

Support Vector Machines



Armin Shmilovici

1 Introduction

Support vector machines (SVMs) are a set of related methods for supervised learning, applicable to both classification and regression problems. Since the introduction of the SVM classifier over two decades ago (Vapnik 1995), SVM gained popularity due to its solid theoretical foundation. The development of efficient implementations led to numerous applications (Nayak et al. 2015).

The support vector learning machine was developed by Vapnik et al. (Scholkopf et al. 1995; Scholkopf 1997) to constructively implement principles from *statistical learning* theory (Vapnik 2013). In the statistical learning framework, learning means to estimate a function from a set of examples (the training sets). To do this, a learning machine must choose one function from a given set of functions, which minimizes a certain risk (the *empirical risk*) that the estimated function is different from the actual (yet unknown) function. However, the risk depends on the complexity of the set of functions chosen as well as on the training set. Thus, a learning machine must find the best set of functions – as determined by its complexity – and the best function in that set. Unfortunately, in practice, a bound on the risk is neither easily computable, nor very helpful for analyzing the quality of the solution (Vapnik and Chapelle 2000).

Let us assume, for the moment, that the training set is separable by a *hyperplane*. It has been proved (Vapnik 1998) that for the class of hyperplanes, the complexity of the hyperplane can be bounded in terms of another quantity, the *margin*. The margin is defined as the minimal distance of an example to a decision surface. Thus, if we bound the margin of a function class from below, we can control its complexity.

A. Shmilovici (✉)
Ben-Gurion University, Beersheba, Israel
e-mail: armin@bgu.ac.il

Support vector learning implements this insight that the *risk is minimized when the margin is maximized*. An SVM chooses a maximum-margin hyperplane that lies in a transformed input space and splits the example classes while maximizing the distance to the nearest cleanly split examples. The parameters of the solution hyperplane are derived from a quadratic programming optimization problem.

For example, consider a simple separable classification method in multi-dimensional space. Given two classes of examples clustered in feature space, any reasonable classifier hyperplane should pass *between* the means of the classes. One possible hyperplane is the decision surface that assigns a new point to the class whose mean is *closer* to it. This decision surface is geometrically equivalent to computing the class of a new point by checking the angle between two vectors – the vector connecting the two cluster means and the vector connecting the mid-point on that line with the new point. This angle can be formulated in terms of a *dot product* operation between vectors. The decision surface is implicitly defined in terms of the *similarity* between any new point and the cluster mean — a *kernel function*. This simple classifier is linear in the feature space while in the input domain it is represented by a kernel expansion in terms of the training examples. In the more sophisticated techniques presented in the next section, the selection of the examples that the kernels are centered on will no longer consider *all* training examples, and the weights that are put on each data point for the decision surface will no longer be uniform. For instance, we might want to remove the influence of examples that are far away from the decision boundary, either since we expect that they will not improve the generalization error of the decision function, or since we would like to reduce the computational cost of evaluating the decision function. Thus, the hyperplane will only depend on a subset of training examples, called *support vectors*.

There are numerous books and tutorial papers on the theory and practice of SVM (Scholkopf and Smola 2002; Cristianini and Shawe-Taylor 2000; Muller et al. 2001; Chen et al. 2003; Smola and Scholkopf 2004; Deng et al. 2012). The aim of this chapter is to introduce the main SVM models and discuss their main attributes in the framework of supervised learning. The rest of this chapter is organized as follows: Sect. 2 describes the separable classifier case and the concept of kernels; Sect. 3 presents the non-separable case and some related SVM formulations; Sect. 4 discusses some practical computational aspects; Sect. 5 discusses some related concepts and applications; and Sect. 6 concludes with a discussion.

2 Hyperplane Classifiers

The task of classification is to find a rule, which based on external observations, assigns an object to one of several classes. In the simplest case, there are only two different classes. One possible formalization of this classification task is to estimate a function $f : \mathbf{R}^N \rightarrow \{-1, +1\}$ using input-output training data pairs generated identically and independently distributed (i.i.d.) according to an unknown

probability distribution $P(\mathbf{x}, y)$ of the data $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \in \mathbf{R}^N \times Y$, $Y = \{-1, +1\}$ such that f will correctly classify unseen examples (\mathbf{x}, y) . The test examples are assumed to be generated from the same probability distribution as the training data. An example is assigned to class $+1$ if $f(\mathbf{x}) \geq 0$ and to class -1 otherwise.

The best function f that one can obtain is the one minimizing the expected error (risk) – the integral of a certain *loss function* l according to the unknown probability distribution $P(\mathbf{x}, y)$ of the data. For classification problems, l is the so-called 0/1 loss function: $l(f(\mathbf{x}), y) = \theta(-yf(\mathbf{x}))$, where $\theta(z) = 0$ for $z < 0$ and $\theta(z) = 1$ otherwise. The loss framework can also be applied to regression problems where $y \in \mathbf{R}$, where the most common loss function is the squared loss: $l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$.

Unfortunately, the risk cannot be minimized directly, since the underlying probability distribution $P(\mathbf{x}, y)$ is unknown. Therefore, we must try to estimate a function that is close to the optimal one based on the available information, i.e., the training sample and properties of the function class from which the solution f is chosen. To design a learning algorithm, one needs to come up with a class of functions whose capacity (to classify data) can be computed. The intuition, which is formalized in Vapnik (2013), is that a simple (e.g., linear) function that explains most of the data is preferable to a complex one (Occam's razor).

