Chapter 3

SVM: Support Vector Machines

Hui Xue, Qiang Yang, and Songcan Chen

Contents

Support vector machines (SVMs), including support vector classifier (SVC) and support vector regressor (SVR), are among the most robust and accurate methods in all well-known data mining algorithms. SVMs, which were originally developed by Vapnik in the 1990s [1–11], have a sound theoretical foundation rooted in statistical learning theory, require only as few as a dozen examples for training, and are often insensitive to the number of dimensions. In the past decade, SVMs have been developed at a fast pace both in theory and practice.

3.1 Support Vector Classifier

For a two-class linearly separable learning task, the aim of SVC is to find a hyperplane that can separate two classes of given samples with a maximal margin which has been proved able to offer the best generalization ability. *Generalization ability* refers to the fact that a classifier not only has good classification performance (e.g., accuracy) on the training data, but also guarantees high predictive accuracy for the future data from the same distribution as the training data.

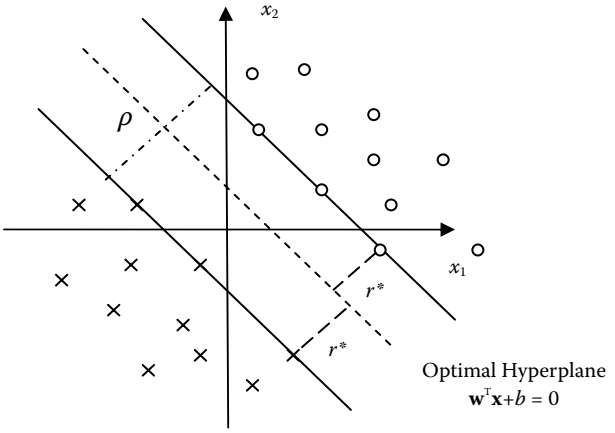

Figure 3.1 Illustration of the optimal hyperplane in SVC for a linearly separable case.

Intuitively, a *margin* can be defined as the amount of space, or separation, between the two classes as defined by a hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points to any point on the hyperplane. Figure 3.1 illustrates a geometric construction of the corresponding optimal hyperplane under the above conditions for a two-dimensional input space.

Let $\mathbf{w}$ and b denote the weight vector and bias in the optimal hyperplane, respectively. The corresponding hyperplane can be defined as

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{3.1}$$

The desired directionally geometrical distance from the sample $\mathbf{x}$ to the optimal hyperplane [12,13] is

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} \tag{3.2}$$

where $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ is the discriminant function [7] as defined by the hyperplane and also called $\mathbf{x}$'s functional margin given $\mathbf{w}$ and b.

Consequently, SVC aims to find the parameters $\mathbf{w}$ and b for an optimal hyperplane in order to maximize the margin of separation [ρ in Equation (3.5)] that is determined by the shortest geometrical distances r^* from the two classes, respectively, thus SVC is also called *maximal margin classifier*. Now without loss of generality, we fix the functional margin [7] to be equal to 1; that is, given a training set $\{\mathbf{x}_i, y_i\}_{i=1}^n \in \mathbf{R}^m \times \{\pm 1\}$, we have

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq 1 \quad &for \quad y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad &for \quad y_i = -1 \end{aligned} \tag{3.3}$$

The particular data points $(\mathbf{x}_i, y_i)$ for which the equalities of the first or second parts in Equation (3.3) are satisfied are called *support vectors*, which are exactly the closest data points to the optimal hyperplane [13]. Then, the corresponding geometrical distance from the support vector $\mathbf{x}^*$ to the optimal hyperplane is

$$r^* = \frac{g(\mathbf{x}^*)}{\|\mathbf{w}\|} = \begin{cases} \dfrac{1}{\|\mathbf{w}\|} & if \quad y^* = +1 \\ -\dfrac{1}{\|\mathbf{w}\|} & if \quad y^* = -1 \end{cases} \tag{3.4}$$

From Figure 3.1, clearly the margin of separation ρ is

$$\rho = 2r^* = \frac{2}{\|\mathbf{w}\|} \tag{3.5}$$

To ensure that the maximum margin hyperplane can be found, SVC attempts to maximize ρ with respect to $\mathbf{w}$ and b:

$$\begin{aligned} &\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|} \\ &s.t.\ y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, \quad i = 1, \ldots, n \end{aligned} \tag{3.6}$$

Equivalently,

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \\ &s.t.\ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n \end{aligned} \tag{3.7}$$

Here, we often use $\|\mathbf{w}\|^2$ instead of $\|\mathbf{w}\|$ for the convenience of carrying out the subsequent optimization steps.

Generally, we solve the constrained optimization problem in Equation (3.7), known as the *primal problem*, by using the method of Lagrange multipliers. We construct the following Lagrange function:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{n} \alpha_i \left[y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) - 1\right] \tag{3.8}$$

where α_i is the Lagrange multiplier with respect to the ith inequality.

Differentiating $L(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b, and setting the results equal to zero, we get the following two conditions of optimality:

$$\begin{cases} \dfrac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 \\ \dfrac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \end{cases} \tag{3.9}$$

Then we obtain

$$\begin{cases} \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{3.10}$$

Substituting Equation (3.10) into the Lagrange function Equation (3.8), we can get the corresponding *dual problem*:

$$\begin{aligned} \max_{\alpha} W(\alpha) = & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ & s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{3.11}$$

And at the same time, the Karush-Kuhn-Tucker complementary condition is

$$\alpha_i \left[y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 \right] = 0, \quad i = 1, \ldots, n \tag{3.12}$$

Consequently, only the support vectors $(\mathbf{x}_i, y_i)$ that are the closest data points to the optimal hyperplane and determine the maximal margin, correspond to the nonzero α_is. All the other α_is equal zero.

The *dual* problem in Equation (3.11) is a typical *convex quadratic programming optimization* problem. In many cases, it can efficiently converge to the global optimum by adopting some appropriate optimization techniques, such as the sequential minimal optimization (SMO) algorithm [7].

