The universal approximation property is also true for the class of RBF functions. For sufficiently large values of k in (4.55) the resulting expansion can approximate arbitrarily closely any continuous function in a compact subset S [Park 91, Park 93].

4.17 SUPPORT VECTOR MACHINES: THE NONLINEAR CASE

In Chapter 3, we discussed the support vector machines (SVM) as an optimal design methodology of a linear classifier. Let us now assume that there exists a mapping

$$\boldsymbol{x} \in \mathcal{R}^l \longrightarrow \boldsymbol{y} \in \mathcal{R}^k$$

from the input feature space into a k-dimensional space, where the classes can satisfactorily be separated by a hyperplane. Then, in the framework discussed in Section 4.12, the SVM method can be mobilized for the design of the hyperplane classifier in the new k-dimensional space. However, there is an elegant property in the SVM methodology, that can be exploited for the development of a more general approach. This will also allow us for (implicit) mappings in infinite dimensional spaces, if required.

Recall from Chapter 3 that, in the computations involved in the Wolfe dual representation the feature vectors participate in pairs, via the inner product operation. Also, once the optimal hyperplane $(\boldsymbol{w}, w_0)$ has been computed, classification is performed according to whether the sign of

$$\begin{aligned} g(\boldsymbol{x}) &= \boldsymbol{w}^T \boldsymbol{x} + w_0 \\ &= \sum_{i=1}^{N_s} \lambda_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + w_0 \end{aligned}$$

is $+$ or $-$, where N_s is the number of support vectors. Thus, once more, only inner products enter into the scene. If the design is to take place in the new k-dimensional space, the only difference is that the involved vectors will be the k-dimensional mappings of the original input feature vectors. A naive look at it would lead to the conclusion that now the complexity is much higher, since, usually, k is much higher than the input space dimensionality l, in order to make the classes linearly separable. However, there is a nice surprise just waiting for us. Let us start with a simple example. Assume that

$$\boldsymbol{x} \in \mathcal{R}^2 \longrightarrow \boldsymbol{y} = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

Then, it is a matter of simple algebra to show that

$$y_i^T y_j = (x_i^T x_j)^2$$

In words, the inner product of the vectors in the new (higher dimensional) space has been expressed as a function of the inner product of the corresponding vectors in the original feature space. Most interesting!

Theorem. *Mercer's Theorem. Let $\boldsymbol{x} \in \mathcal{R}^l$ and a mapping $\boldsymbol{\phi}$*

$$\boldsymbol{x} \longrightarrow \boldsymbol{\phi}(\boldsymbol{x}) \in H$$

where H is a Euclidean space.[5] Then the inner product operation has an equivalent representation

$$\sum_r \phi_r(\boldsymbol{x})\phi_r(\boldsymbol{z}) = K(\boldsymbol{x}, \boldsymbol{z}) \tag{4.60}$$

where $\phi_r(\boldsymbol{x})$ is the r-component of the mapping $\boldsymbol{\phi}(\boldsymbol{x})$ of $\boldsymbol{x}$, and $K(\boldsymbol{x}, \boldsymbol{z})$ is a symmetric function satisfying the following condition

$$\int K(\boldsymbol{x}, \boldsymbol{z})g(\boldsymbol{x})g(\boldsymbol{z})\, d\boldsymbol{x}\, d\boldsymbol{z} \geq 0 \tag{4.61}$$

for any $g(\boldsymbol{x}), \boldsymbol{x} \in \mathcal{R}^l$ such that

$$\int g(\boldsymbol{x})^2\, d\boldsymbol{x} < +\infty \tag{4.62}$$

The opposite is also true, i.e., for *any* function $K(\boldsymbol{x}, \boldsymbol{z})$ satisfying (4.61) and (4.62) there exists a space in which $K(\boldsymbol{x}, \boldsymbol{z})$ defines an inner product! Such functions are also known as *kernels* and the space H as *Reproducing kernel Hilbert space (RKHS)*. What, however, Mercer's theorem does not disclose to us is how to find this space. That is, we do not have a general tool to construct the mapping $\boldsymbol{\phi}(\cdot)$ once we know the inner product of the corresponding space. Furthermore, we lack the means to know the dimensionality of the space, which can even be infinite. For more on these issues, the mathematically inclined reader is referred to [Cour 53].

[5] In general, it can be a Hilbert space, i.e., a complete linear space equipped with an inner product operation.