2.1 The Linear Classifier

Let us assume for a moment that the training sample is separable by a hyperplane (see Fig. 1) and we choose functions of the form

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0 \quad \mathbf{w} \in \mathbf{R}^N, b \in \mathbf{R} \quad (1)$$

corresponding to decision functions

$$f(\mathbf{x}) = \text{sign}((\mathbf{w} \cdot \mathbf{x}) + b) \quad (2)$$

It has been shown (Vapnik 1995) that, for the class of hyperplanes, the capacity of the function can be bounded in terms of another quantity, the margin (Fig. 1). The margin is defined as the minimal distance of a sample to the decision surface. The margin depends on the length of the weight vector \mathbf{w} in Eq. (1): since we assumed that the training sample is separable, we can rescale \mathbf{w} and b such that the points closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1$ (i.e., obtain the so-called canonical representation of the hyperplane). Now consider two samples \mathbf{x}_1 and \mathbf{x}_2 from different classes with $|(\mathbf{w} \cdot \mathbf{x}_1) + b| = 1$ and $|(\mathbf{w} \cdot \mathbf{x}_2) + b| = 1$, respectively. Then, the margin is given by the distance of these two points, measured perpendicular to the hyperplane, i.e., $\left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_1 - \mathbf{x}_2) \right) = \frac{2}{\|\mathbf{w}\|}$.

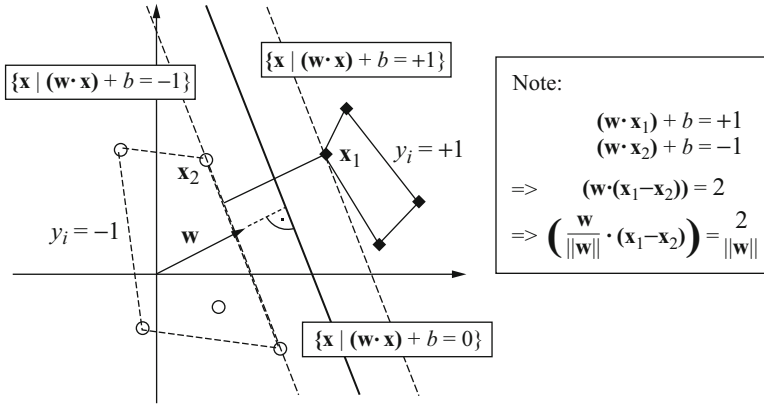


Fig. 1 A toy binary classification problem: separate balls from diamonds. The optimal hyperplane is orthogonal to the shortest line connecting the convex hull of the two classes (dotted), and intersects it half way between the two classes. In this case, the margin is measured perpendicular to the hyperplane. (Figure taken from Muller et al. (2001))

Among all the hyperplanes separating the data, there exists a unique one yielding the maximum margin of separation between the classes:

$$\text{Max}_{\{w,b\}} \min \left\{ \|x - x_i\| : x \in \mathbf{R}^N, (w \cdot x) + b = 0, i = 1, \dots, n \right\} \quad (3)$$

To construct this optimal hyperplane, one solves the following optimization problem:

$$\text{Min}_{\{w,b\}} \frac{1}{2} \|w\|^2 \quad (4)$$

$$\text{Subject to } y_i \cdot ((w \cdot x_i) + b) \geq 1, i = 1, \dots, n \quad (5)$$

This constraint optimization problem can be solved by introducing Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian function

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i \cdot ((w \cdot x_i) + b) - 1) \quad (6)$$

The Lagrangian L has to be minimized with respect to the *primal* variables $\{w, b\}$ and maximized with respect to the *dual* variables α_i . The optimal point is a saddle point and we have the following equations for the primal variables:

$$\frac{\partial L}{\partial b} = 0, \frac{\partial L}{\partial w} = 0 \quad (7)$$

which translate into

$$\sum_{i=1}^n \alpha_i y_i = 0, \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \quad (8)$$

The solution vector thus has an expansion in terms of a subset of the training patterns. The *support vectors* are those patterns corresponding with the non-zero α_i , and the non-zero α_i are called *support values*. By the Karush-Kuhn-Tucker complimentary conditions of optimization, the α_i must be zero for all the constraints in (5) which are not met as equality, thus

$$\alpha_i (y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) - 1) = 0, \quad i = 1, \dots, n \quad (9)$$

and all the support vectors lie on the margin (Figs. 1 and 3) while all the remaining training examples are irrelevant to the solution. The hyperplane is completely captured by the patterns closest to it.

For a nonlinear problem like (4) and (5), called a primal problem, under certain conditions, the primal and dual problems have the same objective values. Therefore, we can solve the dual problem which may be easier than the primal problem. In particular, when working in feature space (Sect. 2.3) solving the dual may be the only way to train the SVM. By substituting (8) into (6), one eliminates the primal variables and arrives at the Wolfe dual (Wolfe 1961) of the optimization problem for the multipliers α_i :

$$\text{Max}_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (10)$$

$$\text{Subject to } \alpha_i \geq 0, \quad i = 1, \dots, n, \quad \sum_{i=1}^n \alpha_i y_i = 0 \quad (11)$$

The hyperplane decision function (2) can now be explicitly written as

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^n \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i) + b \right) \quad (12)$$

where b is computed from (9) and from the set of support vectors \mathbf{x}_i , $i \in I \equiv \{i : \alpha_i \neq 0\}$.

