be linear in the feature space \mathcal{F} but nonlinear in the input space \mathbb{R}^n .

Note that we do not need to know the nonlinear map Φ explicitly. It can be achieved by employing a kernel function. If we let $K(\mathbf{x}, \mathbf{z}) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ be an inner product kernel function satisfying *Mercer's condition* [3, 7, 8, 9, 32], positive semi-definiteness condition (see Definition 2.1) then we can construct a nonlinear map Φ such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$, $i, j = 1, 2, \dots, m$. Hence, we can use a linear SVM on $\Phi(\mathbf{x})$ in the feature space \mathcal{F} by replacing the $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in the objective function of (10) with a nonlinear kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. The resulting dual nonlinear SVM formulation becomes:

$$\begin{aligned} \max_{\mathbf{u} \in \mathbb{R}^m} \quad & \sum_{i=1}^m u_i - \frac{1}{2} \sum_{i,j=1}^m y_i y_j u_i u_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^m y_i u_i = 0 \\ & 0 \leq u_i \leq C \quad \text{for } i = 1, 2, \dots, m. \end{aligned} \quad (14)$$

The nonlinear separating hyperplane is defined by the solution of (14) as follows:

$$\sum_{j=1}^m (y_j u_j K(\mathbf{x}_j, \mathbf{x}_i)) + b = 0 \quad (15)$$

where

$$b = y_i - \sum_{j=1}^m (y_j u_j K(\mathbf{x}_j, \mathbf{x}_i)), \quad i \in I := \{k \mid 0 < u_k < C\}. \quad (16)$$

The “kernel trick” makes the use of the nice feature of SVMs to achieve the nonlinear extension without knowing the nonlinear mapping explicitly. This technique also make the nonlinear extension of SVMs algorithms more easily and a linear algorithm can be directly applied to make the nonlinear extension via replacing the kernel function $K(\mathbf{x}, \mathbf{z})$.

Mercer's Theorem

The basic idea of kernel trick is replacing the inner product between data points by the kernel function $K(\mathbf{x}, \mathbf{z})$. However, not all functions $K(\mathbf{x}, \mathbf{z})$ are allowable to reconstruct a corresponding nonlinear map. For which kernels are allowable or not can be answered by the *Mercer's condition* [32]. We conclude this section with *Mercer's condition* and two examples of a kernel function.

Definition 2.1 (*Mercer's condition*) Let $K(\mathbf{s}, \mathbf{t}) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous symmetric function and X be a compact subset of \mathbb{R}^n . If

$$\int_{X \times X} K(\mathbf{s}, \mathbf{t}) f(\mathbf{s}) f(\mathbf{t}) d\mathbf{s} d\mathbf{t} \geq 0, \quad \forall f \in L_2(X), \quad (17)$$

where the Hilbert space $L_2(X)$ is the set of functions f such that

$$\int_X f(\mathbf{t})^2 d\mathbf{t} < \infty. \quad (18)$$

then the function K satisfies *Mercer's condition*. This is equivalent to requiring that the kernel matrix $\mathbf{K}(\mathbf{A}, \mathbf{A})$ in our application is positive semi-definite [9], where $\mathbf{K}(\mathbf{A}, \mathbf{A})_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, 2, \dots, m$.

Example 2.1 *Polynomial Kernel*:

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + b)^d, \quad (19)$$

where d denotes the degree of the exponentiation.

Example 2.2 *Gaussian (Radial Basis) Kernel*:

$$K(\mathbf{x}, \mathbf{z}) = e^{-\gamma \|\mathbf{x} - \mathbf{z}\|_2^2}, \quad (20)$$

where γ is the width parameter of Gaussian kernel.

3 Variants of Support Vector Machines

SVMs can be formulated differently. The different formulations of SVMs have their properties in dealing with the data. They can be used in different goals and applications. In this section, we will introduce some variants of SVMs, as well as their prosperities and applications.