Typical examples of kernels used in pattern recognition applications are

Polynomials

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z} + 1)^q, \quad q > 0 \tag{4.63}$$

Radial Basis Functions

$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right) \tag{4.64}$$

Hyperbolic Tangent

$$K(\boldsymbol{x}, \boldsymbol{z}) = \tanh\left(\beta \boldsymbol{x}^T \boldsymbol{z} + \gamma\right) \tag{4.65}$$

for appropriate values of β and γ so that Mercer's conditions are satisfied. One possibility is $\beta = 2, \ \gamma = 1$. In [Shaw 04] a unified treatment of kernels is presented focusing on their mathematical properties as well as methods for pattern recognition and regression that have been developed around them.

Once an appropriate kernel has been adopted that implicitly defines a mapping into a higher dimensional space (RKHS), the Wolfe dual optimization task (Eqs. (3.91)–(3.93)) becomes

$$\max_{\boldsymbol{\lambda}} \left(\sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \tag{4.66}$$

$$\text{subject to} \quad 0 \le \lambda_i \le C, \quad i = 1, 2, \ldots, N \tag{4.67}$$

$$\sum_i \lambda_i y_i = 0 \tag{4.68}$$

and the resulting linear classifier is

$$\text{assign } \boldsymbol{x} \text{ in } \omega_1(\omega_2) \text{ if } g(\boldsymbol{x}) = \sum_{i=1}^{N_s} \lambda_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + w_0 > (<)\, 0 \tag{4.69}$$

Similar arguments hold true for the ν-SVM formulation.

Figure 4.22 shows the corresponding architecture. This is nothing else than a special case of the generalized linear classifier of Figure 4.17. The number of nodes is determined by the number of support vectors N_s. The nodes perform the inner products between the mapping of $\boldsymbol{x}$ and the corresponding mappings of the support vectors in the high dimensional space, via the kernel operation.

Figure 4.23 shows the resulting SVM classifier for two nonlinearly separable classes, where the Gaussian radial basis function kernel, with $\sigma = 1.75$, has been used. Dotted lines mark the margin and circled points the support vectors.

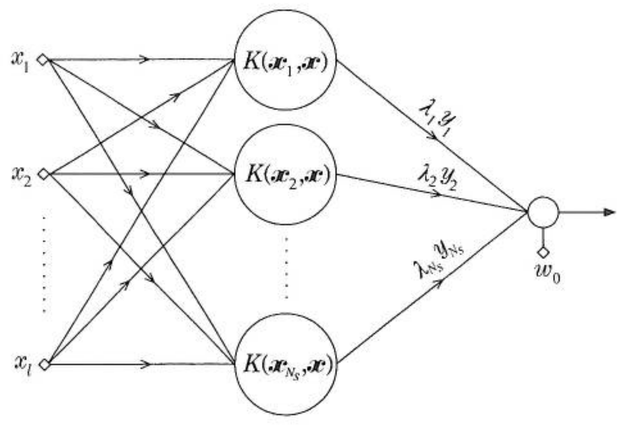
x_1
x_2
x_l
$K(\boldsymbol{x}_1,\boldsymbol{x})$
$K(\boldsymbol{x}_2,\boldsymbol{x})$
$K(\boldsymbol{x}_{N_s},\boldsymbol{x})$
$\lambda_1 y_1$
$\lambda_2 y_2$
$\lambda_{N_s} y_{N_s}$
w_0

FIGURE 4.22: The SVM architecture employing kernel functions.

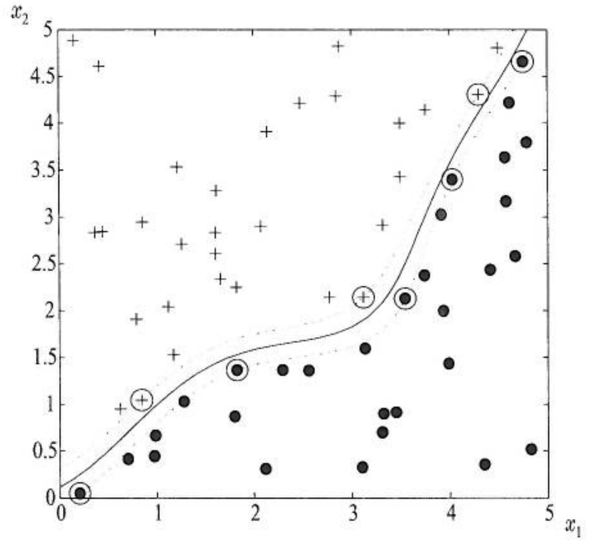

FIGURE 4.23: Example of a nonlinear SVM classifier for the case of two nonlinearly separable classes. The Gaussian RBF kernel was used. Dotted lines mark the margin and circled points the support vectors.

