Introduction to Support Vector Machines and Their Applications in Bankruptcy Prognosis

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

We aim at providing a comprehensive introduction to Support Vector Machines and their applications in computational finance. Based on the advances of the statistical learning theory, one of the first SVM algorithms was proposed in mid 90's. Since then, they have drawn a lot of research interests both in theoretical and application domains and have became the state-of-the-art techniques in solving classification and regression problems. The reason for the success is not only because of their sound theoretical foundation but also their good generalization performance in many real applications. In this chapter, we address the theoretical, algorithmic and computational issues and try our best to make the article self-contained. Moreover, in the end of this chapter, a case study on default prediction is also presented. We discuss the issues when SVM algorithms are applied to bankruptcy prognosis 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.

1 Introduction

Finance classification problems occur in credit scoring, company rating, and many fields. One of the most important task is to predict bankruptcy before the disaster. In the era of Basel Committee on Banking Supervision (Basel II), a powerful tool for bankruptcy prognosis can always help banks to reduce their risks. On one hand, the tool must be precise with high accuracy, but also easily adaptable to the bank's objectives regarding the relation of false acceptances (Type I error) and false rejections (Type II error). The prognosis has become even more important since the Basel II established borrowers' rating as the crucial criterion for minimum capital requirements of banks. The methods for generating rating figures have developed significantly over the last 10 years [32].

Parametric statistical models can be used for finance classification. The first introduced model of this type was discriminant analysis (DA) for univariate [4] and multivariate models [2]. After DA, the logit and probit approach for predicting default were proposed in [39] and [42]. These approaches rely on the *a priori* assumed functional dependence between risk of default and predictor. One of the weakest points of DA is that it requires a linear functional, or a preshaped polynomial functional dependence. Such restrictions often fail to meet the reality of observed data. Semi-parametric models as in [27] are between conventional linear models and non-parametric approaches. Other than that, nonlinear classification methods such as Support Vector Machines (SVMs) or neural networks [52] and [3] are even stronger candidates to meet these demands as they go beyond conventional discrimination methods. In this chapter, we concentrate on providing a comprehensive introduction to SVMs and their applications in bankruptcy prognosis.

In the last decade, significant advances have been made in support vector machines (SVMs) both theoretically, by using statistical learning theory; as well as algorithmically, by applying some optimization techniques [7, 18, 36, 38, 47, 49]. 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, an SVM determine an optimal separating hyperplane that classifies data points into different categories. Here, "optimality" refers to the sense that the *separating hyperplane* has the best generalization ability for unseen data points, based on statistical learning theory. With the help of nonlinear kernel functions, SVMs can discriminate between complex data patterns by generating a highly nonlinear separating hyperplane. The nonlinear extension of SVMs makes them applicable to many important real world problems such as character recognition, face detection, analysis of DNA microarrays, breast cancer diagnosis and prognosis [8, 40], and the problem of bankruptcy prognosis as we will see.

The goal of this chapter is to provide a comprehensive introduction to SVMs and their applications in bankruptcy prognosis. The remainder of the chapter is organized as follows. Section 2 introduces the basic ideas and the typical formulation of SVM. Some variants of SVMs are discussed in Section 3 to solve problems of many kinds. Section 4 details some implementation issues and techniques. We discuss solving SVMs in primal and dual forms. In Section 5, to deal with real world problems, we talk about some practical issues of using SVMs. In Section 6, we apply SVMs on a problem of bankruptcy prognosis. Then, in Section 7, we summarize our conclusions.

2 Support Vector Machine Formulations

In this section, we first introduce the basic idea of SVM and give the formulation of *linear* support vector machine. Even the linear version looks too simple to be powerful enough for real applications, it has a non-trivial nonlinear extension. The concept of nonlinear extension of SVM is a milestone for dealing with nonlinear problems and it has a great influence on the machine learning community in this couple of decades. All the details of nonlinear extension, including the "kernel trick" and Mercer's theorem, are introduced in this section. In the end, we discuss the actual risk bound to show the insight behind SVM induction.

2.1 The Formulation of Conventional Support Vector Machine

In this article, we mainly confine ourselves to binary classification problems, which focus on classifying data into two classes. Given a dataset consisting of m points in the n-dimensional real space $\mathbb{R}^n$, each with a class label y, $+1$ or -1, indicating one of two classes, $\mathbf{A}_+$, $\mathbf{A}_- \subseteq \mathbb{R}^n$ where the point belongs, we want to find the decision boundary between the two classes. 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. In notation, we use capital boldface letters to denote a matrix, lower case boldface letters to denote a column vector, and low case light face letters to denote scalars. The data points are denoted by an $m \times n$ matrix $\mathbf{A}$, where the i^{th} row of the matrix corresponds to the i^{th} data point. We use a column vector $\mathbf{x}_i$ to denote the i^{th} data point. All vectors indicate column vectors unless otherwise specified. The transpose of a matrix $\mathbf{M}$ is denoted by $\mathbf{M}^\top$.

Primal Form of Conventional SVM

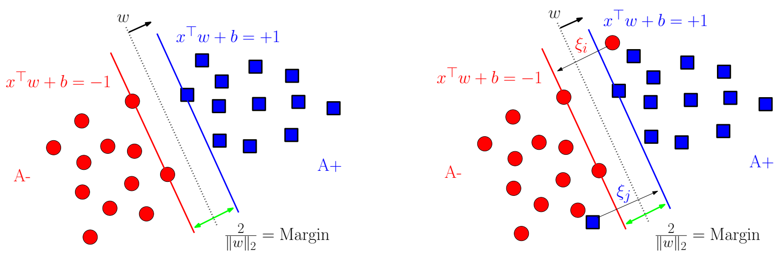
We start with a strictly linearly separable case, *i.e.* there exists a hyperplane which can separate the data $\mathbf{A}_+$ and $\mathbf{A}_-$. In this case we can separate the two classes by a pair of parallel *bounding planes*:

$$\begin{aligned} \mathbf{w}^\top \mathbf{x} + b &= +1\,, \\ \mathbf{w}^\top \mathbf{x} + b &= -1\,, \end{aligned} \tag{1}$$

where $\mathbf{w}$ is the normal vector to these planes and b determines their location relative to the origin. The first plane of (1) bounds the class $\mathbf{A}_+$ and the second plane bounds the class $\mathbf{A}_-$. That is,

$$\begin{aligned} \mathbf{w}^\top \mathbf{x} + b &\geq +1\,, \quad \forall\, \mathbf{x} \in \mathbf{A}_+\,, \\ \mathbf{w}^\top \mathbf{x} + b &\leq -1\,, \quad \forall\, \mathbf{x} \in \mathbf{A}_-\,. \end{aligned} \tag{2}$$

According to the statistical learning theory [54], SVM achieves a better prediction ability via maximizing the margin between two bounding planes. Hence, the "hard margin" 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 the formulation leads to a quadratic program as follows:



(a) linearly separable (b) non-linearly separable

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

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{n+1}} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 \qquad (3)$$
$$\text{s.t.} \quad y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1\,, \quad \text{for } i = 1, 2, \ldots, m\,.$$

The linear separating hyperplane is the plane

$$\mathbf{w}^\top\mathbf{x} + b = 0\,, \qquad (4)$$

midway between the bounding planes (1), as shown in Figure 1(a). For the linearly separable case, the *feasible region* of the above minimization problem (3) is nonempty and the objective function is a quadratic convex function; therefore, there exists an optimal solution, denoted by $(\mathbf{w}^*,\ b^*)$. The data points on the bounding planes, $\mathbf{w}^{*\top}\mathbf{x} + b^* = \pm 1$, are called *support vectors*. It is not difficult to see that, if we remove any point that is not a support vector, the training result will remain the same. This is a nice property of SVM learning algorithms. For the purpose of data compression, once we have the training result, all we need to keep in our database are the support vectors.