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^n \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \right) \quad (13)$$

2.2 The Kernel Trick

The choice of linear classifier functions seems to be very limited (i.e., likely to underfit the data). Fortunately, it is possible to have both linear models and a very rich set of nonlinear decision functions by using the kernel trick (Cortes and Vapnik 1995) with maximum-margin hyperplanes. Using the kernel trick for SVM makes the maximum-margin hyperplane be fit in a feature space F . The feature space F is a nonlinear map $\Phi : \mathbf{R}^N \rightarrow F$ from the original input space, usually of much higher dimensionality than the original input space. With the kernel trick, the same linear algorithm is worked on the transformed data $(\Phi(\mathbf{x}_1), y_1), \dots, (\Phi(\mathbf{x}_n), y_n)$. In this way, nonlinear SVMs can make the maximum-margin hyperplane fit in a feature space. Figure 2 demonstrates such a case. In the original (linear) training algorithm (10), (11), and (12) the data appears in the form of dot products $\mathbf{x}_i \cdot \mathbf{x}_j$. Now, the training algorithm depends on the data through dot products in F , i.e., on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. If there exists a kernel function K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, we would only need to use K in the training algorithm and would never need to explicitly even know what Φ is.

Mercer's condition (Vapnik 2013) tells us the mathematical properties to check whether or not a prospective kernel is actually a dot product in some space, but it does not tell us how to construct Φ , or even what F is. Choosing the best kernel function is a subject of active research (Scholkopf and Smola 2002; Steinwart 2003). It was found that to a certain degree different choices of kernels give similar classification accuracy and similar sets of support vectors (Scholkopf et al.

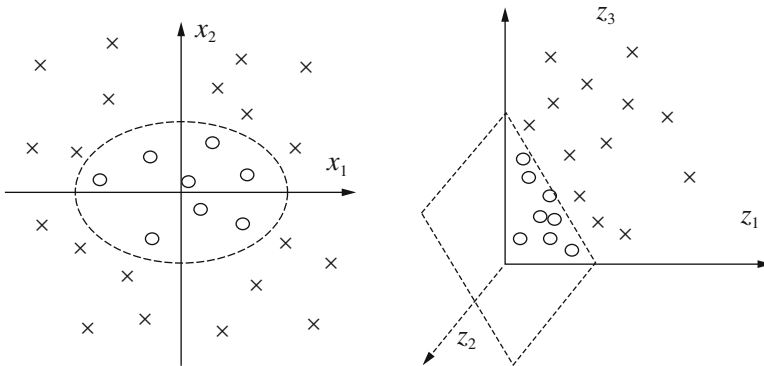


Fig. 2 The idea of SVM is to map the training data into a higher dimensional feature space via Φ , and construct a separating hyperplane with maximum margin there. This yields a nonlinear decision boundary in input space. In the following two-dimensional classification example, the transformation is $\Phi : \mathbf{R}^2 \rightarrow \mathbf{R}^3, (x_1, x_2) \rightarrow (z_1, z_2, z_3) \equiv (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. The separating hyperplane is visible and the decision surface can be analytically found. (Figure taken from Muller et al. (2001))

Table 1 Commonly used kernel functions

Kernel	$K(\mathbf{x}, \mathbf{x}_i)$
Radial Basis Function	$\exp(-\gamma \ \mathbf{x} - \mathbf{x}_i\ ^2)$, $\gamma > 0$
Inverse multiquadratic	$\frac{1}{\sqrt{\ \mathbf{x} - \mathbf{x}_i\ + \eta}}$
Polynomial of degree d	$(\gamma (\mathbf{x}^T \cdot \mathbf{x}_i) + \eta)^d$, $\gamma > 0$
Sigmoidal	$\tanh(\gamma (\mathbf{x}^T \cdot \mathbf{x}_i) + \eta)$, $\gamma > 0$
Linear	$\mathbf{x}^T \cdot \mathbf{x}_i$

1995), indicating that in some sense there exist “important” training points which characterize a given problem.

Some commonly used kernels are presented in Table 1. Note, however, that the Sigmoidal kernel only satisfies the Mercer’s condition for certain values of the parameters and the data. Hsu et al. (2016) advocated the use of the Radial Basis Function as a reasonable first choice.

2.3 The Optimal Margin Support Vector Machine

Using the kernel trick, replace every dot product $(\mathbf{x}_i \cdot \mathbf{x}_j)$ in terms of the kernel K evaluated on input patterns $\mathbf{x}_i, \mathbf{x}_j$. Thus, we obtain the more general form of (12):

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^n \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \right) \quad (14)$$

and the following quadratic optimization problem

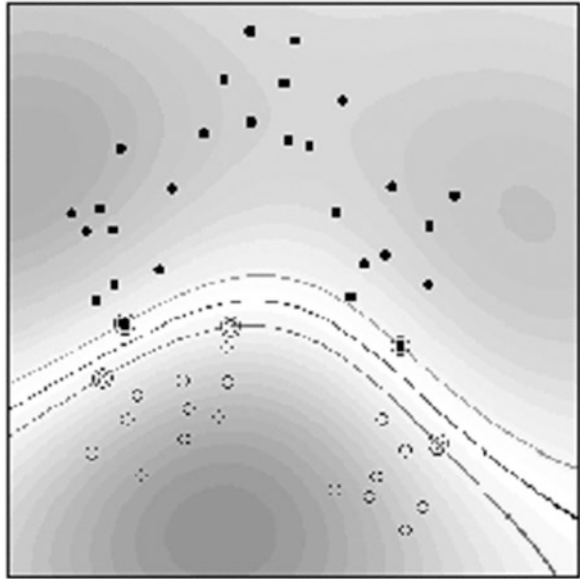
$$\text{Max}_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (15)$$

$$\text{Subject to } \alpha_i \geq 0, \quad i = 1, \dots, n, \quad \sum_{i=1}^n \alpha_i y_i = 0 \quad (16)$$

Formulation (15) and (16) is the standard SVM formulation. This dual problem has the same number of variables as the number of training variables, while the primal problem has a number of variables which depends on the dimensionality of the feature space, which could be infinite. Figure 3 presents an example of a decision function found with an SVM.