After determining the optimal Lagrange multipliers α_i^*, we can compute the optimal weight vector $\mathbf{w}^*$ by Equation (3.10):

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{3.13}$$

Then, taking advantage of a positive support vector $\mathbf{x}_s$, the corresponding optimal bias b^* can be written as [13]:

$$b^* = 1 - \mathbf{w}^{*T} \mathbf{x}_s \quad for \ y_s = +1 \tag{3.14}$$

3.2 SVC with Soft Margin and Optimization

Maximal margin SVC, including the following SVR, represents the original starting point of the SVM algorithms. However, in many real-world problems, it may be too rigid to require that all points are linearly separable, especially in many complex nonlinear classification cases. When the samples cannot be completely linearly separated, the margins may be negative. In these cases, the feasible region of the primal problem is empty, and thus the corresponding dual problem is an unbounded objective function. This makes it impossible to solve the optimization problem [7].

To solve these inseparable problems, we generally adopt two approaches. The first one is to relax the rigid inequalities in Equation (3.7) and thus lead to so-called soft margin optimization. Another method is to apply the kernel trick to linearize those nonlinear problems. In this section, we first introduce soft margin optimization. Consequently, relative to the soft margin SVC, we usually name SVC derived from the optimization problem [Equation (3.7)] the hard margin SVC.

Imagine the cases where there are a few points of the opposite classes mixed together in the data. These points represent the training error that exists even for the maximum margin hyperplane. The "soft margin" idea aims to extend the SVC algorithm so that the hyperplane allows a few of such noisy data to exist. In particular, a slack variable ξ_i is introduced to account for the amount of a violation of classification by the classifier:

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ s.t. \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{3.15}$$

where the parameter C controls the trade-off between complexity of the machine and the number of inseparable points. It may be viewed as a "regularization" parameter and selected by the user either experimentally or analytically.

The slack variable ξ_i has a direct geometric explanation through the distance from a misclassified data instance to the hyperplane. This distance measures the deviation of a sample from the ideal condition of pattern separability. Using the same method of Lagrange multipliers that are introduced in the above section, we can formulate the *dual* problem of the soft margin as:

$$\begin{aligned} \max_{\alpha} W(\alpha) &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ s.t. \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ 0 \leq \alpha_i &\leq C, \quad i = 1, \ldots, n \end{aligned} \tag{3.16}$$

Comparing Equation (3.11) with Equation (3.16), it is noteworthy that the slack variables ξ_is do not appear in the dual problem. The major difference between the linearly inseparable and separable cases is that the constraint $\alpha_i \geq 0$ is replaced with the more stringent constraint $0 \leq \alpha_i \leq C$. Otherwise, the two cases are similar, including the computations of the optimal values of the weight vector $\mathbf{w}$ and bias b, especially the definition of the support vectors [7,13].

The Karush-Kuhn-Tucker complementary condition in the inseparable case is

$$\alpha_i \left[y_i \left(\mathbf{w}^T \mathbf{x}_i + b \right) - 1 + \xi_i \right] = 0, \quad i = 1, \ldots, n \tag{3.17}$$

and

$$\gamma_i \xi_i = 0, \quad i = 1, \ldots, n \tag{3.18}$$

where γ_is are the Lagrange multipliers corresponding to ξ_i that have been introduced to enforce the nonnegativity of ξ_i [13]. At the saddle point at which the derivative of the Lagrange function for the primal problem with respect to ξ_i is zero, the evaluation of the derivative yields

$$\alpha_i + \gamma_i = C \tag{3.19}$$

Combining Equations (3.18) and (3.19), we have

$$\xi_i = 0 \quad if \quad \alpha_i < C \tag{3.20}$$

Consequently, we have the optimal weight $\mathbf{w}^*$ as follows:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{3.21}$$

The optimal bias b^* can be obtained by taking any data point $(\mathbf{x}_i, y_i)$ in the training set for which we have $0 < \alpha_i^* < C$ and the corresponding $\xi_i = 0$, and using the data point in Equation (3.17) [13].

3.3 Kernel Trick

The kernel trick is another commonly used technique to solve linearly inseparable problems. The issue is to define an appropriate kernel function based on the *inner product* between the given data, as a nonlinear transformation of data from the input space to a feature space with higher (even infinite) dimension in order to make the problems linearly separable. The underlying justification can be found in *Cover's theorem* on the separability of patterns; that is, a complex pattern classification problem cast in a high-dimensional space nonlinearly is *more likely* to be linearly separable than in a low-dimensional space [13].

Let $\mathbf{\Phi} : \mathbf{X} \rightarrow \mathbf{H}$ denote a nonlinear transformation from the input space $\mathbf{X} \subset \mathbf{R}^m$ to the feature space $\mathbf{H}$ where the problem can be linearly separable. We may define the corresponding optimal hyperplane as follows:

$$\mathbf{w}^{\mathbf{\Phi}^T}\mathbf{\Phi}(\mathbf{x}) + b = 0 \tag{3.22}$$

Without loss of generality, we set the bias $b = 0$, and simplify Equation (3.22) as:

$$\mathbf{w}^{\mathbf{\Phi}^T}\mathbf{\Phi}(\mathbf{x}) = 0 \tag{3.23}$$

Similar to the linear separable cases, we seek the optimal weight vector $\mathbf{w}^{\mathbf{\Phi}_*}$ in the feature space in virtue of the similar Lagrange multiplier method, and obtain:

$$\mathbf{w}^{\mathbf{\Phi}_*} = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{\Phi}(\mathbf{x}_i) \tag{3.24}$$

Thus, the optimal hyperplane computed in the feature space is:

$$\sum_{i=1}^{n} \alpha_i^* y_i \mathbf{\Phi}^T(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x}) = 0 \tag{3.25}$$

The term $\mathbf{\Phi}^T(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x})$ represents the inner product of two vectors, $\mathbf{\Phi}(\mathbf{x})$ and $\mathbf{\Phi}(\mathbf{x}_i)$. Hence, here we deduce the inner product kernel function:

Definition 3.3.1 (Inner Product Kernel) [7]. Kernel is a function $\mathbf{K}(\mathbf{x}, \mathbf{x}')$, for all $\mathbf{x}, \mathbf{x}' \in \mathbf{X} \subset \mathbf{R}^m$, satisfied:

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \mathbf{\Phi}^T(\mathbf{x})\mathbf{\Phi}(\mathbf{x}') \tag{3.26}$$

where $\mathbf{\Phi}$ is a transformation from the input space $\mathbf{X}$ to the feature space $\mathbf{H}$.