3.1 Smooth Support Vector Machine

In contrast to the conventional SVM of (9), smooth support vector machines minimize the square of the slack vector ξ with weight $\frac{C}{2}$. In addition, the SSVM-methodology appends $\frac{b^2}{2}$ to the term that is to be minimized. This expansion results in the following minimization problem:

$$\begin{aligned} \min_{(\mathbf{w}, b, \xi) \in \mathbb{R}^{n+1+m}} \quad & \frac{C}{2} \sum_{i=1}^m \xi_i^2 + \frac{1}{2} (\|\mathbf{w}\|_2^2 + b^2) \\ \text{s.t.} \quad & y_i (\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \dots, m. \end{aligned} \quad (21)$$

At a solution of (21), ξ is given by $\xi_i = \{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}_+$ for all i where the *plus* function x_+ is defined as $x_+ = \max\{0, x\}$. Thus, we can replace ξ_i in (21) by $\{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}_+$. This will convert the problem (21) into an unconstrained minimization problem as follows:

$$\min_{(\mathbf{w}, b) \in \mathbb{R}^{n+1}} \quad \frac{C}{2} \sum_{i=1}^m \{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}_+^2 + \frac{1}{2} (\|\mathbf{w}\|_2^2 + b^2). \quad (22)$$

This formulation reduces the number of variables from $n+1+m$ to $n+1$. However, the objective function to be minimized is not twice differentiable, which precludes the use of a fast Newton method. In the SSVM, the plus function x_+ is approximated by a smooth *p-function*, $p(x, \alpha) = x + \frac{1}{\alpha} \log(1 + e^{-\alpha x})$, $\alpha > 0$.

By replacing the plus function with a very accurate smooth approximation p -function gives the smooth support vector machine formulation:

$$\min_{(\mathbf{w}, b) \in \mathbb{R}^{n+1}} \frac{C}{2} \sum_{i=1}^m p(\{1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}, \alpha)^2 + \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2), \quad (23)$$

where $\alpha > 0$ is the smooth parameter. The objective function in problem (23) is strongly convex and infinitely differentiable. Hence, it has a unique solution and can be solved by using a fast Newton-Armijo algorithm. For the nonlinear case, this formulation can be extended to the nonlinear SVM by using the kernel trick as follows:

$$\min_{(\mathbf{u}, b) \in \mathbb{R}^{m+1}} \frac{C}{2} \sum_{i=1}^m p([1 - y_i \{ \sum_{j=1}^m u_j K(\mathbf{x}_i, \mathbf{x}_j) + b \}], \alpha)^2 + \frac{1}{2}(\|\mathbf{u}\|_2^2 + b^2), \quad (24)$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function. The nonlinear SSVM classifier $f(\mathbf{x})$ can be expressed as follows:

$$f(x) = \sum_{u_j \neq 0} u_j K(\mathbf{x}_j, \mathbf{x}) + b. \quad (25)$$

3.2 Least Square Support Vector Machine

3.3 Lagrangian Support Vector Machine

3.4 Reduced Support Vector Machine

In large scale problems, the full kernel matrix will be very large so it may not be appropriate to use the full kernel matrix when dealing with (24). In order to avoid facing such a big full kernel matrix, we brought in the reduced kernel technique [19]. The key idea of the reduced kernel technique is to randomly select a portion of data and to generate a thin rectangular kernel matrix, then to use this much smaller rectangular kernel matrix to replace the full kernel matrix. In the process of replacing the full kernel matrix by a reduced kernel, we use the Nyström approximation [28, 33] for the full kernel matrix:

$$\mathbf{K}(\mathbf{A}, \mathbf{A}) \approx \mathbf{K}(\mathbf{A}, \tilde{\mathbf{A}})\mathbf{K}(\tilde{\mathbf{A}}, \tilde{\mathbf{A}})^{-1}\mathbf{K}(\tilde{\mathbf{A}}, \mathbf{A}), \quad (26)$$

where $\tilde{\mathbf{A}}_{\tilde{m} \times n}$ is a subset of \mathbf{A} and $\mathbf{K}(\mathbf{A}, \tilde{\mathbf{A}}) = \tilde{\mathbf{K}}_{m \times \tilde{m}}$ is a reduced kernel. Thus, we have