One of the most important properties of the SVM is that the solution is *sparse* in α , i.e., many patterns are outside the margin area and their optimal α_i is zero. Without this sparsity property, SVM learning would hardly be practical for large data sets.

Fig. 3 Example of a support vector classifier found by using a radial basis function kernel. Circles and disks are two classes of training examples. Extra circles mark the support vectors found by the algorithm. The middle line is the decision surface. The outer lines precisely meet the constraint (16). The shades indicate the absolute value of the argument of the sign function in (14). (Figure taken from Chen et al. (2003))



3 Non-separable SVM Models

The previous section considered the separable case. However, in practice, a separating hyperplane may not exist, e.g., if a high noise level causes some overlap of the classes. Using the previous SVM might not minimize the empirical risk. This section presents some SVM models that extend the capabilities of hyperplane classifiers to more practical problems.

3.1 *Soft Margin Support Vector Classifiers*

To allow for the possibility of examples violating constraint (5), Cortes and Vapnik (1995) introduced slack variables ξ_i that relax the hard margin constraints

$$y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n \quad (17)$$

A classifier that generalizes well is then found by controlling both the classifier capacity (via $\|\mathbf{w}\|$) and the sum of the slacks $\sum_{i=1}^n \xi_i$, i.e., the number of training errors. One possible realization, called C-SVM, of a soft margin classifier is minimizing the following objective function

$$\text{Minimize}_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad (18)$$

The regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term. Incorporating Lagrange multipliers and solving leads to the following dual problem:

$$\text{Max}_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (19)$$

$$\text{Subject to } 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n, \quad \sum_{i=1}^n \alpha_i y_i = 0 \quad (20)$$

The only difference from the separable case is the upper bound C on the Lagrange multipliers α_i . The solution remains sparse and the decision function retains the same form as (14).

Another possible realization of a soft margin, called ν -SVM (Chen et al. 2003), was originally proposed for regression. The rather non-intuitive regularization constant C is replaced with another constant $\nu \in [0, 1]$. The dual formulation of the ν -SVM is the following:

$$\text{Max}_{\alpha} - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (21)$$

$$\text{Subject to } 0 \leq \alpha_i \leq \frac{1}{n}, \quad i = 1, \dots, n, \quad \sum_{i=1}^n \alpha_i y_i = 0, \quad \sum_{i=1}^n \alpha_i \geq \nu \quad (22)$$

For appropriate parameter choices, the ν -SVM yields exactly the same solutions as the C-SVM. The significance of ν is that under some mild assumptions about the data, ν is an upper bound on the fraction of margin errors (and hence also on the fraction of training errors); and ν is also a lower bound on the fraction of support vectors. Thus, controlling ν influences the trade-off between the model's accuracy and the model's complexity.

3.2 Support Vector Regression

One possible formalization of the regression task is to estimate a function $f : \mathbf{R}^N \rightarrow \mathbf{R}$ using input-output training data pairs generated identically and independently distributed (i.i.d.) according to an unknown probability distribution $P(\mathbf{x}, y)$ of the data. The concept of margin is specific to classification. However, we

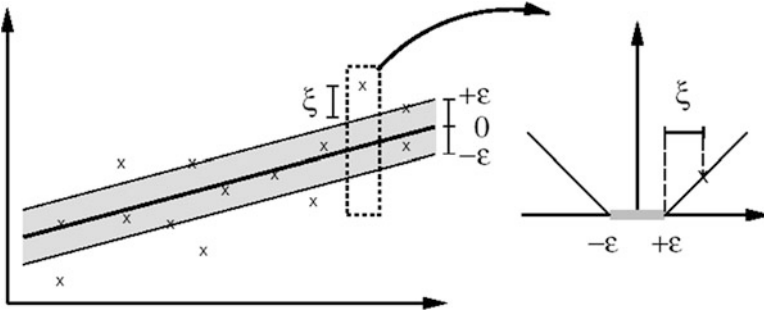


Fig. 4 In SV regression, a tube with radius ε is fitted to the data. The optimization determines a trade-off between model complexity and points lying outside of the tube. (Figure taken from Smola and Scholkopf (2004))

would still like to avoid too complex regression functions. The idea of SVR (Smola and Scholkopf 2004) is that we find a function that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, errors are unimportant as long as they are less than ε , but we do not tolerate deviations larger than this. An analog of the margin is constructed in the space of the target values $y \in \mathbf{R}$. By using Vapnik’s ε -sensitive loss function (Fig. 4).

$$|y - f(\mathbf{x})|_\varepsilon \equiv \max \{0, |y - f(\mathbf{x})| - \varepsilon\} \tag{23}$$

A tube with radius ε is fitted to the data, and a regression function that generalizes well is then found by controlling both the regression capacity (via $\|\mathbf{w}\|$) and the loss function. One possible realization, called C-SVR, of a is minimizing the following objective function

$$\text{Minimize}_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n |y_i - f(\mathbf{x})|_\varepsilon \tag{24}$$

The regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term.