The significance of the kernel is that we may use it to construct the optimal hyperplane in the feature space *without having to consider the concrete form of the transformation* $\mathbf{\Phi}$, which usually need not be explicitly formulated in the higher dimension (even infinite) feature space. As a result, the application of the kernel can make the algorithm insensitive to the dimension, so as to train a linear classifier in a space with higher dimension to solve linearly inseparable problems efficiently. This is done by using $\mathbf{K}(\mathbf{x}_i, \mathbf{x})$ in Equation (3.25) to substitute $\mathbf{\Phi}^T(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x})$; then the optimal hyperplane is:

$$\sum_{i=1}^{n} \alpha_i^* y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) = 0 \tag{3.27}$$

As indicated, the kernel trick is an appealing method for simplifying the computation, by which we can avoid computing the complex feature space directly not only in the computation of the inner products but also in the design of the classifier.

However, before implementing the kernel trick, we should consider how to construct a kernel function, that is, a kernel function should satisfy which characteristics. To answer this question, we first introduce Mercer's theorem, which characterizes the property of a function $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ for when it is considered a true kernel function:

Theorem 3.3.2 ***Mercer's Theorem*** [13] *Let* $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ *be a continuous symmetric kernel that is defined in the closed interval* $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$ *and likewise for* $\mathbf{x}'$. *The kernel* $\mathbf{K}(\mathbf{x}, \mathbf{x}')$ *can be expanded in the series*

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(\mathbf{x}) \varphi_i(\mathbf{x}') \tag{3.28}$$

with positive coefficients, $\lambda_i > 0$ *for all* i. *For this expansion to be valid and for it to converge, it is necessary and sufficient that the condition*

$$\int_{\mathbf{b}}^{\mathbf{a}} \int_{\mathbf{b}}^{\mathbf{a}} \mathbf{K}(\mathbf{x}, \mathbf{x}') \psi(\mathbf{x}) \psi(\mathbf{x}') \, d\mathbf{x} \, d\mathbf{x}' \geq 0 \tag{3.29}$$

holds for all $\psi(\cdot)$ *for which*

$$\int_{\mathbf{b}}^{\mathbf{a}} \psi^2(\mathbf{x}) \, d\mathbf{x} < \infty \tag{3.30}$$

In light of the theorem, we can summarize the most useful characteristic in the construction of the kernel, which is termed Mercer kernel. That is, for any random limited subsets belonging to the input space $\mathbf{X}$, the corresponding matrix constructed by the kernel function $\mathbf{K}(\mathbf{x}, \mathbf{x}')$

$$\mathbf{K} = \left(\mathbf{K}\left(\mathbf{x}_i, \mathbf{x}_j'\right)\right)_{i,j=1}^{n} \tag{3.31}$$

is a symmetric and semidefinite matrix, which is called a Gram matrix [7].

Under this requirement, there is still some freedom in how to choose a kernel function in practice. For example, besides linear kernel functions, we can also define polynomial or radial basis kernel functions. More studies in recent years have gone into the research of different kernels for SVC classification and for many other statistical tests. We will mention these in the following section.

In Section 3.2, we introduced the soft margin SVC to solve linearly inseparable problems. Compared with the kernel trick, it is obvious that the two approaches actually solve the problems in different manners. The soft margin slackens the constraints in the original input space and allows some errors to exist. However, when the problem is heavily linearly inseparable and the misclassified error is too high, the soft margin is unworkable. The kernel trick maps the data to a high-dimension feature space implicitly by the kernel function in order to make the inseparable problems separable. However, in fact the kernel trick cannot always guarantee the problems to be absolutely linearly separable due to the complexity of the problems. Therefore,

in practice we often integrate them to exert the different advantages of the two techniques and solve the linearly inseparable problems more efficiently. As a result, the corresponding dual form for the constrained optimization problem in the kernel soft margin SVC is as follows:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$$

$$s.t. \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \tag{3.32}$$

Following the similar Lagrange multipliers method, we can obtain the optimal classifier:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}) + b^* \tag{3.33}$$

where $b^* = 1 - \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{K}(\mathbf{x}_i, \mathbf{x}_s)$, for a positive support vector $y_s = +1$.

Example 3.3.3 **(Illustrative Example)** The XOR problem is a typical extremely linearly inseparabe problem in classification. Here we use it to illustrate the significance of the soft margin SVC combined with kernel trick in the complex classification problems. A two-dimensional XOR dataset can be randomly generated under four different Gaussian distributions, where "*" and "•" denote the samples in the two classes, respectively.

As shown in Figure 3.2*a*, the hard margin SVC in the linear kernel completely fails in the XOR problem. A linear boundary cannot discriminate the two classes and can be seen to divide all the samples into two parts. This clearly cannot achieve the classification objective for the problem. Consequently, we use the soft margin SVC combined with a radial basis kernel to solve the problem

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{\sigma^2}\right)$$

We fix the regularization parameter $C = 1$ and the kernel parameter or bandwidth $\sigma = 1$. The corresponding discriminant boundary is presented in Figure 3.2*b*. By using the kernel trick, the boundary is no longer linear, for it now encloses only one class. By judging the samples inside or outside the boundary, the classifier can be seen to classify the samples accurately.

Example 3.3.4 **Real Application Example** SVC algorithm has been widely applied in many important scientific fields, such as bioinformatics, physics, chemistry, iatrology, astronomy, and so on. Here we carefully select five datasets in the iatrology area from the UCI Machine Learning Repository (http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm) to illustrate real applications of SVC. The five