$$\mathbf{K}(\mathbf{A}, \mathbf{A})\mathbf{u} \approx \mathbf{K}(\mathbf{A}, \tilde{\mathbf{A}})\mathbf{K}(\tilde{\mathbf{A}}, \tilde{\mathbf{A}})^{-1}\mathbf{K}(\tilde{\mathbf{A}}, \mathbf{A})\mathbf{u} = \mathbf{K}(\mathbf{A}, \tilde{\mathbf{A}})\tilde{\mathbf{u}}, \quad (27)$$

where $\tilde{\mathbf{u}} \in \mathbb{R}^{\tilde{m}}$ is an approximated solution of \mathbf{u} via the reduced kernel technique. The reduced kernel method constructs a compressed model and cuts down the computational cost from $\mathcal{O}(m^3)$ to $\mathcal{O}(\tilde{m}^3)$. It has been shown that the solution of reduced kernel matrix approximates the solution of full kernel matrix well.

3.5 1-norm Support Vector Machine

The 1-norm support vector machine replaces the regularization term $\|\mathbf{w}\|_2^2$ in (9) with the ℓ_1 -norm of \mathbf{w} . The ℓ_1 -norm regularization term is also called the LASSO penalty [31]. It tends to shrink the coefficients \mathbf{w} 's towards zeros in particular for those coefficients corresponding to redundant noise features [34]. This nice feature will lead to a way of selecting the important ratios in our prediction model. The formulation of 1-norm SVM is described as follows:

$$\begin{aligned} \min_{(\mathbf{w}, b, \xi) \in \mathbb{R}^{n+1+m}} \quad & C \sum_{i=1}^m \xi_i + \|\mathbf{w}\|_1 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \dots, m. \end{aligned} \quad (28)$$

The objective function of (28) is a piecewise linear convex function. We can reformulate it as the following linear programming problem:

$$\begin{aligned} \min_{(\mathbf{w}, \mathbf{s}, b, \xi) \in \mathbb{R}^{n+n+1+m}} \quad & C \sum_{i=1}^m \xi_i + \sum_{j=1}^n s_j \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \xi_i \geq 1 \\ & -s_j \leq w_j \leq s_j, \quad \text{for } j = 1, 2, \dots, n, \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \dots, m, \end{aligned} \quad (29)$$

where s_j is the upper bound of the absolute value of w_j . At the optimal solution of (29) the sum of s_j is equal to $\|\mathbf{w}\|_1$.

The 1-norm SVM can generate a very sparse solution \mathbf{w} and lead to a parsimonious model. In a linear SVM classifier, solution sparsity means that the separating function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ depends on very few input attributes. This characteristic can significantly suppress the number of the nonzero coefficients \mathbf{w} 's, especially when there are many redundant noise features [11, 34]. Therefore the 1-norm SVM can be a very promising tool for the variable selection tasks. We will use it to choose the important financial indices for our bankruptcy prognosis model.

3.6 The Smooth ϵ -Support Vector Regression

In regression problems, the response y belongs to the real number. We would like to find a linear or nonlinear regression function, $f(\mathbf{x})$, tolerating a small error in fitting this given dataset. This can be achieved by utilizing the ϵ -insensitive loss function that sets an ϵ -insensitive "tube" around the data, within which errors are discarded. Also, applying the idea of support vector machines (SVMs) [3, 32, 9], the function $f(\mathbf{x})$ is made as *flat* as possible in fitting the training dataset. We start with the linear case that is the regression function $f(\mathbf{x})$ is defined as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. This problem can be formulated as an unconstrained minimization problem given as follows:

$$\min_{(\mathbf{w}, b) \in \mathbb{R}^{n+1}} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^m |\xi_i|_\epsilon, \quad (30)$$

where $(|\xi_i|_\epsilon)_i = \max\{0, |\mathbf{w}^\top \mathbf{x}_i + b - y_i| - \epsilon\}$ that represents the fitting errors and the positive control parameter C here weights the tradeoff between the fitting errors and the flatness of the linear regression function $f(\mathbf{x})$. To deal with the ϵ -insensitive loss function in the objective function of the above minimization problem, conventionally, it is reformulated as a constrained minimization problem defined as follows:

$$\begin{aligned} \min_{(\mathbf{w}, b, \xi, \xi^*) \in \mathbb{R}^{n+1+2m}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^m (\xi_i + \xi_i^*) \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{x}_i + b - y_i \leq \epsilon + \xi_i \\ & -\mathbf{w}^\top \mathbf{x}_i - b + y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \text{ for } i = 1, 2, \dots, m. \end{aligned} \quad (31)$$

This formulation (31), which is equivalent to the formulation (30), is a convex quadratic minimization problem with $n + 1$ free variables, $2m$ nonnegative variables and $2m$ inequality constraints. However, introducing more variables and constraints in the formulation enlarges the problem size and could increase computational complexity for solving the regression problem.