Generalization to kernel-based regression estimation is carried out in complete analogy with the classification problem. Introducing Lagrange multipliers and choosing a priori the regularization constants C, ε one arrives at a dual quadratic optimization problem. The support vectors and the support values of the solution define the following regression function

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{25}$$

There are degrees of freedom for constructing SVR, such as how to penalize or regularize different parts of the vector, how to use the kernel trick, and the loss function to use. For example, in the ν -SVR algorithm implemented in LIBSVM (Chang and Lin 2011) one specifies an upper bound $0 \leq \nu \leq 1$ on the fraction of points allowed to be outside the tube (asymptotically, the number of support vectors). For a priori chosen constants C, ν the dual quadratic optimization problem is as follows:

$$\text{Max}_{\alpha, \alpha^*} \sum_{i=1}^n (\alpha_i^* - \alpha_i) y_i - \frac{1}{2} \sum_{i,j=1}^n (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j) \quad (26)$$

$$\text{Subject to } 0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{n}, \quad \begin{aligned} \sum_{i=1}^n (\alpha_i^* + \alpha_i) &\leq C\nu \\ \sum_{i=1}^n (\alpha_i^* - \alpha_i) &\leq C\nu \end{aligned} \quad i = 1, \dots, n \quad (27)$$

and the regression solution is expressed as

$$f(\mathbf{x}) = \sum_{i=1}^n (\alpha_i^* - \alpha_i) K(\mathbf{x}, \mathbf{x}_i) + b \quad (28)$$

3.3 SVM-Like Models

The power of SVM comes from the kernel representation that allows a nonlinear mapping of input space to a higher dimensional feature space. However, the resulting quadratic programming equations may be computationally expensive for large problems. Smola et al. (1999) suggested an SVR-like *linear programming* formulation that retains the form of the solution (25) while replacing the quadratic function (26) with a linear function subject to constraints on the error of kernel expansion (25).

Suykens et al. (2002) introduced the least squares SVM (LS-SVM) in which they modify the classifier of Eqs. (17) and (18) with the following equations:

$$\text{Minimize}_{\mathbf{w}, b, \mathbf{e}} \frac{1}{2} \|\mathbf{w}\|^2 + \gamma \frac{1}{2} \sum_{i=1}^n e_i^2 \quad (29)$$

$$\text{Subject to } y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) = 1 - e_i, \quad i = 1, \dots, n \quad (30)$$

Important differences with standard SVM are the equality constraint (30) and the sum squared error terms, which greatly simplify the problem. Incorporating Lagrange multipliers and solving leads to the following dual *linear problem*:

$$\begin{bmatrix} \mathbf{0} & \mathbf{Y}^T \\ \mathbf{Y} & \mathbf{\Omega} + \gamma^{-1}\mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \quad (31)$$

where the primal variables $\{\mathbf{w}, b\}$ define as before a decision surface like (14), $Y = (y_1, \dots, y_n)$, $(\mathbf{\Omega})_{i,j} = y_i y_j K(x_i, x_j)$, $\mathbf{1}, \mathbf{0}$ are appropriate size all ones (all zeros) matrices, and γ is a tuning parameter to be optimized. Equivalently, modifying the regression problem (26) and (27) also results in a linear system like (31) with additional tuning parameter.

The LS-SVM can realize strongly nonlinear decision boundaries, and efficient matrix inversion methods can handle very large datasets. However, $\boldsymbol{\alpha}$ is *not* sparse anymore (Suykens et al. 2002).

4 Implementation Issues with SVM

The purpose of this section is to overview some problems that face the application of SVM in machine learning.

4.1 Optimization Techniques

The solution of the SVM problem is the solution of a constraint (convex) quadratic programming (QP) problem such as (15) and (16). Equation (15) can be rewritten as maximizing $-\frac{1}{2}\boldsymbol{\alpha}^T \hat{\mathbf{K}}\boldsymbol{\alpha} + \mathbf{1}^T \boldsymbol{\alpha}$, where $\mathbf{1}$ is a vector of all ones and $\hat{K}_{i,j} = y_i y_j k(x_i, x_j)$. When the Hessian matrix $\hat{\mathbf{K}}$ is positive definite, the objective function is convex and there is a *unique global solution*. If matrix $\hat{\mathbf{K}}$ is positive semi-definite, every maximum is also a global maximum, however, there can be several optimal solutions (different in their $\boldsymbol{\alpha}$) which might lead to different performance on the testing dataset.

In general, the support vector optimization can be solved analytically only when the number of training data is very small. The worst case computational complexity for the general analytic case results from the inversion of the Hessian matrix, thus is of order N_S^3 , where N_S is the number of support vectors. There exists a vast literature on solving quadratic programs (Bertsekas 2016; Bazaraa et al. 2006) and several software packages are available. However, most quadratic programming algorithms are either only suitable for small problems or assume that the Hessian matrix $\hat{\mathbf{K}}$ is sparse, i.e., most elements of this matrix are zero. Unfortunately, this is not true for the SVM problem. Thus, using standard quadratic programming codes with more than a few hundred variables results in enormous training times and more demanding memory needs. Nevertheless, the structure of the SVM optimization problem allows the derivation of specially tailored algorithms, which allow for fast convergence with small memory requirements, even on large problems.