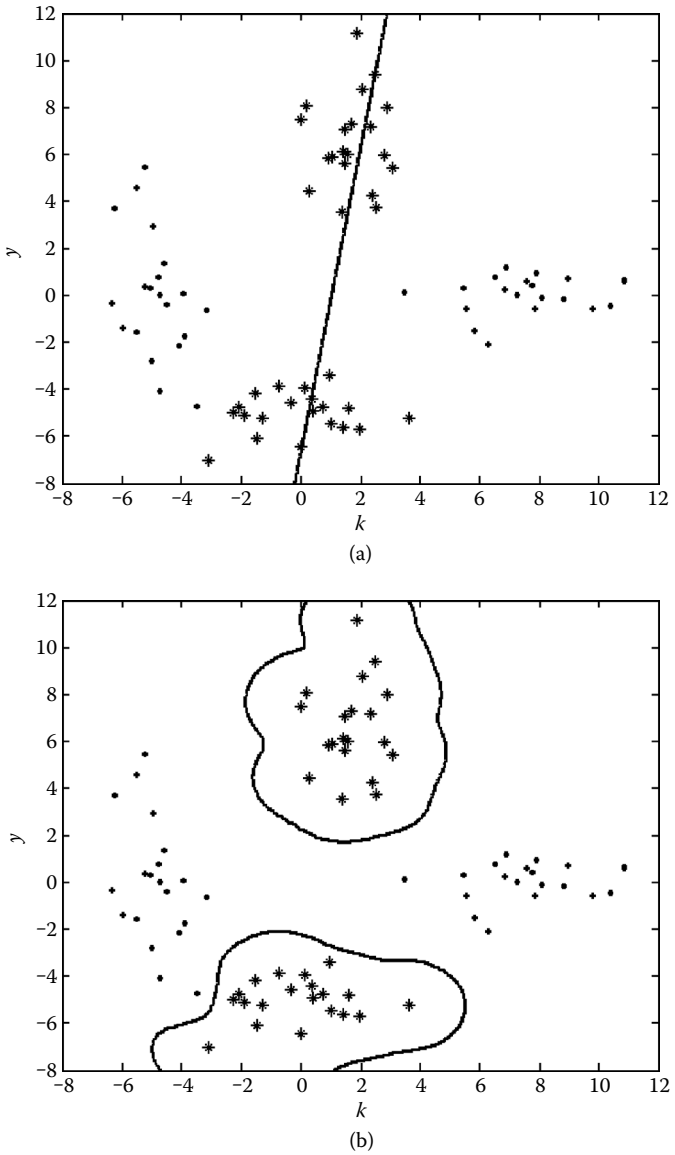

Figure 3.2 The discriminant boundaries of SVC in the XOR problem. (a) The hard margin SVC in the linear kernel. (b) The soft margin SVC in the radial basis kernel.

TABLE 3.1 Results of the SVC Algorithm for the Five Datasets

Dataset	Dimension	Training	Testing	C	σ	SV	Accuracy
B.-cancer	9	200	77	1.519e+01	5.000e+01	138.80	0.7396±4.74
Diabetes	8	468	300	1.500e+01	2.000e+01	308.60	0.7647±1.73
Heart	13	170	100	3.162e+00	1.200e+02	86.00	0.8405±3.26
Thyroid	5	140	75	1.000e+01	3.000e+00	45.80	0.9520±2.19
Splice	60	1000	2175	1.000e+03	7.000e+01	762.40	0.8912±0.66

datasets, respectively, are B.-cancer (breast cancer Wisconsin data), diabetes (Pima Indians diabetes data), heart (heart data), thyroid (thyroid disease data), and splice (splice-junction gene sequences data).

The two to four columns of Table 3.1 summarize some characteristics about the datasets, where Dimension denotes the dimension of the samples, and Training and Testing denote the numbers of the training and testing samples in each dataset. We perform independently repeated 100 runs and 20 runs, respectively, for the first four datasets and splice dataset, which have been offered by the database. Then the average experimental results of the SVC algorithm have been reported in the five to eight columns of Table 3.1. C and σ are the optimal regularization and kernel parameters selected by the cross-validation. SV is the average number of support vectors. Accuracy denotes the corresponding classification accuracies and variances.

As shown in Table 3.1, the values of SV are typically less than the numbers of training samples, which validates the good sparsity of the algorithm. Furthermore, the high accuracies show the good classification performance; meanwhile, the relatively low variances show the good stability of SVC in the real applications.

3.4 Theoretical Foundations

In the above sections, we have described the SVC algorithm both in the linearly separable and inseparable cases. The introduction of the kernel trick further improves the expression performance of the classifier, which can keep the inherent linear property in a high-dimensional feature space and avoid the possible *curse of dimension*. In this section, we will discuss the theoretical foundation of the SVC. By the Vapnik-Chervonenkis (VC) theory [4,5], we will first present a general error bound of a linear classifier which can guide globally how to control the classifier complexity. We will then deduce a concrete generalization bound of the SVC to explain the significance of the maximum margin in the SVC to guarantee the good generalization capacity of the algorithm.

The VC theory generalizes the probably approximately correct (PAC) learning model in statistical learning and directly leads to the proposal of the SVMs. It provides

an analytical generalization bound that can be used for estimating generalization error by defining a new measure of complexity, known as the *VC dimension* [14,15].

Concretely, assume that training and testing data are generated according to a fixed but unknown probability distribution $\mathbf{D}$, we define the error $err_{\mathbf{D}}(h)$ of a classification function h on the $\mathbf{D}$ as

$$err_{\mathbf{D}}(h) = \mathbf{D}\{(\mathbf{x}, y) : h(\mathbf{x}) \neq y\} \tag{3.34}$$

which measures the expected error [7].

PAC models bound the distribution of the generalization error random variable $err_{\mathbf{D}}(h_s)$ and the corresponding PAC bound has the form $\varepsilon = \varepsilon(n, H, \delta)$; that is, a PAC model considers that in the hypothesis h_s, the probability of the error in the training data S satisfies [7]:

$$\mathbf{D}^n\{S : err_{\mathbf{D}}(h_s) > \varepsilon(n, H, \delta)\} < \delta \tag{3.35}$$

If there are $|H|$ hypotheses having large errors in the set S, then the PAC bound is

$$\varepsilon = \varepsilon(n, H, \delta) = \frac{1}{n} \ln \frac{|H|}{\delta} \tag{3.36}$$

PAC bound presents that the function class H can directly influence the error bound. VC theory further generalizes the PAC bound to the unlimited function class and introduces the concept of the VC dimension d. The VC dimension d measures the maximum number of training data where the function class can still be used to learn perfectly, by obtaining zero error rates on the training data, for any assignment of class labels to these points. Then the generalized PAC bound of a linear classifier can be described as follows:

Theorem 3.4.1 ***Vapnik and Chervonenkis*** [7] *Let H denote a hypothesis space whose VC dimension is d. For random probability distribution $\mathbf{D}$ on $\mathbf{X} \times \{-1, 1\}$, with probability $1 - \delta$, the generalization error of random hypothesis $h \in H$ on the training set S is no more than*

$$err_{\mathbf{D}}(h) \leq \varepsilon(n, H, \delta) = \frac{2}{n}\left(\log \frac{2}{\delta} + d \log \frac{2en}{d}\right) \tag{3.37}$$

under the condition that $d \leq n$, $n > 2/\varepsilon$.