In our smooth approach, we change the model slightly and solve it as an unconstrained minimization problem directly without adding any new variable and constraint. That is, the squares of 2-norm ϵ -insensitive loss, $\sum_{i=1}^m |\mathbf{w}^\top \mathbf{x}_i + b - y_i|_\epsilon^2$, is minimized with weight $\frac{C}{2}$ instead of the 1-norm of ϵ -insensitive loss as in (30). In addition, we add the term $\frac{1}{2}b^2$ in the objective function to induce strong convexity and to guarantee that the problem has a unique global optimal solution. These yield the following unconstrained minimization problem:

$$\min_{(\mathbf{w}, b) \in \mathbb{R}^{n+1}} \frac{1}{2} (\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2} \sum_{i=1}^m |\mathbf{w}^\top \mathbf{x}_i + b - y_i|_\epsilon^2. \quad (32)$$

This formulation has been proposed in active set support vector regression [24] and solved in its dual form. Inspired by smooth support vector machine for classification (SSVM) [20], the squares of ϵ -insensitive loss function in the above formulation can be accurately approximated by a smooth function which is infinitely differentiable and defined below. Thus, we are allowed to use a fast Newton-Armijo algorithm to solve the approximation problem. Before we derive the smooth approximation function, we show some interesting observations:

$$\begin{aligned} |x|_\epsilon &= \max\{0, |x| - \epsilon\} \\ &= \max\{0, x - \epsilon\} + \max\{0, -x - \epsilon\} \\ &= (x - \epsilon)_+ + (-x - \epsilon)_+. \end{aligned} \quad (33)$$

Furthermore, $(x - \epsilon)_+ \cdot (-x - \epsilon)_+ = 0$ for all $x \in \mathbb{R}$ and $\epsilon > 0$. Thus we have

$$|x|_\epsilon^2 = (x - \epsilon)_+^2 + (-x - \epsilon)_+^2. \quad (34)$$

In SSVM [20], the plus function x_+ is approximated by a smooth p -function, $p(x, \alpha) = x + \frac{1}{\alpha} \log(1 + e^{-\alpha x})$, $\alpha > 0$. It is straightforward to replace $|x|_\epsilon^2$ by a very accurate smooth approximation given by:

$$p_\epsilon^2(x, \alpha) = (p(x - \epsilon, \alpha))^2 + (p(-x - \epsilon, \alpha))^2. \quad (35)$$

We call this approximation p_ϵ^2 -function with smoothing parameter α . This p_ϵ^2 -function is used here to replace the squares of ϵ -insensitive loss function of (32) to obtain our smooth support vector regression (ϵ -SSVR):

$$\min_{(\mathbf{w}, b) \in \mathbb{R}^{n+1}} \frac{1}{2} (\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2} \sum_{i=1}^m p_\epsilon^2(\mathbf{w}^\top \mathbf{x}_i + b - y_i, \alpha), \quad (36)$$

where $p_\epsilon^2(\mathbf{w}^\top \mathbf{x}_i + b - y_i, \alpha) \in \mathbb{R}$. This problem is a strongly convex minimization problem without any constraint. It is easy to show that it has a unique solution. Moreover, the objective function in (36) is infinitely differentiable, thus we can use a fast Newton-Armijo method (only requiring twice differentiability) to solve the problem.