If the classes are not linearly separable, in some cases, two planes may bound the two classes with a "soft margin". That is, given a nonnegative slack vector variable $\boldsymbol{\xi} := (\xi_1, \ldots, \xi_m)$, we would like to have:

$$\begin{array}{ll} \mathbf{w}^\top\mathbf{x}_i + b + \xi_i \geq +1\,, & \forall\, \mathbf{x}_i \in \mathbf{A}_+ \\ \mathbf{w}^\top\mathbf{x}_i + b - \xi_i \leq -1\,, & \forall\, \mathbf{x}_i \in \mathbf{A}_-\,. \end{array} \qquad (5)$$

The 1-norm of the slack vector variable $\boldsymbol{\xi}$, $\sum_{i=1}^m \xi_i$, is called the penalty term. In principle, we are going to determine a separating hyperplane that not only correctly classifies the training data, but also performs well on test data. We depict the geometric property in Figure 1(b). With a soft margin, we can extend equation (3) and produce the conventional SVM [54] as the following formulation:

$$\begin{aligned} \min_{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{n+1+m}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top\mathbf{x}_i + b) + \xi_i \geq 1\,, \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \ldots, m\,, \end{aligned} \tag{6}$$

where $C > 0$ is a positive parameter that balances the weight of the penalty term $\sum_{i=1}^{m}\xi_i$ and the margin maximization term $\frac{1}{2}\|\mathbf{w}\|_2^2$. Alternatively, we can replace the penalty term by the 2-norm measure as follows:

$$\begin{aligned} \min_{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{n+1+m}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top\mathbf{x}_i + b) + \xi_i \geq 1\,, \\ & \text{for } i = 1, 2, \ldots, m\,. \end{aligned} \tag{7}$$

The 1-norm penalty is considered less sensitive to outliers than the 2-norm penalty, therefore it receives more attention in real applications. However, mathematically the 1-norm is more difficult to manipulate such as when we need to compute the derivatives.

Dual Form of Conventional SVM

The conventional support vector machine formulation (6) is a standard convex quadratic program [6, 37, 41]. The Wolfe dual problem of (6) is expressed as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i\alpha_i = 0\,, \\ & 0 \leq \alpha_i \leq C \qquad \text{for } i = 1, 2, \ldots, m\,, \end{aligned} \tag{8}$$

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:

$$\mathbf{w} = \sum_{\alpha_i > 0} y_i \alpha_i \mathbf{x}_i\,. \tag{9}$$

Each dual variable α_i corresponds to a training point $\mathbf{x}_i$. The normal vector $\mathbf{w}$ can be expressed in terms of a linear combination of training data points which have corresponding positive dual variables α_i (namely, the support vectors). By the Karush-Kuhn-Tucker complementarity conditions [6, 37]:

$$\begin{array}{lcl} 0 \leq \quad \alpha_i & \perp & y_i(\mathbf{w}^\top\mathbf{x}_i + b) + \xi_i - 1 \geq 0 \\ 0 \leq C - \alpha_i & \perp & \xi_i \geq 0 \quad , \quad \text{for } i = 1, 2, \ldots, m\,, \end{array} \tag{10}$$

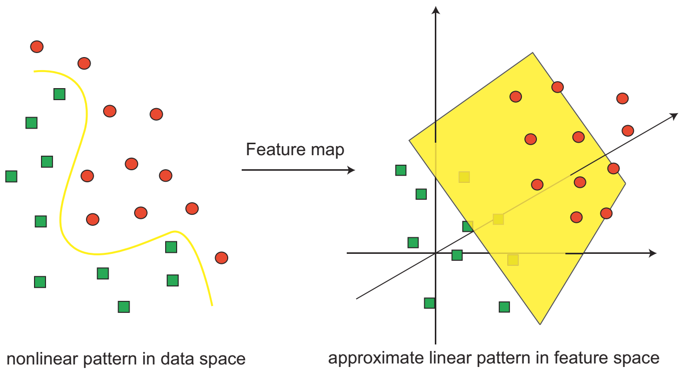

Feature map
nonlinear pattern in data space
approximate linear pattern in feature space

Fig. 2. The illustration of nonlinear SVM

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

$$b = y_i - \mathbf{w}^\top \mathbf{x}_i = y_i - \sum_{j=1}^{m} (y_j \alpha_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle) \ . \tag{11}$$

In the dual form, SVMs can be expressed by the form of inner product. It implies that we only need the information of the inner product of the data when expressing the formulation and decision function of SVM. This important characteristic carries SVMs to their nonlinear extension in a simple way.

2.2 Nonlinear Extension of SVMs via Kernel Trick

In many cases, a dataset, as collected in a vector form full of attributes, cannot be well separated by a linear separating hyperplane. However, it is likely that the dataset becomes linearly separable after mapped into a higher dimensional space by a nonlinear map. A nice property of SVM methodology is that we do not even need to know the nonlinear map explicitly; still, we can apply a linear algorithm to the classification problem in the high dimensional space. The property comes from the dual form of SVM which can express the formulation in terms of inner product of data points. By taking the advantage of dual form, the "kernel trick" is used for the nonlinear extension of SVM.

Kernel Trick

From the dual SVM formulation (8), all we need to know is simply the inner product between training data vectors. 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 $\mathbf{x}$ in $\mathcal{F}$ becomes $\Phi(\mathbf{x}) \in \mathbb{R}^\ell$ where ℓ is the dimensionality of the feature space $\mathcal{F}$. Based on the above observation, if we know the inner 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 is a nonlinear surface in the input space $\mathbb{R}^n$ (see Fig. 2).

Note that we do not need to know the nonlinear map Φ explicitly. It can be achieved by employing a kernel function. Let $k(\mathbf{x}, \mathbf{z}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be an inner product kernel function satisfying *Mercer's condition* [7, 14, 15, 18, 54], positive semi-definiteness condition (see Definition 1). We can construct a nonlinear map Φ such that $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top\Phi(\mathbf{x}_j)$ where $i, j = 1, 2, \dots, m$. Hence, the linear SVM formulation can be used 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 (8) with a nonlinear kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. The resulting dual nonlinear SVM formulation becomes:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^{m} y_i \alpha_i = 0 \\ & 0 \le \alpha_i \le C \quad \text{for } i = 1, 2, \dots, m\,. \end{aligned} \tag{12}$$

The nonlinear separating hyperplane is defined by the solution of (12) as follows:

$$\sum_{j=1}^{m} (y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i)) + b = 0\,, \tag{13}$$

where

$$b = y_i - \sum_{j=1}^{m} (y_j \alpha_j k(\mathbf{x}_j, \mathbf{x}_i)), \quad i \in I := \{k \mid 0 < \alpha_k < C\}\,. \tag{14}$$

The "kernel trick" makes the nonlinear extension of linear SVM possible without knowing the nonlinear mapping explicitly. Whatever computation code ready for linear SVM can also be modified to the nonlinear version easily with a substitution of the inner product computation in the input space by the inner product computation in the feature space.

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, it is not always possible for a given function $k(\mathbf{x}, \mathbf{z})$ to reconstruct its corresponding nonlinear maps. We can answer the question by the so-called *Mercer's condition* [54]. We conclude this section with *Mercer's condition* and two examples of kernel function.

Definition 1 (Mercer's condition). *Let $k(\mathbf{s}, \mathbf{t}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a continuous symmetric function and X be a compact subset of $\mathbb{R}^n$. If*

$$\int_{X\times X} k(\mathbf{s},\mathbf{t})f(\mathbf{s})f(\mathbf{t})\,d\mathbf{s}\,d\mathbf{t} \geq 0,\ \forall f \in \ell_2(X)\,, \tag{15}$$

where the Hilbert space $\ell_2(X)$ is the set of functions f such that

$$\int_X f(\mathbf{t})^2\,d\mathbf{t} < \infty\,. \tag{16}$$

then the function k satisfies Mercer's condition.

This is equivalent to say that the kernel matrix $\mathbf{K}(\mathbf{A},\mathbf{A})$ in our application is positive semi-definite [18], where $\mathbf{K}(\mathbf{A},\mathbf{A})_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$ for $i,j = 1,2,\ldots,m$. Below are two most popular kernel functions in real applications. The choice of kernel functions may rely on the result of a *cross-validation* or model selection procedure.