In light of the theorem, the first term of Equation (3.37) is the training error, and the second term is proportional to the VC dimension d. Thus, the theorem shows that if we can minimize d, we can minimize the future error, as long as the hypothesis h controls the empirical risk error in a small degree.

Theorem 3.4.1 provides a general error bound of a linear classifier and gives the global guidance on how to control the classifier complexity. In the following, we will generalize the bound for the SVC algorithm and deduce the corresponding generalization error bound of SVC.

We first give a formal definition of the margin:

Definition 3.4.2 (Margin) [7]. Consider using a real value function class F to classify in the input space $\mathbf{X}$, and the threshold value is 0. We define the margin of the example $(\mathbf{x}_i, y_i) \in \mathbf{X} \times \{-1, 1\}$ to the function or hyperplane $f \in F$ as:

$$\gamma_i = y_i f(\mathbf{x}_i) \tag{3.38}$$

Note that $\gamma_i > 0$ denotes that the example $(\mathbf{x}_i, y_i)$ is correctly classified. The marginal distribution of f corresponding to the training set S is the marginal distribution of the examples in S. The minimum of the marginal distribution is called the *margin* $m_S(f)$ of f corresponding to the training set S.

Although the VC dimension d is theoretically meaningful, in practice d is sometimes infinite and thus the generalization bound is inapplicable to many real problems. Consequently, we introduce a similar measure related to the margin in SVC instead of the traditional VC dimension:

Definition 3.4.3 (Cover of Function Class) [7]. Let F be a real value function class in $\mathbf{X}$. For a series of input data

$$S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

The γ-cover of F is the limited function set B, such that for all $f \in F$, existing $g \in B$, there is $\max_{1 \le i \le n}(|f(\mathbf{x}_i) - g(\mathbf{x}_i)|) < \gamma$. $N(F, S, \gamma)$ denotes the minimal size of the cover. The number of data that F covers is

$$N(F, n, \gamma) = \max_{S \in X^n} N(F, S, \gamma) \tag{3.39}$$

Then we use $N(F, n, \gamma)$ to reformulate Theorem 3.4.1 for the case that the hypothesis f is such that $m_S(f) = \gamma$ on the training set S.

Theorem 3.4.4 ***VC Theorem with Margin*** [7] *Consider a bounded real value function space F and fix $\gamma \in \mathbf{R}^+$. For any probability distribution $\mathbf{D}$ on $\mathbf{X} \times \{-1, 1\}$, with probability $1 - \delta$, the generalization error of a hypothesis $f \in F$ on the training set S, which has a margin $m_S(f) \ge \gamma$, satisfies*

$$err_{\mathbf{D}}(f) \le \varepsilon(n, F, \delta, \gamma) = \frac{2}{n}\left(\log\frac{2}{\delta} + \log N(F, 2n, \gamma/2)\right) \tag{3.40}$$

under the condition that $n > 2/\varepsilon$.

Theorem 3.4.4 shows how to use $m_S(f)$ to bound the generalization error which can be obtained by the training data. $N(F, 2n, \gamma/2)$ may be viewed as another form of the VC dimension, where a larger γ corresponds to a smaller $N(F, 2n, \gamma/2)$. As a result, we may draw a conclusion that large margin can ensure good generalization performance of the classifier for small size samples.

Although Theorem 3.4.4 is a generalization of Theorem 3.4.1, the value $N(F, 2n, \gamma/2)$ cannot be efficiently quantified in the real-world problems. Consequently, we further deduce a more concrete error bound for the specific SVC algorithm:

Theorem 3.4.5 ***Generalization Bound of SVC*** [7] *Assume that the input space* $\mathbf{X}$ *is a hyperball in the inner product space* $\mathbf{H}$ *whose radius is* R, $\mathbf{X} = \{\mathbf{x} \in \mathbf{H} : \|\mathbf{x}\|_{\mathbf{H}} \leq R\}$. *Consider the function class* Ψ*:*

$$\Psi = \left\{\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} : \|\mathbf{w}\|_{\mathbf{H}} \leq 1, \mathbf{x} \in \mathbf{X}\right\}$$

Fix $\gamma \in \mathbf{R}^+$. *For a probability distribution* $\mathbf{D}$ *on* $\mathbf{X} \times \{-1, 1\}$, *with probability* $1 - \delta$, *the generalization error of a hypothesis* $f \in \Psi$ *on the training set* S, *which has the margin* $m_S(f) \geq \gamma$, *is no more than*

$$err_{\mathbf{D}}(f) \leq \varepsilon(n, \Psi, \delta, \gamma) = \frac{2}{n}\left(\log\frac{4}{\delta} + \frac{64R^2}{\gamma^2}\log\frac{en\gamma}{4R}\log\frac{128nR^2}{\gamma^2}\right) \tag{3.41}$$

under the condition that $n > 2/\varepsilon$, $64R^2/\gamma^2 < n$.

It is noteworthy that the dimension of the input space does not appear in the bound. Hence the bound can be used in the infinite dimension space, which denotes that the bound may overcome the curse of dimension. Furthermore, when the samples distribute well, the bound may guarantee in a high probability that there is a small error for random testing samples. In that case, the margin γ can be viewed as a measure about the quality of the sample distribution, and thus may further measure the generalization performance of the SVC algorithm [7].

3.5 Support Vector Regressor

Up to this point, we have focused on the SVC method for classification tasks. In this section, we will consider using SVM to solve nonlinear *regression* problems, thus called SVR. Similar to the classification algorithm, we also expect to explore the main characteristics of the maximum margin method by exploiting nonlinear functions, which can be obtained using linear learning methods and the kernel trick. In addition, the corresponding algorithms must be efficient under high dimensions [7].

However, for regression problems, the traditional least-squares estimator may not be quite feasible in the presence of outliers, resulting in the regressor to perform poorly when the underlying distribution of the additive noise has a long tail [13]. Thus we need to develop a robust estimator insensitive to small changes in the model; that is, we seek a so-called ε-insensitive loss function.