4 Implementation of SVMs

The support vector machine, in either its primal formulation (9) or dual formulation (10), is simply a standard convex quadratic program (for the nonlinear SVM, the kernel function $K(\mathbf{x}, \mathbf{x})$ used in (14) has to satisfy *Mercer's condition* in order to keep the *convexity* of the objective function). The most straightforward way for solving it is to employ a standard quadratic programming solver such as CPLEX [14] or using an interior point method for quadratic programming [10]. Because of the simple structure of the dual formulation of linear (10) and nonlinear (14) SVM, many SVM algorithms are applied in the dual space and convert the optimal dual solution to the optimal primal solution, w and b , using the relations (11) and (13). This works well for small or moderately sized dataset problems. However, for a massive dataset one would like to avoid dealing with a huge dense kernel matrix $\mathbf{A}\mathbf{A}^\top$ or $\mathbf{K}(\mathbf{A}, \mathbf{A})$. Hence the standard optimization techniques cannot be applied here because of memory size limitations and computational complexity. Here we present a brief review of support vector machine algorithms that have been extensively developed and used in many applications.

4.1 Iterative Chunking

The simplest heuristic is known as chunking. It starts with an arbitrary subset or “chunk” of the data, and trains an SVM using a generic optimizer on that portion of the data. The algorithm then retains the support vectors from the chunk while discarding the other points and then it uses the hypothesis found

to test the points in the remaining part of the data. The M points that most violate the KKT conditions (where M is a parameter of the system) are added to the support vectors of the previous problem, to form a new chunk. This procedure is iterated, initializing \mathbf{u} for each new sub-problem with the values output from the previous stage, finally halting when some stopping criterion is satisfied. The chunk of data being optimized at a particular stage is sometimes referred to as the working set. Typically the working set grows, though it can also decrease, until in the last iteration the machine is trained on the set of support vectors representing the active constraints.

4.2 Decomposition Method

By using the interesting observation that removing all data points from a training dataset which are not support vectors will not affect the classifier, Osuna, Freund and Girosi [26] proposed a decomposition method. This method iteratively selects a small subset of training data (the working set) to define a quadratic programming subproblem. The solution of current iteration is updated by solving the quadratic programming subproblem, defined by the selected working set, such that the objective function value of the original quadratic program strictly decreases at every iteration. The decomposition algorithm only updates a fixed size subset of multipliers u_i , while the others are kept constant. So every time a new point is added to the working set, another point has to be removed. In this algorithm, the goal is not to identify all of the active constraints in order to run the optimizer on all of them, but is rather to optimize the global problem by only acting on a small subset of data at a time.

The Sequential Minimal Optimization (SMO) algorithm is derived by taking the idea of the decomposition method to its extreme and optimizing a minimal subset of just two points at each iteration. The power of this technique resides in the fact that the optimization problem for two data points admits an analytical solution, eliminating the need to use an iterative quadratic programme optimizer as part of the algorithm. The requirement that the condition $\sum_{i=1}^m y_i u_i = 0$ is enforced throughout the iterations implies that the smallest number of multipliers that can be optimized at each step is 2. The reason is that whenever one multiplier is updated, at least one other multiplier needs to be adjusted in order to keep the condition true.

At each step SMO chooses two elements u_i and u_j to jointly optimize, finds the optimal values for those two parameters given that all the others are fixed, and updates the u vector accordingly. The choice of the two points is determined by a heuristic, while the optimization of the two multipliers is performed analytically. Different strategies to select the working set lead to different algorithms such as BSVM [12] and SVM^{light} [15]. Despite needing more iterations to converge, each iteration uses so few operations that the algorithm exhibits an overall speed-up of some orders of magnitude. Besides convergence time, other important features of the algorithm are that it does

not need to store the kernel matrix in memory, since no matrix operations are involved, that it does not use other packages, and that it is fairly easy to implement. Notice that since standard SMO does not use a cached kernel matrix, its introduction could be used to obtain a further speed-up, at the expense of increased space complexity. The convergence analysis has been carried out in [5, 16].

4.3 Interior Point Method with Row Rank Approximation

Instead of solving a sequence of broken down problems, this approach directly solves the problem as a whole. To avoid solving a linear system involving the large kernel matrix, a row rank approximation to the kernel matrix is often used.

5 Practical Issues in SVMs

In this section, we list some practical issues in SVMs. These topics including how to deal with the multi-class classification, unbalanced data distribution, and model selection.