Example 1. Polynomial Kernel

$$k(\mathbf{x},\mathbf{z}) = (\mathbf{x}^\top\mathbf{z} + b)^d\,, \tag{17}$$

where d denotes the degree of the exponentiation.

Example 2. Gaussian (Radial Basis) Kernel

$$k(\mathbf{x},\mathbf{z}) = e^{-\gamma\|\mathbf{x}-\mathbf{z}\|_2^2}\,, \tag{18}$$

where γ is the width parameter of Gaussian kernel.

2.3 A Bound on Actual Risk

The main goal of the classification problem is to predict the label of new unseen data points correctly. That is, we seek for a classifier $f(\mathbf{x},\alpha)$ with output values 1 and -1 that can minimize the following test error:

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x},\alpha)|\,dP(\mathbf{x},y)\,, \tag{19}$$

where $\mathbf{x}$ is an instance and y is the class label of $\mathbf{x}$, with $(\mathbf{x},y)$ drawn from some unknown probability distribution $P(\mathbf{x},y)$, and α is an adjustable parameter of f. The error, so called the actual risk, in which we are interested can represent the true mean error but it needs to know what $P(\mathbf{x},y)$ is. However, estimating $P(\mathbf{x},y)$ is usually not possible so that (19) is not very useful in practical usage. The usual way is to approximate the actual risk by using the empirical risk:

$$R_{emp}(\alpha) = \frac{1}{2m}\sum_{i=1}^{m}|y_i - f(\mathbf{x}_i,y)|\,. \tag{20}$$

This empirical risk is obtained by considering only a finite number of training data. Looking for a model that fits the given dataset usually is not a good way to do. There always exists a model that can classify the training data perfectly as long as there is no identical data points that have different labels. However this model might overfit the training data and perform poorly on the unseen data. 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.* Vapnik et al. [54] proposed a upper bound for $R(\alpha)$ with probability $1 - \eta$ as follows:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2m/h) + 1) - \log(\eta/4)}{m}}\,, \tag{21}$$

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. The second term on the right-hand side of (21) is called the VC confidence.

The upper bound in (21) gives a principle for choosing a learning model for a given task. Thus given several different learning models and a fixed, sufficiently small η, choosing a model that minimizes the right-hand side is equivalent to choosing a model that 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 may 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 (21). This idea of balancing the model complexity and empirical risk is considered in SVMs. The objective functions of (6) and (7) can be interpreted as the upper bound of actual risk in (21) [7, 54]. Basically, SVM 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 and lead to poor generalization. The positive parameter C which can be determined by a *tuning procedure* such as cross-validation, plays the role of balancing this trade-off. We will discuss the issue in more details in Section 5.

3 Variants of Support Vector Machines

Since the typical SVM was proposed for the first time in late 90's, to deal with various kinds of applications, many variants of SVM have been proposed. The different formulations of SVM have their own approaches in dealing with data. In this section, we will introduce some of them, as well as their properties and applications.

3.1 Smooth Support Vector Machine

In contrast to the conventional SVM of (6), smooth support vector machine (SSVM) [36] minimizes the square of the slack vector $\boldsymbol{\xi}$. In addition, the SSVM prefers a solution with a small value of b (also in 2-norm). That leads to the following minimization problem:

$$\begin{aligned} \min_{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{n+1+m}} \quad & \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top\mathbf{x}_i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \ldots, m\,. \end{aligned} \tag{22}$$

As a solution of (22), $\boldsymbol{\xi}$ 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 (22) by $\{1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\}_+$. It converts the problem (22) into an unconstrained minimization problem as follows:

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

Compared to (22), this formulation reduces the number of variables from $n + 1 + m$ to $n + 1$; however, the objective function to be minimized is no longer twice differentiable. In SSVM, we prefer a twice differentiable form so that a fast Newton method can be applied. We approximate the plus function x_+ by a smooth p-function:

$$p(x, \beta) = x + \frac{1}{\beta}\log(1 + e^{-\beta x})\,, \tag{24}$$

where $\beta > 0$ is the smooth parameter which controls the "steepness" of the curve or how close it is to the original plus function x_+. By replacing the plus function x_+ with a very accurate approximation p-function gives the SSVM formulation:

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{n+1}} \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{m} p(\{1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\}, \beta)^2\,, \tag{25}$$

The objective function in problem (25) is strongly convex and infinitely differentiable. Hence, it has a unique solution and can be solved by using a fast *Newton-Armijo* algorithm (discussed in the implementation part, Section 4). For the nonlinear case, this formulation can be extended to the nonlinear version by utilizing the kernel trick as follows:

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{m+1}} \frac{1}{2}(\|\mathbf{u}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{m} p([1 - y_i\,\{\sum_{j=1}^{m} u_j k(\mathbf{x}_i, \mathbf{x}_j) + b\}\,], \beta)^2\,, \tag{26}$$

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

$$f(\mathbf{x}) = \sum_{u_j \neq 0} u_j k(\mathbf{x}_j, \mathbf{x}) + b\,. \tag{27}$$

3.2 Reduced Smooth Support Vector Machine

In these days, very often we have classification or regression problems with large-scale data, such as the data from network traffic, gene expressions, web documents, etc. To solve large-scale problems by SVM, the full kernel matrix will be very large, so it may not be appropriate to use the full matrix when dealing with (26). In order to avoid facing such a large full matrix, we brought in the *reduced* kernel technique [35]. The key idea of the reduced kernel technique is to randomly select a small 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 [48, 56] for the full kernel matrix:

$$\mathbf{K}(\mathbf{A},\mathbf{A}) \approx \mathbf{K}(\mathbf{A},\tilde{\mathbf{A}})\mathbf{K}(\tilde{\mathbf{A}},\tilde{\mathbf{A}})^{-1}\mathbf{K}(\tilde{\mathbf{A}},\mathbf{A})\,, \tag{28}$$

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

$$\mathbf{K}(\mathbf{A},\mathbf{A})\mathbf{u} \approx \mathbf{K}(\mathbf{A},\tilde{\mathbf{A}})\mathbf{K}(\tilde{\mathbf{A}},\tilde{\mathbf{A}})^{-\mathbf{1}}\mathbf{K}(\tilde{\mathbf{A}},\mathbf{A})\mathbf{u} \;= \mathbf{K}(\mathbf{A},\tilde{\mathbf{A}})\tilde{\mathbf{u}}\,, \tag{29}$$

where $\tilde{\mathbf{u}} \in \mathbb{R}^{\tilde{m}}$ is an approximated solution of $\mathbf{u}$ via the reduced kernel technique. By using the approximation, reduced SVM randomly selects a small subset $\tilde{\mathbf{A}}$ to generate the basis functions $\mathcal{B}$:

$$\mathcal{B}= \{1\} \cup \left\{k(\cdot,\tilde{x}^i)\right\}_{i=1}^{\tilde{m}}\,.$$

The formulation of reduced SSVM, hence, is expressed as follows:

$$\min_{\tilde{u},b,\boldsymbol{\xi}} \; \frac{1}{2}(\left\|\tilde{u}\right\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{\tilde{m}} p([1 - y_i\{\sum_{j=1}^{\tilde{m}} \tilde{u}_j k(\mathbf{x}_i,\tilde{\mathbf{x}}_j) + b\}],\beta)^2 \tag{30}$$

and its decision function is in the form

$$f(x) = \sum_{i=1}^{\tilde{m}} \tilde{u}_i k(\mathbf{x},\tilde{\mathbf{x}}_i) + b\,. \tag{31}$$

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 [35].