Definition 3.5.1 (ε**-Insensitive Loss Function**) [7]
Let f be a real valued function in $\mathbf{X}$. The ε-insensitive loss function $L^{\varepsilon}(\mathbf{x}, y, f)$ is defined as:

$$L^{\varepsilon}(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_{\varepsilon} = \max(0, |y - f(\mathbf{x})| - \varepsilon) \tag{3.42}$$

Note that $L^{\varepsilon}(\mathbf{x}, y, f) = 0$ if the absolute value of the deviation about the estimator output $f(\mathbf{x})$ from the desired response y is less than ε or equal to zero. It is equal to the absolute value of the deviation minus ε otherwise.

Now consider a nonlinear regression model

$$y = g(\mathbf{x}) + v \tag{3.43}$$

where the additive noise term v is statistically independent of the input vector $\mathbf{x}$. The function $g(\cdot)$ and the statistics of noise v are unknown. All that we have available is a set of training data

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

and a function class

$$F = \{f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \mathbf{w} \in \mathbf{R}^m, b \in \mathbf{R}\}$$

The objective is to select appropriate parameters $\mathbf{w}$ and b, so as to make $f(\mathbf{x})$ approximate the unknown target function $g(\mathbf{x})$. The primal problem can be represented as follows:

$$\begin{aligned} \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \; + \; C\sum_{i=1}^{n}(\xi_i + \hat{\xi}_i) \\ s.t. \left(\mathbf{w}^T\mathbf{x}_i + b\right) - y_i \le \varepsilon + \xi_i, \quad i = 1, \ldots, n \\ y_i - \left(\mathbf{w}^T\mathbf{x}_i + b\right) \le \varepsilon + \hat{\xi}_i, \quad i = 1, \ldots, n \\ \xi_i, \hat{\xi}_i \ge 0 \quad i = 1, \ldots, n \end{aligned} \tag{3.44}$$

Using the similar method of Lagrange multipliers, the dual problem is:

$$\begin{aligned} \max_{\alpha,\hat{\alpha}} W(\alpha, \hat{\alpha}) = \sum_{i=1}^{n} y_i(\hat{\alpha}_i - \alpha_i) \; - \; \varepsilon\sum_{i=1}^{n}(\hat{\alpha}_i + \alpha_i) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\mathbf{x}_i^T\mathbf{x}_j \\ s.t. \quad \sum_{i=1}^{n}(\hat{\alpha}_i - \alpha_i) = 0 \\ 0 \le \alpha_i, \hat{\alpha}_i \le C, \quad i = 1, \ldots, n \end{aligned} \tag{3.45}$$

We can further introduce the inner product kernel in the optimization problem Equation (3.45), and extend the regression algorithm to a feature space so as to make the nonlinear functions able to be obtained by means of the linear learning machines in the kernel space.

Compared with SVC, SVR has an additional free parameter ε. The two free parameters ε and C control the VC dimension of the approximating function

$$f(x) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{n}(\hat{\alpha}_i - \alpha_i)\mathbf{K}(\mathbf{x}_i, \mathbf{x}) \tag{3.46}$$

when we set the bias $b = 0$. ε and C should be selected by the user and directly influence the complexity control for regression. How to select ε and C simultaneously to get a better approximation function is an open research problem.

3.6 Software Implementations

LibSVM [16] and SVMlight [17] are two of the most famous software about the implementation of SVM algorithms.

LibSVM provides not only compiler languages used in the Windows system, but also C++ and Java source codes which are easy to improve, revise, and apply in other operating systems. Specially, LibSVM has relatively fewer tunable parameters involved in SVM algorithms than other software and provides lots of default parameters to solve real application problems effectively.

SVMlight is another implementation in C language. It adopts an efficient set selection technique based on steepest feasible descent, and two effective computational policies "Shrinking" and "Caching" of kernel evaluations. SVMlight mainly includes two C programs: SVM_learn, used for learning training samples and training the corresponding classifier, and SVM_classifiy, used for classifying testing samples. The software also provides two efficient estimation methods for assessing the generalization performance: XiAlpha-estimates, computed at essentially no computational expense but conservatively biased, and Leave-one-out testing, almost unbiased.

Furthermore, there are lots of complete machine learning toolboxes, including SVM algorithms, such as Torch (in C++), Spider (in MATLAB), and Weka (in Java), which are all available at http://www.kernel-machines.org.

3.7 Current and Future Research

In the past decade, SVMs have been developed at a fast pace both in theory and in practice. Many future works remain. In this section, we enumerate a few of the major research directions where major progress is being made and many research problems are still open.

3.7.1 Computational Efficiency

One of the initial drawbacks of the SVMs is its costly computational complexity in the training phase, which leads to inapplicable algorithms in the large datasets. However, this problem is being solved with great success. One approach is to break a large optimization problem into a series of smaller problems, where each problem only involves a couple of carefully chosen variables so that the optimization can be done efficiently. The process iterates until all the decomposed optimization problems are solved successfully.

A more recent approach is to consider the problem of learning SVMs as that of finding an approximate minimum enclosing ball of a set of instances [18–21]. These instances, when mapped to an N-dimensional space, represent a core set that can be used to construct an approximation to the minimum enclosing ball. Solving the SVMs'

learning problems on these core sets can produce good approximation solutions in very fast speed. For example, the core vector machine [18] and the further ball vector machine [21] can learn SVMs for millions of data in seconds.

3.7.2 Kernel Selection

In the kernel SVMs, the selection of the kernel function is generally required to satisfy the Mercer's theorem. Hence, the common kernel functions involve three types, that is, sigmoid, polynomial, and radial basis functions, which may sometimes limit the applicability of the kernel trick. Recently, Pekalska et al. provided a novel view to design a kernel function based on a general proximity relation mapping [22]. The new kernel function needs neither be satisfied by the Mercer's conditions nor be limited to only one feature space, and shows better classification performance than the common Mercer kernels experimentally. However, the theoretical foundation of the new generalized kernel needs further research.

Furthermore, another popular approach is multiple kernel learning which considers more than one kernel; through the combinations one can achieve better results [23–29]. This is similar to using an ensemble of kernels. By setting the proper objective functions, better selection of the kernel parameters can be done to allow mixture kernels.