5.1 Multit-class Problems

In the previous sections, we only focus on the binary classification in SVMs. However, the labels might be drawn from several categories in the real world. There are many methods have been proposed for dealing with the multi-class problem. The most common strategy to handle the multi-class problem is dividing it into a series of binary classification problems.

Two common methods to build such a series of binary classifiers are one-versus-all and one-versus-one. In the one-versus-all scheme, it creates a series of binary classifiers with one of the labels to the rest. The classification of new instances for one-versus-all is using the winner-take-all strategy. That is, we assign the label by the classifier with the highest output value. On the other hand, one-versus-one scheme generate a series of binary classifiers between every pair of classes. The classification of one-versus-one is usually associated with a simple voting strategy. In the voting strategy, every classifier assigns the instance to one of the two classes and then new instances will be classified to a class with most votes.

5.2 Unbalanced Problems

In reality, there might be only a small portion of instances belonging to a class compared to the number of instances with the other label. Due to the small share in a sample that reflects reality, using SVMs on this kind of data may

tend to classify every instance as the class with the majority of the instances. Such models are useless in practice. In order to deal with this problem, the common ways start off with more balanced training than reality can provide.

One of these methods is a down-sampling strategy and work with balanced (50%/50%)-samples. The chosen bootstrap procedure repeatedly randomly selects a fixed number of the majority instances from the training set and adds the same number of the minority instances. However, the random choosing of the majority instances might cause a high variance of the model. In order to avoid this unstable model building, an over-sampling scheme could also be applied to reach a balanced sample. The over-sampling scheme duplicates the number of the minority instances a certain number of times. This over-sampling scheme considers all the instances in hand and will generate a more robust model than the down-sampling scheme.

5.3 Model Selection of SVMs

Model selection is usually done by minimizing an estimate of generalization error. We focus on selecting the regularization parameter and the Gaussian kernel width parameter. This problem can be treated as finding the maximum (or minimum) of a function which is only vaguely specified and has many local maxima (or minima). One standard method to deal with the model selection is to use a simple exhaustive grid search over the parameter space. It is obvious that the exhaustive grid search can not effectively perform the task of automatic model selection due to its high computational cost. Therefore, many improved model selection methods have been proposed to reduce the number of trials in parameter combinations [17, 6, 18, 1, 30, 13]. In this section, we will introduce the simple grid model selection method and the efficient nested uniform design model selection method [13].

As mentioned above, the most common and reliable approach for model selection is exhaustive grid search method. When searching for a good combination of parameters for C and γ , it is usual to form a two dimension uniform grid (say $p \times p$) of points in a pre-specified search range and find a combination (point) that gives the least value for some estimate of generalization error. It is expensive since it requires the trying of $p \times p$ pairs of (C, γ) . The grid method is obviously very clear and simple, but it also has an apparent shortcoming of time-consuming. The grid method along with estimate methods will take a lot of time in model selection. For example, we use a grid method with 400 trying parameter combinations and 10-fold cross-validation for a model selection procedure. This model selection procedure takes about 4000 times of SVMs training for obtaining a good parameter combination.

Except for the exhaustive grid search method, we introduce the 2-stage uniform design model selection. The 2-stage uniform design procedure first sets out a crude search for a highly likely candidate region of global optimum and then confines a finer second-stage search therein. At the first stage, we use a 13-runs UD sampling pattern (see Fig. 3) in the appropriate search

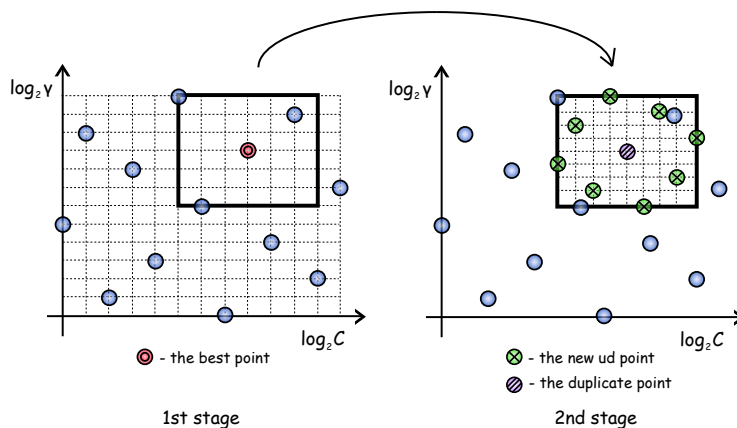


Fig. 3. The nested UD model selection with a 13-points UD at the first stage and a 9-points UD at the second stage