Remarks

- Notice that if the kernel function is the RBF, then the architecture is the same as the RBF network architecture of Figure 4.17. However, the approach followed here is different. In Section 4.15, a mapping in a k-dimensional space was first performed and the centers of the RBF functions had to be estimated. In the SVM approach, the number of nodes as well as the centers are the result of the optimization procedure.

- The hyperbolic tangent function is a sigmoid one. If it is chosen as a kernel, the resulting architecture is a special case of a two layer perceptron. Once more, the number of nodes is the result of the optimization procedure. This is important. Although the SVM architecture is the same as that of a 2-layer perceptron, the training procedure is entirely different for the two methods. The same is true for the RBF networks.
- A notable characteristic of the support vector machines is that the computational complexity is independent of the dimensionality of the kernel space, where the input feature space is mapped. Thus, the curse of dimensionality is bypassed. In other words, one designs in a high dimensional space without having to adopt explicit models using a large number of parameters, as this would be dictated by the high dimensionality of the space. This has also an influence on the generalization properties and indeed, SVM's tend to exhibit *good generalization performance*. We will return to this issue at the end of Chapter 5.
- Another major limitation of the support vector machines is that up to now there is not an efficient practical method for the selection of the best kernel function. This is still an unsolved, yet challenging, research issue. Once a kernel function has been adopted, the so-called kernel parameters (e.g., σ for the Gaussian kernel) as well as the smoothing parameter, C, in the cost function are selected so that the error performance of the resulting classifier can be optimized. Indeed, this set of parameters, also known as *hyperparameters*, is crucial for the generalization capabilities of the classifier (i.e., its error performance when it is "confronted" with data outside the training set).

  To this end, a number of easily computed bounds, which relate to the generalization performance of the classifier, have been proposed and used for the best choice of the hyperparameters. The most common procedure is to solve the SVM task for different sets of hyperparameters and finally select the SVM classifier corresponding to the set optimizing the adopted bound. See, for example, [Bart 02, Lin 02, Duan 03, Angu 03, Lee 04]. In [Chap 02] this problem is treated in a minimax framework: maximize the margin over the $\boldsymbol{w}$ and minimize the bound over the hyperparameters.

  A different approach to the task of data-adaptive kernel tuning, with the same goal of improving the error performance, is via information geometry arguments [Amar 99]. The basic idea behind this approach is to introduce a conformal mapping into the Riemannian geometry induced by the chosen kernel function, aiming at enhancing the margin. In [Burg 99] it is pointed out that the feature vectors, which originally lie in the l-dimensional space, after the mapping induced by the kernel function lie in an l-dimensional surface, S, in the high-dimensional space. It turns out that (under some very general assumptions) S is a Riemannian manifold with a metric that can be expressed solely in terms of the kernel.

- Support vector machines have been applied to a number of diverse applications, ranging from handwritten digit recognition ([Cort 95]), to object recognition ([Blan 96]), person identification ([Ben 99]), spam categorization ([Druc 99]), channel equalization ([Seba 00]), and medical imaging [ElNa 02]. The results from these applications indicate that SVM classifiers exhibit enhanced generalization performance, which seems to be the power of support vector machines. An extensive comparative study concerning the performance of SVM against sixteen other popular classifiers, using twenty one different data sets, is given in [Meye 03]. The results verify that SVM classifiers rank at the very top among these classifiers, although there are cases for which other classifiers gave lower error rates.
- Besides support vector machines any other linear classifier that employs inner products can implicitly be executed in higher dimensional spaces by using kernels. In this way one can elegantly construct nonlinear versions of a linear algorithm. For a review on this issue see, for example, [Mull 01, Shaw 04].

4.18 DECISION TREES

In this section we briefly review a large class of nonlinear classifiers known as *decision trees*. They are *multistage* decision systems in which classes are sequentially rejected until we reach a finally accepted class. To this end, the feature space is split into unique regions, corresponding to the classes, *in a sequential manner*. Upon the arrival of a feature vector, the searching of the region to which the feature vector will be assigned is achieved via a sequence of decisions along a path of *nodes* of an appropriately constructed *tree*. Such schemes offer advantages when a large number of classes is involved. The most popular among the decision trees are those that split the space into hyperrectangles with sides parallel to the axes. The sequence of decisions is applied to individual features, and the questions to be answered are of the form "*is feature* $x_i \leq \alpha$?" where α is a threshold value. Such trees are known as *ordinary binary classification trees (OBCTs)*. Other types of trees are also possible that split the space into convex polyhedral cells or into pieces of spheres.

The basic idea behind an OBCT is demonstrated via the simplified example of Figure 4.24. By a successive sequential splitting of the space we have created regions corresponding to the various classes.