3.7.3 Generalization Analysis

We are accustomed to using the VC dimension to estimate the generalization error bound of the kernel machines. However, the bound involves a fixed complexity penalty which does not depend on the training data, which as a result, cannot be made universally effective [30]. To solve this problem, Rademacher's complexity is introduced as an alternative to evaluate the complexity of a classifier instead of the classical VC dimension [31–34], which is based on the intuition that we can measure the capacity (or complexity) of a classifier by its ability to fit random data. It is defined as follows:

Definition 3.7.1 (Rademacher Complexity) [35]. For the sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ generated by a distribution D on a set $\mathbf{X}$ and a real value function class F with domain $\mathbf{X}$, the empirical Rademacher complexity of F is the random variable

$$\hat{R}_n(\mathbf{F}) = E_{\boldsymbol{\sigma}}\left[\sup_{f\in F}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f(\mathbf{x}_i)\right| \middle\| \mathbf{x}_1, \ldots, \mathbf{x}_n\right] \tag{3.47}$$

where $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_n\}$ are independent uniform $\{\pm 1\}$-valued (Rademacher) random variables. The Rademacher complexity of F is

$$R_n(\mathbf{F}) = E_S[\hat{R}_n(\mathbf{F})] = E_{S\boldsymbol{\sigma}}\left[\sup_{f\in F}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i f(\mathbf{x}_i)\right|\right] \tag{3.48}$$

The sup part inside the expectation formula measures the best correlation that can be found between a function of the class and the random labels. Furthermore, in the kernel machines, we can obtain an upper bound to the Rademacher complexity:

Theorem 3.7.2 ***Complexity Analysis*** [35]. *If* $k : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{R}$ *is a kernel, and* $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ *is a sample of points from* $\mathbf{X}$*, then the empirical Rademacher complexity of the classifier* F_B *satisfies*

$$\hat{R}_n(\mathbf{F}_B) \leq \frac{2B}{n}\sqrt{\sum_{i=1}^{n} k(\mathbf{x}_i, \mathbf{x}_i)} = \frac{2B}{n}\sqrt{tr(\mathbf{K})} \tag{3.49}$$

where B *is the bound of the weights* $\mathbf{w}$ *in the classifier.*

It is noteworthy that the bound of the Rademacher complexity only involves the trace of the corresponding kernel matrix, which is determined by the concrete training data. It is more feasible to use than the traditional VC dimension to control the complexity of a classifier as well as estimate the generalization performance.

3.7.4 Structural SVM Learning

Margin maximization is the initial motivation of the SVM algorithms [36]. Consequently, SVM (SVC) usually places more focus on the separability between the classes of samples but does not sufficiently use the prior data distribution information within classes. The well-known "No Free Lunch" theorem [12] indicates that there does not exist a pattern classification method that is inherently superior to any other, or even to random guessing without using additional information. It is the type of problem, prior information, and the amount of training samples that determine the form of classifier to apply. In fact, corresponding to different real-world problems, different classes may have different underlying data structures. A classifier should adjust the discriminant boundaries to fit the structures which are vital for classification, especially for the generalization capacity of the classifier. However, the traditional SVM does not differentiate the structures, and the derived decision hyperplane lies unbiasedly right in the middle of the support vectors [36,37], which may lead to a nonoptimal classifier in the real-world problems.

Recently, some algorithms have been developed to give more concern to the structural information than the traditional SVM. They provide a novel view to design a classifier, where the classifier can be sensitive to the structure of the data distribution. These algorithms are mainly divided into two approaches. The first approach is through manifold learning. It assumes that the data actually live on a submanifold in the input space, and the most typical algorithm involves Laplacian support vector machines (LapSVM) [38,39]. We can construct LapSVM first through a Laplacian graph in each class. Then we introduce a manifold structure of the data within the corresponding Laplacian matrices into the traditional framework of SVM as an additional term.

The second approach is by exploiting clustering algorithms [40] by assuming that the data contain several clusters that hold the prior distribution information. This assumption seems more general than the manifold assumption, which has in fact led to several popular large margin machines. A recent approach is known as *structured large margin machine* (SLMM) [37]. SLMM applies clustering techniques to capture the structural information in the different classes first. It then uses the Mahalanobis distance as a distance measure from the samples to the decision hyperplanes, instead of the traditional Euclidean distance, to introduce the involved structure information into the constraints. Some popular large margin machines, such as support vector machine minimax probability machine (MPM) [41], and maxi-min margin machine (M^4) [36], can all be viewed as the special cases of SLMM. Experimentally, SLMM has shown better classification performance. However, since the optimization problem of SLMM is formulated as sequential second order cone programming (SOCP) rather than the QP in SVM, SLMM has much higher computational cost in training time as compared to traditional SVM. Furthermore, it is not easy to be generalized to large-scale or multiclass problems. Consequently, a novel structural support vector machine (SSVM) was developed in [42] to exploit the classical framework of SVM rather than as constraints in SLMM. As a result, the corresponding optimization problem can still be solved by the QP as in SVM, and keep the solution not only sparsity but also scalability. Furthermore, SSVM has been shown to be theoretically and empirically better in generalization than SVM and SLMM.

3.8 Exercises

1. Consider a simple binary classification problem:

$$c_1 : (1, 1)^T \quad (-1, 3)^T \quad (2, 6)^T$$
$$c_2 : (-1, -2)^T \quad (1, -3)^T \quad (-5, -7)^T$$

   (a) Compute the optimal hyperplane and geometrical margin.
   (b) Point out the support vectors.
   (c) Using the method of Lagrange multipliers, compute the solution in the dual space.

2. Consider another binary classification problem:

$$c_1 : (1, 1)^T \quad (3, 7)^T \quad (5, 9)^T$$
$$c_2 : (-1, -2)^T \quad (1, 6)^T \quad (2, -1)^T$$

   Use a soft margin SVC to construct the optimal hyperplane and compute the corresponding solution in the dual space.

3. Construct a simple XOR problem similar to Example 3.3.3, and discuss how the selection of the kernel parameter in the radial basis kernel can influence the classification performance.