3.3 Least Squares Support Vector Machine

The least squares support vector machine [51] considers the equality constraints which make the formulation of the classification problem in the sense of least squares as follows:

$$\begin{aligned}\min_{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{n+1+m}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i^2 \\ \text{s.t.} \quad & \xi_i = 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b) \text{ for } i = 1,2,\ldots,m\,.\end{aligned} \tag{32}$$

The same idea, called proximal support vector machine, is also proposed simultaneously in [22], with adding the square of the bias term b in the objective function. With the least squares form, one can obtain the solution of the classification problem via solving a set of linear equations. Consider the Lagrangian function of (32):

$$\mathcal{L}(\mathbf{w},b,\boldsymbol{\xi};\boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i^2 - \sum_{i=1}^{m}\alpha_i[y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 + \xi_i]\,, \tag{33}$$

where $\alpha_i \in \mathbb{R}$ are Lagrange multipliers. Setting the gradient of $\mathcal{L}$ to zeros gives the following Karush-Kuhn-Tucker optimality conditions:

$$\begin{aligned}&\mathbf{w} = \sum_{i=1}^{m}\alpha_i y_i \mathbf{x}_i \qquad (34)\\ &\sum_{i=1}^{m}\alpha_i y_i = 0\\ &\alpha_i = C\xi_i, \quad i = 1,\ldots,m\\ &y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 + \xi_i = 0\,,\end{aligned}$$

which are equivalent to the following linear equations:

$$\begin{bmatrix} I & 0 & 0 & -\hat{\mathbf{A}}^\top \\ 0 & 0 & 0 & -\mathbf{y}^\top \\ 0 & 0 & C\mathbf{I} & -\mathbf{I} \\ \hat{\mathbf{A}} & \mathbf{y} & \mathbf{I} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \\ \boldsymbol{\xi} \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1} \end{bmatrix}, \tag{35}$$

or, equivalently,

$$\begin{bmatrix} 0 & -\mathbf{y}^\top \\ \mathbf{y} & \hat{\mathbf{A}}\hat{\mathbf{A}}^\top + \frac{1}{C}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{36}$$

where $\hat{\mathbf{A}} = [\mathbf{x}_1 y_1; \mathbf{x}_2 y_2; \ldots; \mathbf{x}_m y_m]$, $\mathbf{y} = [y_1; y_2; \ldots; y_m]$, and $\mathbf{1} = [1; 1; \ldots; 1]$. From (36), the nonlinear least squares SVM also can be extended via the inner product form. That is, the nonlinear least squares SVM solves the following linear equations:

$$\begin{bmatrix} 0 & -\mathbf{y}^\top \\ \mathbf{y} & \mathbf{K}(\mathbf{A},\mathbf{A}) + \frac{1}{C}\mathbf{I} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{37}$$

where $\mathbf{K}(\mathbf{A},\mathbf{A})$ is the kernel matrix. The (36) or (37) gives an analytic solution to the classification problem via solving a system of linear equations. This brings a lower computational cost by comparing with solving a conventional SVM while obtaining a least squares SVM classifier.

3.4 1-norm Support Vector Machine

The 1-norm support vector machine replaces the regularization term $\|\mathbf{w}\|_2^2$ in (6) by a ℓ_1-norm of $\mathbf{w}$. The ℓ_1-norm regularization term is also called the LASSO penalty [53]. It tends to shrink the coefficients $\mathbf{w}$'s towards zeros in particular for those coefficients corresponding to redundant noise features [57]. This nice feature will lead to a way of selecting the important attributes in our prediction model. The formulation of 1-norm SVM is described as follows:

$$\begin{aligned} \min_{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{n+1+m}} \quad & \|\mathbf{w}\|_1 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top\mathbf{x}_i + b) + \xi_i \geq 1 \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \ldots, m\,. \end{aligned} \tag{38}$$

The objective function of (38) is a piecewise linear convex function. We can reformulate it as the following linear programming problem:

$$\begin{aligned} \min_{(\mathbf{w},\mathbf{s},b,\boldsymbol{\xi})\in\mathbb{R}^{n+n+1+m}} \quad & \sum_{j=1}^{n} s_j + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top\mathbf{x}_i + b) + \xi_i \geq 1 \\ & -s_j \leq w_j \leq s_j, \quad \text{for } j = 1, 2, \ldots, n\,, \\ & \xi_i \geq 0, \quad \text{for } i = 1, 2, \ldots, m\,, \end{aligned} \tag{39}$$

where s_j is the upper bound of the absolute value of w_j. At the optimal solution of (39) 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 [23, 57]. Therefore the 1-norm SVM can be a very promising tool for *variable selection.* In Section 6, we will use it to choose the important financial indices for our bankruptcy prognosis model.

3.5 ε-Support Vector Regression

In regression problems, the response $\mathbf{y}$ belongs to real numbers. We would like to find a linear or nonlinear regression function, $f(\mathbf{x})$, that tolerates a small error in fitting the given dataset. It can be achieved by utilizing the ε-insensitive loss function that sets an ε-insensitive "tube" around the data, within which errors are discarded.

We start with the linear case, that is the regression function $f(\mathbf{x})$ defined as $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b$. The SVM minimization can be formulated as an unconstrained problem given by:

$$\min_{(\mathbf{w},b,\boldsymbol{\xi})\in\mathbb{R}^{n+1}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}|\xi_i|_\varepsilon\,, \tag{40}$$

where $|\xi_i|_\varepsilon = \max\{0, |\mathbf{w}^\top\mathbf{x}_i + b - y_i| - \varepsilon\}$, 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 $f(\mathbf{x})$. Similar to the idea in SVM, the regularization term $\|\mathbf{w}\|_2^2$ in (40) is also applied for improving the generalization ability. 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:

$$\begin{aligned}\min_{(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*)\in R^{n+1+2m}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & \mathbf{w}^\top\mathbf{x}_i + b - y_i \le \varepsilon + \xi_i\,, \\ & -\mathbf{w}^\top\mathbf{x}_i - b + y_i \le \varepsilon + \xi_i^*\,, \\ & \xi_i, \xi_i^* \ge 0 \text{ for } i = 1, 2, \ldots, m\,.\end{aligned} \tag{41}$$

This formulation (41) is equivalent to the formulation (40) and its corresponding dual form is

$$\begin{aligned}\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}\in\mathbb{R}^m} \quad & \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)y_i - \varepsilon\sum_{i=1}^{m}(\hat{\alpha}_i + \alpha_i) \\ & -\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\langle\mathbf{x}_i, \mathbf{x}_j\rangle\,, \\ \text{s.t.} \quad & \sum_{i=1}^{m}(\hat{u}_i - u_i) = 0\,, \\ & 0 \le \alpha_i, \hat{\alpha}_i \le C\,, \quad \text{for } i = 1\,, \ldots, m\,.\end{aligned} \tag{42}$$

From (42), one also can apply the kernel trick on this dual form of ε-SVR for the nonlinear extension. That is, $\langle\mathbf{x}_i, \mathbf{x}_j\rangle$ is directly replaced by a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ as follows:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}\in\mathbb{R}^m} \quad & \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)y_i - \varepsilon\sum_{i=1}^{m}(\hat{\alpha}_i + \alpha_i) \\ & -\sum_{i=1}^{m}\sum_{j=1}^{m}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)k(\mathbf{x}_i, \mathbf{x}_j)\,, \\ \text{s.t.} \quad & \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i) = 0\,, \\ & 0 \le \alpha_i, \hat{\alpha}_i \le C\,, \quad \text{for } i = 1\,, \ldots, m\,.\end{aligned} \tag{43}$$

with the decision function $f(\mathbf{x}) = \sum\limits_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)k(\mathbf{x}_i, \mathbf{x}) + b\,.$

Similar to the smooth approach in SSVM, the formulation (40) can be modified slightly as a smooth unconstrained minimization problem. Before we derive the smooth approximation function, we show some interesting observations:

$$|x|_\varepsilon = (x-\varepsilon)_+ + (-x-\varepsilon)_+ \tag{44}$$

and

$$(x-\varepsilon)_+ \cdot (-x-\varepsilon)_+ = 0 \quad \text{for all } x \in \mathbb{R} \text{ and } \varepsilon > 0\,. \tag{45}$$

Thus we have

$$|x|^2_\varepsilon = (x-\varepsilon)^2_+ + (-x-\varepsilon)^2_+\,. \tag{46}$$

It is straightforward to replace $|x|^2_\varepsilon$ by a very accurate smooth approximation given by:

$$p^2_\varepsilon(x,\beta) = (p(x-\varepsilon,\beta))^2 + (p(-x-\varepsilon,\beta))^2\,. \tag{47}$$