A key observation in solving large-scale SVM problems is the sparsity of the solution (Steinwart 2004). Depending on the problem, many of the optimal α_i will either be zero or on the upper bound. If one could know beforehand which α_i were zero, the corresponding rows and columns could be removed from the matrix $\hat{\mathbf{K}}$ without changing the value of the quadratic form. Furthermore, a point can only be optimal if it fulfills the KKT conditions (such as Eq. (5)). SVM solvers decompose the quadratic optimization problem into a sequence of smaller quadratic optimization problems that are solved in sequence. Decomposition methods are based on the observations of Osuna et al. (1997) that each QP in a sequence of QPs always contains at least one sample violating the KKT conditions. The classifier built from solving the QP for part of the training data is used to test the rest of the training data. The next partial training set is generated from combining the support vectors already found (the “working set”) with the points that most violate the KKT conditions, such that the partial Hessian matrix will fit the memory. The algorithm will eventually converge to the optimal solution. Decomposition methods differ in the strategies for generating the smaller problems and use sophisticated heuristics to select several patterns to add and remove from the sub-problem plus efficient caching methods. They usually achieve fast convergence even on large data sets with up to several thousands of support vectors. A quadratic optimizer is still required as part of the solver. Elements of the SVM solver can take advantage of parallel processing: such as simultaneous computing of the Hessian matrix, dot products, and the objective function. More details and tricks can be found in the literature (Cai and Cherkassky 2012; Tavara 2019; Smola et al. 2000; Deng et al. 2012; Chang and Lin 2001; Chew et al. 2003; Chung et al. 2004). For real-time pattern recognition, on-line SVM algorithms (Zhou et al. 2015) avoid full SVM retraining when comes a new sample via off-line pre-selection of few important training data (Wang 2013).

A fairly large selection of optimization codes for SVM classification and regression has been developed. They range from simple MATLAB implementation to sophisticated C, C++, or FORTRAN programs (e.g., LIBSVM: Chang and Lin 2011, SVMlight: Joachims 2008). Some solvers include integrated model selection and data rescaling procedures for improved speed and numerical stability. Hsu et al. (2016) advise about working with an SVM software on practical problems.

4.2 Model Selection

To obtain a high level of performance, some parameters of the SVM algorithm have to be tuned. These include (1) the selection of the kernel function; (2) the kernel parameter(s); (3) the regularization parameters (C , ν , ϵ) for the trade-off between the model complexity and the model accuracy. Model selection techniques provide principled ways to select a proper kernel. Usually, a sequence of models is solved, and using some heuristic rules, next set of parameters is tested. The process is continued until a given criterion is obtained (e.g., 99% correct classification). For example, if we consider 3 alternative (single parameter) kernels, 5 partitions of the

kernel parameters, and one regularization parameter with 5 partitions each, then we need to consider a total of $3 \times 5 \times 5 = 125$ SVM evaluations.

The *cross-validation* technique is widely used for the prediction of the generalization error, and is included in some SVM packages (such as LIBSVM: Chang and Lin 2011). Here, the training samples are divided into k subsets of equal size. Then, the classifier is trained k times: in the i -th iteration ($i = 1, \dots, k$), the classifier is trained on all subsets except the i -th one. Then, the classification error is computed for the i -th subset. It is known that the average of these k errors is a rather good estimate of the generalization error. k is typically 5 or 10. Thus, for the example above we need to consider at least 625 SVM evaluations to identify the model of the best SVM classifier.

In the *Bayesian evidence framework* the training of an SVM is interpreted as Bayesian inference, and the model selection is accomplished by maximizing the marginal likelihood (i.e., evidence). Law and Kwok (2000) and Wenzel et al. (2017) provide iterative parameter updating formulas, and report a significantly smaller number of SVM evaluations.

4.3 Multi-Class SVM

Though SVM was originally designed for two-class problems, several approaches have been developed to extend SVM for multi-class data sets.

One approach to k -class pattern recognition is to consider the problem as a collection of binary classification problems. The technique of *one-against-the-rest* requires k binary classifiers to be constructed (when the label $+1$ is assigned to each class in its turn and the label -1 is assigned to the other $k - 1$ classes). In the prediction stage, a voting scheme is applied to classify a new point. In the *winner-takes-all* voting scheme, one assigns the class with the largest real value. The *one-against-one* approach trains a binary SVM for any two classes of data and obtains a decision function. Thus, for a k -class problem, there are $k(k - 1)/2$ decision functions where the voting scheme is designated to choose the class with the maximum number of votes. More elaborate voting schemes, such as *error-correcting-codes*, consider the combined outputs from the n -parallel classifiers as a binary n -bit code word and select the class with the closest (e.g., Hamming distance) code.

In Hsu and Lin (2002), it was experimentally shown that for general problems, using the C-SVM classifier, various multi-class approaches give similar accuracy. Rifkin and Klautau (2004) have similar observation, however, this may not always be the case. Multi-class methods must be considered together with parameter-selection strategies. That is, we search for appropriate regularization parameters and kernel parameters for constructing a better model (Didiot and Lauer 2015). Chen, Lin and Scholkopf (2003) experimentally demonstrate inconsistent and marginal improvement in the accuracy when the parameters are trained differently for each classifier inside a multi-class C-SVM and ν -SVM classifiers.