range proposed above. At the second stage, we halve the search range for each parameter coordinate in the log-scale and let the best point from the first stage be the center point of the new search box. We do allow the second stage UD points to fall outside the prescribed search box. Then we use a 9-runs UD sampling pattern in the new range. The total number of parameter combinations is 21 (the duplicate point, i.e., the center point at the second stage, is trained and counted only once). Moreover, to deal with large sized datasets, we combine a 9-runs and a 5-runs sampling pattern at these two stages. The total number of parameter combinations is reduced to 13 (again, the duplicate point, i.e., the center point at the second stage, is trained and counted only once), and the UD based method can still make the resulting SVM model perform well. The numerical results in [13] show merits of the nested UD model selection method. The method of nested UD is not limited to 2 stages and can be applied in a sequential manner and one may consider a finer net of UD to start with. The reason that we use a crude 13-runs or a 9-runs design at the first stage is that it is simply enough for the purpose of model selection in the real data SVM problems

References

1. Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000.
2. Dimitri P. Bertsekas. *Nonlinear programming*. Athena Scientific Belmont, Mass, 1999.
3. Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discover*, 2(2):121–167, 1998.

4. LJ Cao and FEH Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, 2003.
5. Chih-Chung Chang, Chih-Wei Hsu, and Chih-Jen Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.
6. Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.
7. Vladimir Cherkassky and Filip Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, New York, 1998.
8. R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience Publishers, New York, 1953.
9. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 1999.
10. Michael C. Ferris and Todd S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal of Optimization*, 13:783–804, 2003.
11. Glenn M. Fung and Olvi L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational optimization and applications*, 28(2):185–202, 2004.
12. Chih-Wei Hsu and Chih-Jen Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46(1):291–314, 2002.
13. Chien-Ming Huang, Yuh-Jye Lee, Dennis K. J. Lin, and Su-Yun Huang. Model selection for support vector machines via uniform design. *A special issue on Machine Learning and Robust Data Mining of Computational Statistics and Data Analysis*, 52:335–346, 2007.
14. C.O. Inc. Using the cplex callable library and cplex mixed integer library. *Incline Village, NV*, 1992.
15. Thorsten Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169 – 184, 1999.
16. S. S. Keerthi and E. G. Gilbert. Convergence of a generalized smo algorithm for svm. *Machine Learning*, 46(1):351–360, 2002.
17. S. Sathya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.
18. Jan Larsen, Claus Svarer, Lars Nonboe Andersen, and Lars Kai Hansen. Adaptive regularization in neural network modeling. *Lecture notes in computer science*, pages 113–132, 1998.
19. Yuh-Jye Lee and Su-Yun Huang. Reduced support vector machines: A statistical theory. *IEEE Transactions on Neural Networks*, 18(1):1–13, 2007.
20. Yuh-Jye Lee and Olvi L. Mangasarian. Ssvm: A smooth support vector machine for classification. *Computational Optimization and Applications*, 20(1):5–22, 2001.
21. Olvi L. Mangasarian. *Nonlinear programming*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.
22. Olvi L. Mangasarian. *Advances in Large Margin Classifiers*, chapter Generalized Support Vector Machines, pages 135–146. MIT Press, 2000.

23. Jae H. Min and Young-Chan Lee. Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems With Applications*, 28(4):603–614, 2005.
24. David R. Musicant and Er Feinberg. Active set support vector regression. *IEEE Transactions on Neural Networks*, 15(2):268–275, 2004.
25. Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, 2006.
26. Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. 1997.
27. Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.
28. Alexander J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 911–918, 2000.
29. Alexander J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.
30. Carl Staelin. Parameter selection for support vector machines. *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1*, 2003.
31. Robert Tibshiran. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.
32. Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000.
33. Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13, pages 682–688, 2001.
34. Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machine. In *Advances in Neural Information Processing Systems*, volume 13, 2004.