7

SUPPORT VECTOR MACHINES

> A method of solution is perfect if we can foresee from the start, and even prove, that following that method we shall attain our aim.
>
> Gottfried Leibniz

This chapter describes a family of learning algorithms known as Support Vector Machines (SVM). SVM methodology was developed in Statistical Learning Theory, and later was adopted by researchers in machine learning, statistics and signal processing. Currently, there are many successful SVM applications ranging from genomics to financial engineering. Our description of SVMs is based on important VC-theoretical concepts, such as margin, structural risk minimization, VC-dimension etc. The material presented in this chapter is closely related to Chapters 2 and 4 which can be considered as prerequisite.

Classical VC-theory describes mathematical analysis of predictive learning with finite samples. It does not, however, advocate a particular constructive learning algorithm. All predictive modeling methods implement a trade-off between the accuracy of data fitting and the model complexity. This trade-off is implemented via the Structural Risk Minimization (SRM) formalism in VC-theory. It requires that the admissible models $f(\mathbf{x},\omega)$ should have a nested structure, or complexity ordering. Most statistical and neural network methods described in Chapters 5 and 6 implement a dictionary structure or feature selection structure, in which the goal is to find an *accurate* yet *compact* model for the training data. Here 'accurate' refers to a small fitting error (or empirical risk), and 'compact' refers to a small number of (nonlinear) basis functions (in dictionary methods), or small number of features (in feature selection methods). In both cases, the model complexity is associated with the dimensionality of the derived feature space. This approach may not be feasible for multivariate problems due to the curse of dimensionality. For instance, with polynomial estimators the number of parameters (polynomial coefficients) that require estimation grows exponentially with the problem dimensionality. More generally, polynomial estimators can be viewed as a special case of a mapping from the input ($\mathbf{x}$) space to an intermediate feature ($\mathbf{z}$) space. The dimensionality of $\mathbf{z}$-space determines the size of the optimization problem associated with parameter estimation via data fitting. For example, for MLP neural networks the number of hidden units corresponds to the dimensionality of $\mathbf{z}$-space. Various adaptive statistical and neural network algorithms select a small number of 'informative' features in $\mathbf{z}$-space, in order to control the model complexity.

Technical implementation of this approach is challenging, due to the intrinsic complexity of *nonlinear optimization*, which is required for fitting flexible models. As a result, model complexity may depend on several design parameters of a nonlinear optimization algorithm. For example, for MLP networks trained using backpropagation, model complexity is affected by the initialization of network parameters (weights), the learning rate schedule, the stopping criterion, etc. In practice, model complexity of nonlinear statistical and neural network methods usually depends on half-a-dozen or so tuning parameters. Commercial and public domain software for nonlinear statistical and neural network algorithms provide little guidance for tuning these 'user-defined' parameters. Optimal tuning of model parameters, or model selection, becomes very challenging when the number of parameters exceeds 2 or 3. Practitioners typically resort to default parameter values (that may be sub-optimal), or perform tuning just a few arbitrarily selected model parameters via resampling.

Predictive Learning
By Vladimir Cherkassky

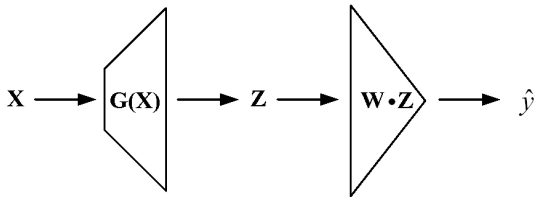

FIGURE 7.1 The support vector machine maps input data $\mathbf{x}$ into a high-dimensional feature space $\mathbf{z}$ using a nonlinear function $G(\mathbf{x})$. A linear approximation in the feature space is used to predict the output.

These difficulties motivate the need for learning algorithms that:

a) Have a few well-defined parameters controlling the model complexity *independent* of the dimensionality of the input space, or the derived feature space;
b) Provide flexible nonlinear parameterization necessary to model diverse data sets with varying statistical characteristics;
c) Provide global minimization of the empirical risk functional, in order to achieve accurate model fitting.

SVM methods satisfy all of these requirements**,** and this explains their success in many applications. However, SVM models lack simple interpretation and are viewed as black-box models. This is often considered as a limitation by practitioners. As argued earlier in Section 4.6, interpretation of predictive data-analytic models is outside the scope of VC theory.

The SVM approach follows conceptual structure shown in Fig. 7.1. This figure illustrates the two main parts of SVM modeling:

1) *Mapping of inputs onto a higher-dimensional feature space* $\mathbf{z}$ using a set of nonlinear basis functions defined *a priori*. Whereas this mapping superficially resembles the feature selection step in traditional statistical and machine learning algorithms, the SVM mapping is quite different. SVM mapping is specified a priori via selection of a kernel function. The purpose of the kernel function is to control the nonlinearity of estimated models. Furthermore, SVM puts no restriction on the number of basis functions (features) used to construct a high-dimensional mapping of the input variables. For the support vector machine, complexity is controlled independently of the dimensionality of the feature space ($\mathbf{z}$-space).

2) *Linear functions with constraints on complexity* are used to model the input samples in the high-dimensional space. SVM modeling uses a linear parameterization $f(\mathbf{z},\omega) = (\mathbf{w}\cdot\mathbf{z}) + b$ to perform data fitting. However, the SVM complexity constraints are implemented using a special type of loss function called margin-based loss. This margin-based loss effectively controls model complexity independent of dimensionality of $\mathbf{z}$-space. The resulting mathematical formulation (described later in this chapter) yields a tractable optimization problem that ensures global minimum of the empirical risk. In contrast, traditional approaches, such as neural networks and statistical methods, perform model fitting via nonlinear optimization. Such nonlinear estimators suffer from two serious drawbacks: lack of efficient complexity control and lack of optimization approaches providing a globally optimal solution.

The SVM methodology is universal, in the sense that it applies to various types of learning problems, such as classification, regression, single class learning etc. In addition, many non-standard learning settings presented later in Chapter 9, can be viewed as an extension of standard SVM. This chapter presents two standard inductive SVM formulations: SVM classification and SVM regression. Even though each type of a learning problem has its specific SVM formulation, they share important concepts, i.e., margin-based loss and kernel mapping. Descriptions in this chapter emphasize these common aspects of different SVM formulations. Section 7.1 describes margin-based loss for different learning problems. Section 7.2 presents linear SVM formulation for classification problems. Nonlinear mapping and its kernel implementation are introduced in Section 7.3. Section 7.4 describes practical issues for SVM classifiers and presents several application examples. SVM methodology for regression is discussed in Section 7.5. Finally, Section 7.6 provides summary.

Material presented in this chapter may be challenging because SVM technology is based on VC-theoretical concepts and several abstract mathematical ideas, such as duality in optimization theory and inner product kernels. Fortunately, there are many commercial and public-domain software implementations of SVM methods. Hence, non-expert users can successfully apply SVM software, even with little understanding of underlying optimization algorithms. To this end, we focus on the descriptions of SVM problems and their optimization formulations. However, we do not provide coverage of the actual optimization algorithms.