5 Extensions and Application

Kernel algorithms have solid foundations in statistical learning theory and functional analysis; thus, kernel methods combine statistics and geometry. Kernels provide an elegant framework for studying fundamental issues of machine learning, such as similarity measures that can incorporate prior knowledge about the problem, and data representations. SVM have been one of the major kernel methods for supervised learning. It is not surprising that recent methods integrate SVM with kernel methods (Scholkopf et al. 1999; Scholkopf and Smola 2002; Shawe-Taylor and Cristianini 2004) for unsupervised learning problems such as density estimation (Weston and Herbrich 2000).

SVM has a strong analogy in *regularization theory* (Williamson et al. 2001). Regularization is a method of solving problems by making some a priori assumptions about the desired function. A penalty term that discourages over-fitting is added to the error function. A common choice of regularizer is given by the sum of the squares of the weight parameters and results in a functional similar to (6). Like SVM, optimizing a functional of the learning function, such as its smoothness, leads to sparse solutions.

Boosting is a machine learning technique that attempts to improve a “weak” learning algorithm, by a convex combination of the original “weak” learning function, each one trained with a different distribution of the data in the training set. SVM can be translated to a corresponding boosting algorithm using the appropriate regularization norm (Ratsch et al. 2001).

Successful applications of SVM algorithms have been reported for various fields, such as pattern recognition (Martin et al. 2002), text categorization (Dumais 1998; Joachims 2002), time series prediction (Mukherjee et al. 1997), and bio-informatics (Zien et al. 2000). Historically, classification experiments with the U.S. Postal Service benchmark problem – the first real-world experiment of SVM (Cortes and Vapnik 1995; Scholkopf 1997) – demonstrated that plain SVMs give a performance very similar to other state-of-the-art methods. SVMs have been achieving excellent results also on the Reuters-22173 text classification benchmark problem (Dumais 1998). SVMs have been strongly improved by using prior knowledge about the problem to engineer the kernels and the support vectors with techniques such as virtual support vectors (Scholkopf 1997; Scholkopf et al. 1998). Nayak et al. (2015) present many more applications.

6 Conclusion

Since the introduction of the SVM classifier over two decades ago, SVM gained popularity due to its solid theoretical foundation in statistical learning theory. They differ radically from comparable approaches such as neural networks: they have a simple geometrical interpretation and SVM training always finds a global mini-

num. The development of efficient implementations led to numerous applications. Selected real-world applications served to exemplify that SVM learning algorithms are indeed highly competitive on a variety of problems.

SVMs are a set of related methods for supervised learning, applicable to both classification and regression problems. This chapter provides an overview of the main SVM methods for the separable and non-separable case and for classification and regression problems. However, SVM methods are being extended to unsupervised learning problems.

An SVM is largely characterized by the choice of its kernel. The kernel can be viewed as a nonlinear similarity measure, and should ideally incorporate prior knowledge about the problem at hand. The best choice of kernel for a given problem is still an open research issue. A second limitation is the speed of training. Training for very large datasets (millions of support vectors) is still an unsolved problem.

References

- Bazaraa M. S., Sherali H. D., and Shetty C. M. *Nonlinear programming: theory and algorithms*. Wiley, third edition, 2006.
- Bertsekas D.P. *Nonlinear Programming*. Athena Scientific, third edition, 2016.
- Cai, F., Cherkassky, V.: Generalized SMO algorithm for SVM-based multitask learning. *IEEE Trans. Neural Network Learning Systems* 2012; **23**(6):997–1003.
- Chang C.-C. and Lin C.-J. Training support vector classifiers: Theory and algorithms. *Neural Computation* 2001; **13**(9):2119–2147.
- Chang C.-C. and Lin C.-J. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2011; **2**:(27):1–27. Software available at <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- Chen P.-H., Lin C. -J., and Scholkopf B. A tutorial on nu-support vector machines. 2003.
- Chew H. G., Lim C. C., and Bogner R. E. An implementation of training dual-nu support vector machines. In Qi, Teo, and Yang, editors, *Optimization and Control with Applications*. Kluwer, 2003.
- Chung K.-M., Kao W.-C., Sun C.-L., and Lin C.-J. Decomposition methods for linear support vector machines. *Neural Computation* 2004; **16**(8):1689–1704.
- Cortes C. and Vapnik V. Support vector networks. *Machine Learning* 1995; **20**:273–297.
- Cristianini N. and Shawe-Taylor J. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.
- Deng N., Tian Y., Zhang C. *Support Vector Machines Optimization Based Theory, Algorithms, and Extensions*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2012.
- Didot E. and Lauer F., Efficient optimization of multi-class support vector machines with MSVM-pack. In *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Springer, 23–34, 2015.
- Dumais S. Using SVMs for text categorization. *IEEE Intelligent Systems* 1998; **13**(4).
- Hsu C.-W. and Lin C.-J. A comparison of methods for multi-class support vector machines *IEEE Transactions on Neural Networks* 2002; **13**(2); 415–425.
- Hsu C.-W. Chang C.-C and Lin C.-J. A practical guide to support vector classification. 2016. Available Online: www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
- Joachims T. *Learning to Classify Text using Support Vector Machines Methods, Theory, and Algorithms*. Kluwer Academic Publishers, 2002.
- Joachims T. 2008, SVMlight, available online http://www.cs.cornell.edu/People/tj/svm_light/