We use this approximation p^2_ε-function with smoothing parameter β to obtain the smooth support vector regression (ε-SSVR) [34]:

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{n+1}} \frac{1}{2}(\|\mathbf{w}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{m} p^2_\varepsilon(\mathbf{w}^\top \mathbf{x}_i + b - y_i, \beta)\,, \tag{48}$$

where $p^2_\varepsilon(\mathbf{w}^\top \mathbf{x}_i + b - y_i, \beta) \in \mathbb{R}$. For the nonlinear case, this formulation can be extended to the nonlinear ε-SSVR by using the kernel trick as follows:

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{m+1}} \frac{1}{2}(\|\mathbf{u}\|_2^2 + b^2) + \frac{C}{2}\sum_{i=1}^{m} p^2_\varepsilon(\sum_{j=1}^{m} u_j K(\mathbf{x}_j, \mathbf{x}_i) + b - y_i, \beta)\,, \tag{49}$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function. The nonlinear ε-SSVR decision function $f(\mathbf{x})$ can be expressed as follows:

$$f(x) = \sum_{i=1}^{m} u_i k(\mathbf{x}_j, \mathbf{x}) + b\,. \tag{50}$$

Note that the reduced kernel technique also can be applied to ε-SSVR while encountering a large scale regression problem.

4 Implementation of SVMs

The support vector machine, either in its primal formulation (6) or dual formulation (8), is simply a standard convex quadratic program (for the nonlinear SVM, the kernel function $k(\mathbf{x}, \mathbf{x})$ used in (12) 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 [28], or using an interior point method for quadratic programming [21]. Because of the simple structure of the dual formulation of either linear (8) or nonlinear (12) SVM, many SVM algorithms are operated in the dual space. However solving SVMs in the primal form can also be efficient [36, 34, 35, 11], such as the approaches to solve SSVM, SSVR, and RSVM which were introduced in the previous section. In the following, we will introduce main methods in solving SVMs in their primal and dual forms.

4.1 SVMs Training in the Primal Form

The standard way to solve SVMs in the primal is reformulating (6) or (7) as an unconstrained minimization problem:

$$\min_{(\mathbf{w},b)\in\mathbb{R}^{n+1}} \tfrac{1}{2}\|\mathbf{w}\|_2^2 + C\textstyle\sum_{i=1}^{m} L(y_i, \mathbf{w}^\top\mathbf{x}+b)\,, \tag{51}$$

with the loss function $L(y, f(\mathbf{x})) = \max(0, 1-y_i f(\mathbf{x}))^p$. Note that the decision function can be written as a linear combination of data points such as $\mathbf{w} = \sum_i^m u_i\mathbf{x}_i$. Thus, we can rewrite the nonlinear form for SVMs in the primal by utilizing kernel trick as follows

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{m+1}} \tfrac{1}{2}\mathbf{u}^\top\mathbf{K}(\mathbf{A},\mathbf{A})\mathbf{u} + C\textstyle\sum_{i=1}^{m} L(y_i, \sum_{j=1}^{m} u_j k(\mathbf{x}_i,\mathbf{x}_j)+b)\,, \tag{52}$$

or in another slightly different form based on the generalized SVM [38]:

$$\min_{(\mathbf{u},b)\in\mathbb{R}^{m+1}} \tfrac{1}{2}\mathbf{u}^\top\mathbf{u} + C\textstyle\sum_{i=1}^{m} L(y_i, \sum_{j=1}^{m} u_j k(\mathbf{x}_i,\mathbf{x}_j)+b)\,. \tag{53}$$

For solving unconstrained minimization problems, Newton-like optimization methods are widely used, so we only focus on solving the minimization problems via Newton method here. The Newton method needs the objective function to be twice differentiable to calculate the Hessian matrix. One way is to replace the loss function by a twice differentiable approximated function. We take SSVM as an example to illustrate the idea. SSVM adopts the quadratic loss and uses p-function to smooth the quadratic loss function as follows:

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

for the linear case; and for the nonlinear case, the function becomes:

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

Once reformulating SVM as an unconstrained minimization problem with twice differentiable objective function $\mathbf{\Psi}(\mathbf{w},b)$ (or $\mathbf{\Psi}(\mathbf{u},b)$), Newton-Armijo optimization method is applied to obtain the solution. The Armijo condition

$$\mathbf{\Psi}(\mathbf{w}_i, b_i) \leq \mathbf{\Psi}(\mathbf{w}_{i-1}, b_{i-1}) - \eta\lambda \mathbf{d}^\top \nabla \mathbf{\Psi}(\mathbf{w}, b)$$

is applied here to avoid the divergence and oscillation in Newton method where η is assigned with a small value. For the nonlinear case, one only needs to replace the original data $\mathbf{A}$ by the kernel data $\mathbf{K}(\mathbf{A}, \mathbf{A})$ or reduced kernel data $\mathbf{K}(\mathbf{A}, \tilde{\mathbf{A}})$ and simply obtains the solution without revising the algorithm.

4.2 SVMs Training in the Dual Form

The most popular strategy in solving SVM with dual form is the decomposition method [43, 29, 20, 24]. The decomposition method is designed to avoid the access of the full kernel matrix while searching for the optimal solution. 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 a selected working set $\mathcal{W}$, 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 α_i, while the others are kept constant. 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.

Suppose suppose α' are the coefficients of data belonging to the current working set $\mathcal{W}$. One can reformulate (8) to a subproblem and solve it iteratively for updating α as follows:

$$\begin{aligned} \max_{\alpha'} \quad & \sum_{i\in\mathcal{B}} \alpha'_i - \frac{1}{2} \sum_{i,j\in\mathcal{B}} y_i y_j \alpha'_i \alpha'_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha'_i \leq C \quad \text{for } i \in \mathcal{W}\,, \\ & \sum_{i\in\mathcal{B}} y_i \alpha'_i + \sum_{i\notin\mathcal{B}} y_i \alpha_i = 0\,. \end{aligned} \tag{54}$$

The critical issue of decomposition methods is selecting an appropriate working set. The sequential minimal optimization (SMO) [44] which is an extreme case of the decomposition methods. It only selects a working set with smallest size, two data points, at each iteration. The criterion of selecting these two data points is based on the maximum violating pair scheme. Besides, this smallest working size leads the subproblem to a single variable minimization problem which has a analytic form of solution. Different strategies to select the working set lead to different algorithms. Many methods of selecting a appropriate working set has been proposed [29, 20, 24]. Some of them also been well implemented, such as SVM^{light}[1] [29], and LIBSVM[2] [10]. The LIBSVM provides an advanced working set selection scheme based on the information

[1] SVM^{light} is available in `http://svmlight.joachims.org/`

[2] LIBSVM is available in `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

of seconde order. Its efficiency in performance has attracted many people to use in their applications.

In a nutshell, decomposition methods take the advantage of sparsity in SVM to adjust the solution with a small minimization problem iteratively. This strategy makes decomposition methods avoid to access the whole full kernel in seeking the solution. On the other hand, the selection of working set is a key factor for the computational cost. Different working sets and their sizes lead to different rates of convergence. The convergence analysis has been carried out in [9, 30].

5 Practical Issues and Their Solutions in SVMs

In this section, we discuss some practical issues in SVMs. The topics including dealing with the multi-class classification, dealing with unbalanced data distribution, and the strategy of model selection.

5.1 Multi-class Problems

In the previous sections, we only focus on the binary classification problem in SVM. 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. These methods can simply be divided into two types. One handles the multi-class problem by dividing it into a series of binary classification problems [54, 45, 17, 46]. The other formulates the multi-class problem as a single optimization problem [54, 55, 16, 46].