4. Let $\mathbf{K}_1$ and $\mathbf{K}_2$ be the kernels in $\mathbf{X} \times \mathbf{X}, \mathbf{X} \subseteq \mathbf{R}^n$, $a \in \mathbf{R}^+$, $f(\cdot)$ be a real value function in $\mathbf{X}$:

$$\phi : \mathbf{X} \to \mathbf{R}^m$$

where $\mathbf{K}_3$ is a kernel in $\mathbf{R}^m \times \mathbf{R}^m$, and $\mathbf{B}$ is an $n \times n$ symmetrical semidefinite matrix. Prove the following functions are kernel functions:

(a) $\mathbf{K}(\mathbf{x}, \mathbf{z}) = \mathbf{K_1}(\mathbf{x}, \mathbf{z}) + \mathbf{K_2}(\mathbf{x}, \mathbf{z})$
(b) $\mathbf{K}(\mathbf{x}, \mathbf{z}) = \mathbf{aK_1}(\mathbf{x}, \mathbf{z})$
(c) $\mathbf{K_1}(\mathbf{x}, \mathbf{z})\mathbf{K_2}(\mathbf{x}, \mathbf{z})$
(d) $\mathbf{K}(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$
(e) $\mathbf{K}(\mathbf{x}, \mathbf{z}) = \mathbf{K_3}(\phi(\mathbf{x}), \phi(\mathbf{z}))$
(f) $\mathbf{K}(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{Bz}$

5. Discuss the generalization bounds of SVR derived from the VC theorem.
6. We have discussed the use of SVC for binary classification problems. Discuss how to extend SVC to solve multiclass classification problems.
7. Discuss the robustness properties of SVM algorithms.
8. Discuss the cases that SVC does not sufficiently use the prior data distribution information within classes, where the resulting discriminant hyperplane lies right in the middle of the support vectors.

References

[1] V. Vapnik. *The Nature of Statistical Learning Theory*, Springer Verlag, 1995.

[2] V. Vapnik. *Statistical Learning Theory*, Wiley, 1998.

[3] B. Schölkopf, C.J.C. Burges, and A.J. Smola. *Advances in Kernel Methods—Support Vector Learning*, MIT Press, 1999.

[4] O. Chapelle, P. Haffner, and V. Vapnik. Support vector machines for histogram-based image classification. *IEEE Trans. on Neural Networks*, vol. 10(3.5), 1055–1064, 1999.

[5] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, vol. 20, 273–297, 1995.

[6] N. Cristianini, C. Campbell, and J. Shawe-Taylor. A multiplicative updating algorithm for training support vector machine. In *Proceedings of the 6th European Symposium on Artificial Neural Networks (ESANN)*, 1999.

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, 2000.

[8] M.S. Kearns, S.A. Solla, and D.A. Cohn. *Advances in Neural Information Processing Systems*, MIT Press, 1999.

[9] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smythand, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.

[10] A.J. Smola, P. Bartlett, B. Schölkopf, and C. Schuurmans. *Advances in Large Margin Classifiers*, MIT Press, 1999.

[11] B. Schölkopf. *Support Vector Learning*, R. Oldenbourg Verlag, 1997.

[12] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*, Wiley, 2001.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation*, Tsinghua University Press, 2001.

[14] V. Cherkassky, X. Shao, F. Mulier, and V. Vapnik. Model complexity control for regression using VC generalization bounds. *IEEE Transactions on Neural Networks*, vol. 10, 1075–1089, 1999.

[15] V. Cherkassky and F. Mulier. *Learning From Data: Concepts, Theory and Methods,* Wiley, 1998.

[16] C.-C. Chang and C.-J. Lin. LibSVM: A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[17] T. Joachims. Making Large-scale SVM learning practical. *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (eds.), MIT Press, 1999.

[18] I. W. Tsang, J.T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, vol. 6, 363–392, 2005.

[19] I.W. Tsang, J.T. Kwok, and K.T. Lai. Core vector regression for very large regression problems. *ICML*, 913–920, 2005.

[20] I.W. Tsang and J.T. Kwok. Large-scale sparsified manifold regularization. *NIPS*, Vancouver, Canada, 2006.

[21] I.W. Tsang, A. Kocsor, and J.T. Kwok. Simpler core vector machines with enclosing balls. *ICML*, 2007.

[22] E. Pekalska, P. Paclik, and R.P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, vol. 2, 175–211, 2001.

[23] J. Bi, T. Zhang, and K. Bennett. Column-generation boosting methods for mixture of kernels. *KDD*, 521–526, 2004.

[24] I.M. de Diego, J.M. Moguerza, and A. Munoz. Combining kernel information for support vector classification. Multiple Classifier Systems, 102–111, 2004.

[25] Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. *Neural Information Processing Systems*, 2002.

[26] G.R.G. Lanckriet, T.D. Bie, N. Cristianini, M.I. Jordan, and W.S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, vol. 20(3.16), 2626–2635, 2004.

[27] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, vol. 5, 27–72, 2004.

[28] C.S. Ong, A.J. Smola, and R.C. Williamson. Learning the kernel with hyperkernels. *JMLR*, vol. 6, 1043–1071, 2005.

[29] Z. Wang, S. Chen, and T. Sun. MultiK-MHKS: A novel multiple kernel learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30(3.2), 348–353, 2008.

[30] P.L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, vol. 3, 463–482, 2002.

[31] P.L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, vol. 44(3.2), 525–536, 1998.

[32] V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions Information Theory*, vol. 47(3.5), 1902–1914, 2001.

[33] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. Technical Report, Department of Mathematics and Statistics, University of New Mexico, 2000a.

[34] V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In E. Gine, D. Mason, and J. Wellner (ed.), *High Dimensional Probability II*, 443–459, 2000b.

[35] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[36] K. Huang, H. Yang, I. King, and M.R. Lyu. Learning large margin classifiers locally and globally. *ICML*, 2004.

[37] D.S. Yeung, D. Wang, W.W.Y. Ng, E.C.C. Tsang, and X. Zhao. Structured large margin machines: Sensitive to data distributions. *Machine Learning*, vol. 68, 171–200, 2007.

[38] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from examples. Department of Computer Science, University of Chicago, Tech. Rep, TR-2004-06, 2004.

[39] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proceedings of International Workshop on Artificial Intelligence and Statistics*, 2005.

[40] P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, vol. 8, 1369–1392, 2007.

[41] G.R.G. Lanckriet, L.E. Ghaoui, C. Bhattacharyya, and M.I. Jordan. A robust minimax approach to classification. *Journal of Machine Learning Research*, vol. 3, 555–582, 2002.

[42] H. Xue, S. Chen, and Q. Yang. Structural support vector machine. *The Fifth International Symposium on Neural Networks*, Part I, LNCS5263, 2008.