- Law M. H. and Kwok J. T. Bayesian support vector regression. Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics (AISTATS) pages 239–244, Key-West, Florida, USA, January 2000.
- Martin D. R., Fowlkes C. C., and Malik J. Learning to detect natural image boundaries using brightness and texture. In *Advances in Neural Information Processing Systems*, volume 14, 2002.
- Mukherjee S., Osuna E., and Girosi F. Nonlinear prediction of chaotic time series using a support vector machine. In Principe J., Gile L., Morgan N. and Wilson E. editors, *Neural Networks for Signal Processing VII - proceedings of the 1997 IEEE Workshop*, pages 511–520, New-York, IEEE Press, 1997.
- Muller K.-R., Mika S., Ratsch G., Tsuda K., and Scholkopf B., An introduction to kernel-based learning algorithms. *IEEE Neural Networks* 2001; 12(2):181–201.
- Nayak J, Naik B, and Behera H. S. A comprehensive survey on support vector machine in data mining tasks: Applications & challenges. *International Journal of Database Theory and Application* 2015; 8,(1):169–186.
- Osuna E., Freund R., and Girosi F. An improved training algorithm for support vector machines. In Principe J., Gile L., Morgan N. and Wilson E. editors, *Neural Networks for Signal Processing VII - proceedings of the 1997 IEEE Workshop*, pages 276–285, New-York, IEEE Press, 1997.
- Ratsch G., Onoda T., and Muller K.R. Soft margins for AdaBoost. *Machine Learning* 2001; 42(3):287–320.
- Rifkin R. and Klautau A.. In Defense of One-vs-All Classification, *Journal of Machine Learning Research* 2004; 5:101–141.
- Scholkopf B., *Support Vector Learning*. Oldenbourg Verlag, Munich, 1997.
- Scholkopf B., Burges C.J.C., and Vapnik V.N. Extracting support data for a given task. In Fayyad U.M. and Uthurusamy R., Editors, *Proceedings, First International Conference on Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1995.
- Scholkopf B., Simard P.Y., Smola A.J., and Vapnik V.N.. Prior knowledge in support vector kernels. In Jordan M., Kearns M., and Solla S., Editors, *Advances in Neural Information Processing Systems* 10, pages 640–646. MIT Press, Cambridge, MA, 1998.
- Scholkopf B., Burges C. J. C., and Smola A. J., editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, MIT Press, 1999.
- Scholkopf B. and Smola A. J. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- Scholkopf B., Smola A. J., Williamson R. C., and Bartlett P. L. New support vector algorithms. *Neural Computation* 1999; 12:1207–1245.
- Shawe-Taylor J. and Cristianini N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
- Smola A. J., Bartlett P. L., Scholkopf B. and Schuurmans D. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.
- Smola A.J. and Scholkopf B. A tutorial on support vector regression. *Statistics and Computing* 2004; 14(13):199–222.
- Smola A.J., Scholkopf B. and Ratsch G. Linear programs for automatic accuracy control in regression. Proceedings of International Conference on Artificial Neural Networks ICANN'99, Berlin, Springer 1999.
- Steinwart I. On the optimal parameter choice for nu-support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2003; 23(10).
- Steinwart I. Sparseness of support vector machines. *Journal of Machine Learning Research* 2004; 4(6):1071–1105.
- Suykens J.A.K., Van Gestel T., De Brabanter J., De Moor B., and Vandewalle J. *Least Squares Support Vector Machines*. World Scientific Publishing, Singapore, 2002.
- Tavara S. Parallel Computing of Support Vector Machines: A Survey. *ACM Computing Surveys*, 2019;. 51(6):1–38.
- Vapnik V. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
- Vapnik V. *Statistical Learning Theory*. Wiley, NY, 1998.
- Vapnik V. *The Nature of Statistical Learning Theory*. Springer Science. 2013.

- Vapnik V. and Chapelle O. Bounds on error expectation for support vector machines. *Neural Computation* 2000; 12(9):2013–2036.
- Wang, D.I.: Online Support Vector Machine Based on Convex Hull Vertices Selection. *IEEE Transactions on Neural Networks Learning Systems* **2013**; **24**(4), 593–608.
- Weston J. and Herbrich R.. Adaptive margin support vector machines. In Smola A.J., Bartlett P.L., Scholkopf B., and Schuurmans D., Editors, *Advances in Large Margin Classifiers*, pages 281–296, MIT Press, Cambridge, MA, 2000.
- Wenzel F. Galy-Fajou T. Deutsch M., Kloft M. Bayesian Nonlinear Support Vector Machines for Big Data. *ECML/PKDD 2017*: 307–322
- Williamson R. C., Smola A. J., and Scholkopf B. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory* 2001; 47(6):2516–2532.
- Wolfe P. A duality theorem for non-linear programming. *Quarterly of Applied Mathematics* 1961; 19:239–244.
- Zhou X., Zhang X. and Wang B., Online Support Vector Machine: A Survey, in Kim J.H and Geem Z. W. editors, *Harmony Search Algorithm*, pages 269–278, *Proceedings of the 2nd International Conference on Harmony Search Algorithm (ICHSA 2015)*.
- Zien A., Ratsch G., Mika S., Scholkopf B., Lengauer T. and Muller K.R. Engineering support vector machine kernels that recognize translation initiation sites. *Bio-Informatics* 2000; 16(9):799–807.