In the approach of combining a series of binary classifiers, the popular schemes are one-versus-rest, one-versus-one, directed acyclic graph (DAG) [45], and error-correcting coding [19, 1, 17]. Now suppose we have k classes in the data. In the one-versus-rest scheme, it creates a series of binary classifiers with one of the labels to the rest so we have k binary classifiers for prediction. The classification of new instances for one-versus-rest 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 generates a series of binary classifiers between every pair of classes. It means we need to construct $\binom{k}{2}$ classifiers in the one-versus-one scheme. 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 certain class with most votes. The DAG strategy is a variant of one-versus-one scheme. It also constructs $\binom{k}{2}$ classifiers for each pair of classes but uses a different prediction strategy. DAG places the $\binom{k}{2}$ classifiers in a directed acyclic graph and each path from the root to a leaf is an evaluation path. In an evaluation path, a possible labeling is eliminated while passing through a binary classification node. A predicted label is concluded after finishing a evaluation path (see Fig. 3). In the error-correcting coding

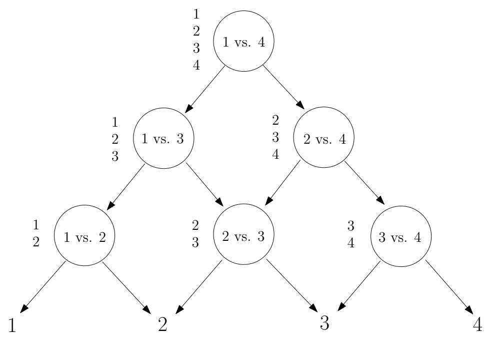


Fig. 3. An example of DAG approach in the multi-class problem

scheme, output coding for multi-class problems consists of two phases. In the training phase, one need to construct a series of binary classifiers which are based on different partitions of the classes. In the testing phase, the predictions of the binary classifiers are combined to conclude a prediction of a testing instance by using the output coding. Besides, the coding scheme is an issue in the error-correcting coding. There are rich literatures discussing the coding schemes [19, 1, 17]. The reader could get more details in these literatures.

The single machine approach for multi-class problem is first introduced in [54, 55]. The idea behind this approach is still using the concept of maximum margin in binary classification. The difference of single machine formulation is that it considers all regularization terms together and pays the penalties for a misclassified instance with a relative quantity evaluated by different models. It means that each instance is associated with $m(k-1)$ slack values if we have m instances and k classes. For understanding the concept more, we display the formulation of single machine approach in [55]:

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k\in\mathbb{R}^n,\boldsymbol{\xi}\in\mathbb{R}^{m(k-1)}} \quad & \sum_{i=1}^{k}\|\mathbf{w}_i\| + C\sum_{i=1}^{m}\sum_{j\notin y_i}\xi_{ij} \\ \text{s.t.} \quad & \mathbf{w}_{y_i}^{\top}\mathbf{x}_i + b_{y_i} \geq \mathbf{w}_j^{\top}\mathbf{x}_i + b_j + 2 - \xi_{ij}\,, \\ & \xi_{ij} \geq 0\,. \end{aligned} \tag{55}$$

Except for this basic formulation, some further formulations have also been proposed [54, 16, 46]. In a nutshell, the single machine approach could give all the classifiers simultaneously in solving a single optimization problem. However, the complicated formulation also brings a higher complexity for solving it.

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 [13] 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. One advantage of down-sampling strategy is giving a lower cost in the training phase because it removes lots of data points in the majority class. However, the random choosing of the majority instances might cause a high variance of the model.

In order to avoid this unstable model building, a over-sampling scheme [25] 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. It considers all the instances in hand and generates a more robust model than the down-sampling scheme. Comparing the computational cost with down-sampling strategy, over-sampling suffers a higher cost in the training phase while increasing the size of training data.

To avoid the extra cost in the over-sampling strategy, one also can apply different weights on the penalty term. In other words, one need to assign a higher weight (higher C) on the minority class. This strategy of assigning different weights gives the equivalent effect with the over-sampling strategy. The benefit of assigning different weights is that it does not increase the size of training data while achieving a balanced training. However, using this strategy needs to revise the algorithm a little bit. In down-sampling and over-sampling strategies, the thing that one needs to do is adjusting the proportions of training data. Hence, down-sampling and over-sampling strategies are easier to be applied for basic users in practical usage.

5.3 Model Selection of SVMs

Choosing a good parameter setting for a better generalization performance of SVMs is the so called model selection problem. Model selection is usually done by minimizing an estimate of generalization error. 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).

Suppose the Gaussian kernel

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

is used where γ is the width parameter. The nonlinear SVM needs to be assigned two parameters C and γ. The most common and reliable approach for model selection is exhaustive grid search method. The exhaustive grid search method forms a two dimension uniform grid (say $p \times p$) of points in a pre-specified search range and find a good combination (C, γ). It is obvious that

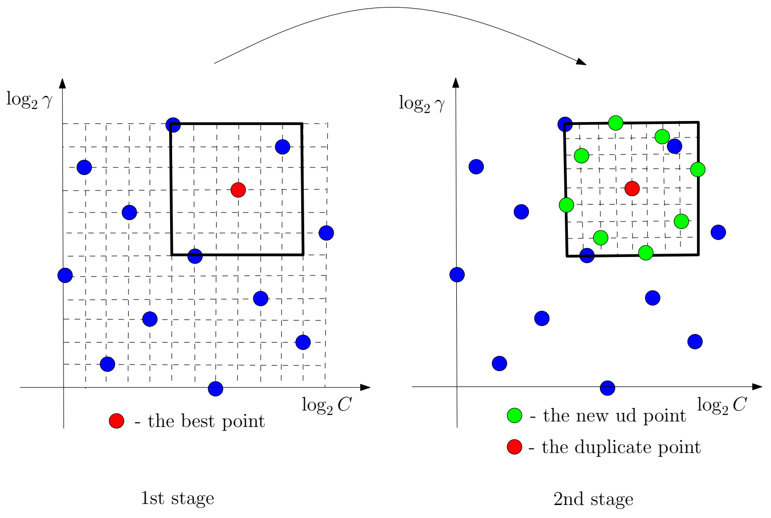


Fig. 4. The nested UD model selection with a 13-points UD at the first stage and a 9-points UD at the second stage

the exhaustive grid search can not effectively perform the task of automatic model selection due to its high computational cost.

Except for the exhaustive grid search method, many improved model selection methods have been proposed to reduce the number of trials in parameter combinations [31, 12, 33, 5, 50, 26]. Here we focus on introducing the 2-stage uniform design model selection [26] because of its good efficiency. 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 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. Then we use a 9-runs UD sampling pattern in the new range. Moreover, to deal with large sized datasets, we combine a 9-runs and a 5-runs sampling pattern at these two stages. The performance in [26] shows merits of the nested UD model selection method. Besides, the method of nested UDs is not limited to 2 stages and can be applied in a sequential manner and one may consider a finer net of UDs to start with.

6 A Case Study for Bankruptcy Prognosis

To demonstrate the use of SVM, we focus on the problem of bankruptcy prognosis as our case study. The studied data set is `CreditReform` where we are given financial company information and the goal is to predict the

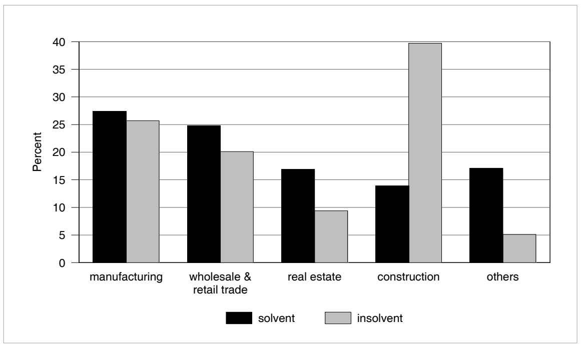


Fig. 5. The distribution of solvent and insolvent companies across industries

possibility of bankruptcy for the companies. The study includes applying the nonlinear SSVM with a reduced kernel, feature selection via 1-norm SVM, conquering the unbalanced problem by over-sampling technique, and model selection by the 2-stage nested uniform design method.

6.1 Data Description

The `CreditReform` database consists of 20,000 financially solvent and 1,000 insolvent German companies observed once in the period from 1997 to 2002. Although the companies were randomly selected, the accounting data in 2001 and 2002 are the majority. Approximately 50% of the observations come from this period. Figure 5 shows the distribution of solvent and insolvent companies across different industries.

A company is described by a set of attributes that includes several balance sheet and income statement items. The attributes include:

- AD (Amortization and Depreciation)
- AP (Accounts Payable)
- AR (Account Receivable)
- CA (Current Assets)
- CASH (Cash and Cash Equivalents)
- CL (Current Liabilities)
- DEBT (Debt)
- EBIT (Earnings before Interest and Tax)
- EQUITY (Equity)
- IDINV (Growth of Inventories)
- IDL (Growth of Liabilities)
- INTE (Interest Expense)
- INV (Inventories)
- ITGA (Intangible Assets)
- LB (Lands and Buildings)

- NI (Net Income)
- OI (Operating Income)
- QA (Quick Assets)
- SALE (Sales)
- TA (Total Assets)
- TL (Total Liabilities)
- WC (Working Capital (=CA-CL))

The companies may appear in the database several times in different years; however, each year of balance sheet information is treated as a single observation. The data of the insolvent companies were collected two years prior to their insolvency. The company size is measured by its total assets. We construct 28 ratios to condense the balance sheet information (see Table 1). However, before dealing with the data set, some companies whose behavior is very different from others (outliers) are ignored in order to make the dataset more compact. The complete pre-processing procedure is described as follows:

1. We excluded companies whose total assets were not in the range of 10^5 to 10^7 euros. There are 967 insolvent companies remain and 15,834 solvent companies remain.
2. In order to compute the accounting ratios AP/SALE, OI/TA, TL/TA, CASH/TA, IDINV/INV, INV/SALE, EBIT/TA and NI/SALE, we have removed companies with zero denominators (remaining insolvent: 816; solvent 11,005, after the pre-processing in previous step).
3. We dropped outliers. That is, the insolvent companies with extreme values of financial indices are removed (remaining insolvent: 811; solvent: 10,468).

After pre-processing, the dataset consists of 11,279 companies (811 insolvent and 10,468 solvent). In all the following analysis, we focus on the revised dataset.

6.2 The Procedure of Bankruptcy Prognosis with SVMs

We conduct the experiments in a scenario in which we train the SSVM bankruptcy prognosis model from the data at hand and then use the trained SSVM to predict the following year's cases. This strategy simulates the real task for analysts who may predict the future outcomes by using the data from past years. The experiment setting is described in Table 2. The number of periods used for the training set changes from one year (S1) to five years (S5) as time goes by. All classifiers we adopt in the experiments are reduced SSVM with Gaussian kernels. We need to determine two parameters, the best combination of C and γ for the kernels. In principle, the 2-D grid search will consume a lot of time. In order to cut down the search time, we adopt the nested uniformed design model selection method [26], introduced in Subsection 5.3 to search for a good pair of parameters for the performance of our classification task.

Table 1. The definition of accounting ratios used in the analysis

Variable	Ratio	Indicator for	Variable	Ratio	Indicator for
X1	NI/TA	Profitability	X15	CASH/TA	Liquidity
X2	NI/SALE	Profitability	X16	CASH/CL	Liquidity
X3	OI/TA	Profitability	X17	QA/CL	Liquidity
X4	OI/SALE	Profitability	X18	CA/CL	Liquidity
X5	EBIT/TA	Profitability	X19	WC/TA	Liquidity
X6	(EBIT+AD)/TA	Profitability	X20	CL/TL	Liquidity
X7	EBIT/SALE	Profitability	X21	TA/SALE	Activity
X8	EQUITY/TA	Leverage	X22	INV/SALE	Activity
X9	(EQUITY-ITGA)/ (TA-ITGA-CASH-LB)	Leverage	X23	AR/SALE	Activity
X10	CL/TA	Leverage	X24	AP/SALE	Activity
X11	(CL-CASH)/TA	Leverage	X25	Log(TA)	Size
X12	TL/TA	Leverage	X26	IDINV/INV	Growth
X13	DEBT/TA	Leverage	X27	IDL/TL	Growth
X14	EBIT/INTE	Leverage	X28	IDCASH/CASH	Growth

Table 2. The prediction scenario of our experiments

Scenario	Observation period of training set	Observation period of testing set
S1	1997	1998
S2	1997-1998	1999
S3	1997-1999	2000
S4	1997-2000	2001
S5	1997-2001	2002

Selection of Accounting Ratios via 1-norm SVM

In principle, many possible combination of accounting ratios could be used as explanatory variables in a bankruptcy prognosis model. Therefore, appropriate performance measures are needed to gear the process of selecting the ratios with the highest separating power. In [13] Accuracy Ratio (AR) and Conditional Information Entropy Ratio (CIER) determine the selection procedure's outcome. It turned out that the ratio "accounts payable divided by sales", X24 (AP/SALE), has the best performance values for a univariate SVM model. The second selected variable was the one combined with X24 that had the best performance of a bivariate SVM model. This is the analogue of forward selection in linear regression modeling. If one keeps on adding new variables one typically observes a declining change in improvement. This was also the case in that work where the performance indicators started to decrease after the model included eight variables. The described selection procedure is quiet lengthy, since there are at least 216 accounting ratio combinations to be considered. We will not employ the procedure here but use the chosen set of 8 variables in [13] denoted as V1. Table 3 presents V1 in the first column.

Table 3. Selected variables in V1 and V2 (the symbol "plus" means the common variables in V1 and V2)

Variable	Definition	V1	V2
$X2^+$	NI/SALE	x	x
$X3^+$	OI/TA	x	x
$X5^+$	EBIT/TA	x	x
X6	(EBIT+AD)/TA		x
X8	EQUITY/TA		x
X12	TL/TA	x	
$X15^+$	CASH/TA	x	x
X22	INV/SALE	x	
X23	AR/SALE		x
$X24^+$	AP/SALE	x	x
X26	IDINV/INV	x	

Except for using V1, we also apply 1-norm SVM which will simplify the selection procedure to select accounting ratios. The 1-norm SVM was applied to the period from 1997 to 1999. We selected the variables according to the size of the absolute values of the coefficients $\mathbf{w}$ from the solution of the 1-norm SVM. We also select 8 variables out of 28. Table 3 displays the 8 selected variables as V2. Note that five variables, X2, X3, X5, X15 and X24 are also in the benchmark set V1. From Table 4 and Table 5, we can the performances of V1 and V2 are quite similar while we need fewer efforts for extract V1.

Applying Over-sampling to Unbalanced Problems

The cleaned data set consists of around 10% of insolvent companies. Thus, the sample is fairly unbalanced although the share of insolvent companies is higher than in reality. In order to deal with this problem, insolvency prognosis models usually start off with more balanced training and testing samples than reality can provide. Here we use over-sampling and down-sampling [13] strategies, to balance the size between the solvent and the insolvent companies. In the experiments, the over-sampling scheme shows better results in the Type I error rate but has slightly bigger total error rates (see Table 4 and see Table 5). It is also obvious, that in almost all models a longer training period works in favor of accuracy of prediction. Clearly, the over-sampling schemes have much smaller standard deviations in the Type I error rate, the Type II error rate, and the total error rate than the down-sampling one. According to this observation, we conclude that the over-sampling scheme will generate a more robust model than the down-sampling scheme.

Table 4. The results in percentage (%) of over-sampling for three variable sets (Reduced SSVM with Gaussian kernel)

Set of accounting ratios	Scenario	Type I Error Rate		Type II Error Rate		Total Error Rate	
		mean	std	mean	std	mean	std
V1	S1	33.16	0.55	26.15	0.13	26.75	0.12
	S2	31.58	0.01	29.10	0.07	29.35	0.07
	S3	28.11	0.73	26.73	0.16	26.83	0.16
	S4	30.14	0.62	25.66	0.17	25.93	0.15
	S5	24.24	0.56	23.44	0.13	23.48	0.13
V2	S1	29.28	0.92	27.20	0.24	27.38	0.23
	S2	28.20	0.29	30.18	0.18	29.98	0.16
	S3	27.41	0.61	29.67	0.19	29.50	0.17
	S4	28.12	0.74	28.32	0.19	28.31	0.15
	S5	23.91	0.62	24.99	0.10	24.94	0.10

Applying the Reduced Kernel Technique for Fast Computation

Over-sampling duplicates the number of insolvent companies a certain number of times. In the experiments, we have to duplicate in each scenario the number of insolvent companies as many times as necessary to reach a balanced sample. Note that in our over-sampling scheme every solvent and insolvent companys information is utilized. This increases the computational burden due to increasing the number of training instances. We employ the reduced kernel technique in Section 3 to mediate this problem. Here the key idea for choosing the reduced set $\tilde{A}$ is extracting the same size of insolvent companies from solvent companies. This leads to not only the balance both in the data size and column basis bit but also the lower computational cost.

Summary

In analyzing `CreditReform` dataset for bankruptcy prognosis, we presented the usage of SVMs in a real case. The results show the selection of accounting ratios via 1-norm SVM can perform as well as the greedy search. The finance indices selected by 1-norm SVM actually can represent the data well in bankruptcy prognosis. The simple procedure of over-sampling strategy also helps to overcome the unbalanced problem while down-sampling will cause a biased model. In accelerating the training procedure, the reduced kernel technique is performed. It helps to build a SVM model in an efficient way without sacrificing the performance in prediction. Finally, the procedure of tuning parameters in a model is usually a heavy work in analyzing data. A good model selection method can help users to decease the long-winded tuning procedure, such as the 2-stage uniform design method used in this case study. In a nutshell, SVMs have been developed maturely. These practical usages presented here

Table 5. The results in percentage (%) of down-sampling for three variable sets (Reduced SSVM with Gaussian kernel)

Set of accounting ratios	Scenario	Type I Error Rate		Type II Error Rate		Total Error Rate	
		mean	std	mean	std	mean	std
V1	S1	32.20	3.12	28.98	1.70	29.26	1.46
	S2	29.74	2.29	28.77	1.97	28.87	1.57
	S3	30.46	1.88	26.23	1.33	26.54	1.17
	S4	31.55	1.52	23.89	0.97	24.37	0.87
	S5	28.81	1.53	23.09	0.73	23.34	0.69
V2	S1	29.94	2.91	28.07	2.15	28.23	1.79
	S2	28.77	2.58	29.80	1.89	29.70	1.52
	S3	29.88	1.88	27.19	1.32	27.39	1.19
	S4	29.06	1.68	26.26	1.00	26.43	0.86
	S5	26.92	1.94	25.30	1.17	25.37	1.06

not only show the variability and ability of SVMs but also give the basic ideas for analyzing data with SVMs.

7 Conclusion

The clear connection to statistic learning theory, efficient performance, and simple usage of SVMs have attracted many researchers to investigate. Many literatures have shown that SVMs are the state of the art in solving classification and regression problems. This reputation has made SVMs be applied in many fields, such as the quantitative finance field. This chapter presented many topics of SVMs as well as a case study in bankruptcy prognosis to give a guide for the usage of SVMs in quantitative finance filed. The possible applications with SVMs are various and potential in the quantitative finance field. The aim is giving that ones can quickly have solutions in their applications with SVMs while there are fertile materials in the wild.

References

1. Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001.
2. Edward Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, 23(4):589–609, September 1968.
3. Edward Altman, Giancarlo Marco, and Franco Varetto. Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the italian experience). *Journal of Banking and Finance*, 18:505–529, 1994.

4. William Beaver. Financial ratios as predictors of failures. *Journal of Accounting Research*, 4:71–111, 1966.
5. Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000.
6. Dimitri P. Bertsekas. *Nonlinear programming.* Athena Scientific Belmont, Mass, 1999.
7. Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discover*, 2(2):121–167, 1998.
8. Li-Juan Cao and Francis E.H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, 2003.
9. 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.
10. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
11. Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.
12. Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.
13. Shiyi Chen, Wolfgang Härdle, and Rouslan Moro. Estimation of default probabilities with support vector machines. *SFB 649 Discussion Paper 2006-077*, 2006.
14. Vladimir Cherkassky and Filip Mulier. *Learning from data: concepts, theory, and methods.* John Wiley & Sons, New York, 1998.
15. Richard Courant and David Hilbert. *Methods of Mathematical Physics.* Interscience Publishers, New York, 1953.
16. Koby Crammer and Yoram Singer. Improved output coding for classification using continuous relaxation. In *Proceeding of Advances in Neural Information Processing Systems 13*, 2001.
17. Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002.
18. 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.
19. Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
20. Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.
21. Michael C. Ferris and Todd S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal of Optimization*, 13:783–804, 2003.
22. Glenn Fung and Olvi. L. Mangasarian. Proximal support vector machine classifiers. In *Proceeding of Seventh ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2001.

23. Glenn Fung and Olvi. L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.
24. Tobias Glasmachers and Christian Igel. Maximum-gain working set selection for svms. *Journal of Machine Learning Research*, 7:1437–1466, 2006.
25. Wolfgang Härdle, Yuh-Jye Lee, Dorothea schäfer, and Yi-Ren Yeh. Variable selection and oversampling in the use of smooth support vector machines for predicting the default risk of companies. *Journal of Forecasting*, 28(6):512–534, 2009.
26. 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.
27. Ruey-Ching Hwang, K. F. Cheng, and Jack C. Lee. A semiparametric method for predicting bankruptcy. *Journal of Forecasting*, 26(5):317–342, 2007.
28. C.O. Inc. Using the cplex callable library and cplex mixed integer library. *Incline Village, NV*, 1992.
29. Thorsten Joachims. Making large-scale support vector machine learning practical. *Advances in Kernel Methods: Support Vector Learning*, pages 169 – 184, 1999.
30. S. Sathiya Keerthi and Elmer G. Gilbert. Convergence of a generalized smo algorithm for svm. *Machine Learning*, 46(1):351–360, 2002.
31. S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15(7):1667–1689, 2003.
32. Jan Pieter Krahnen and Martin Weber. Generally accepted rating principles: A primer. *Journal of Banking and Finance*, 25(1):3–23, January 2001.
33. 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.
34. Yuh-Jye Lee, Wen-Feng Hsieh, and Chien-Ming Huang. Ssvr: A smooth support vector machine for-insensitive regression. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):678–685., 2005.
35. Yuh-Jye Lee and Su-Yun Huang. Reduced support vector machines: A statistical theory. *IEEE Transactions on Neural Networks*, 18(1):1–13, 2007.
36. Yuh-Jye Lee and Olvi L. Mangasarian. Ssvm: A smooth support vector machine for classification. *Computational Optimization and Applications*, 20(1):5–22, 2001.
37. Olvi L. Mangasarian. *Nonlinear Programming.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1994.
38. Olvi L. Mangasarian. Generalized support vector machines. *Advances in Large Margin Classifiers*, pages 135–146, 2000.
39. Daniel Martin. Early warning of bank failure: A logit regression approach. *Journal of Banking and Finance*, 1:249–276, 1977.
40. 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.
41. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springe, 2006.
42. James Ohlson. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 18(1):109–131, Spring 1980.

43. Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. *Neural networks for signal processing VII*, pages 276–285, 1997.
44. John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, pages 185–208, 1999.
45. John Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems*, 12(3):547–553, 2000.
46. Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
47. Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.
48. 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.
49. Alexander J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.
50. Carl Staelin. Parameter selection for support vector machines. *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1*, 2003.
51. Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
52. Kar-Yan Tam and Melody Y. Kiang. Managerial application of neural networks: the case of bank failure prediction. *Management Science*, 38(7):926–947, 1992.
53. Robert Tibshiran. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.
54. Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory.* Springer, 2000.
55. Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 1999.
56. Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Proceeding of Advances in Neural Information Processing Systems*, pages 682–688, 2001.
57. Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machine. In *Proceeding of Advances in Neural Information Processing Systems 16*, 2004.