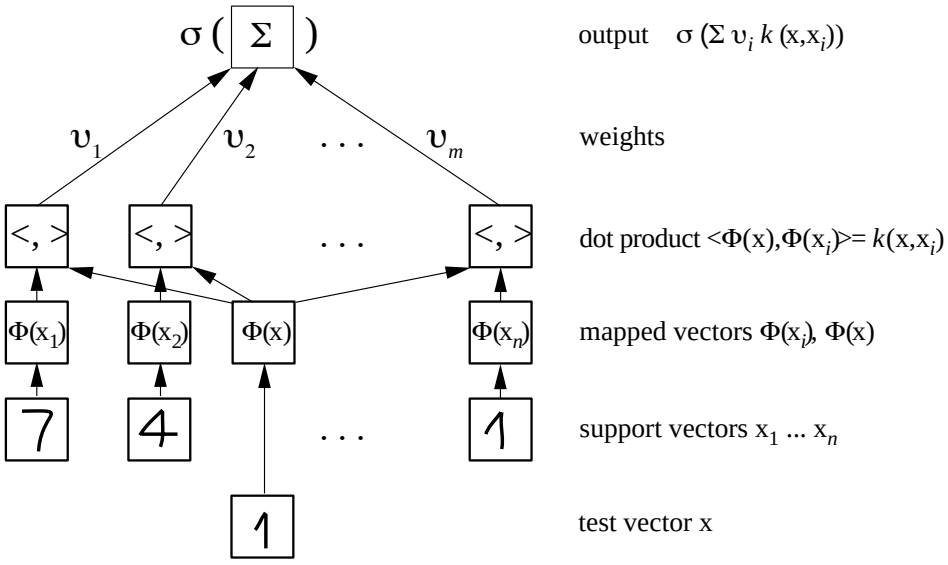

Figure 1.9 Architecture of SVMs and related kernel methods. The input x and the expansion patterns (SVs) x_i (we assume that we are dealing with handwritten digits) are nonlinearly mapped (by Φ) into a feature space $\mathcal{H}$ where dot products are computed. Through the use of the kernel k, these two layers are in practice computed in one step. The results are linearly combined using weights v_i, found by solving a quadratic program (in pattern recognition, $v_i = y_i \alpha_i$; in regression estimation, $v_i = \alpha_i^* - \alpha_i$) or an eigenvalue problem (Kernel PCA). The linear combination is fed into the function σ (in pattern recognition, $\sigma(x) = \operatorname{sgn}(x + b)$; in regression estimation, $\sigma(x) = x + b$; in Kernel PCA, $\sigma(x) = x$).

1.8 Empirical Results and Implementations

Examples of Kernels

Having described the basics of SVMs, we now summarize some empirical findings. By the use of kernels, the optimal margin classifier was turned into a high-performance classifier. Surprisingly, it was observed that the polynomial kernel

$$k(x, x') = \langle x, x' \rangle^d, \tag{1.61}$$

the Gaussian

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\,\sigma^2}\right), \tag{1.62}$$

and the sigmoid

$$k(x, x') = \tanh\left(\kappa\,\langle x, x' \rangle + \Theta\right), \tag{1.63}$$

with suitable choices of $d \in \mathbb{N}$ and $\sigma, \kappa, \Theta \in \mathbb{R}$ (here, $\mathcal{X} \subset \mathbb{R}^N$), empirically led to SV classifiers with very similar accuracies and SV sets (Section 7.8.2). In this sense, the SV set seems to characterize (or *compress*) the given task in a manner which

to some extent is independent of the type of kernel (that is, the type of classifier) used, provided the kernel parameters are well adjusted.

Applications

Initial work at AT&T Bell Labs focused on OCR (optical character recognition), a problem where the two main issues are classification accuracy and classification speed. Consequently, some effort went into the improvement of SVMs on these issues, leading to the *Virtual SV* method for incorporating prior knowledge about transformation invariances by transforming SVs (Chapter 7), and the *Reduced Set* method (Chapter 18) for speeding up classification. Using these procedures, SVMs soon became competitive with the best available classifiers on OCR and other object recognition tasks [87, 57, 419, 438, 134], and later even achieved the world record on the main handwritten digit benchmark dataset [134].

Implementation

An initial weakness of SVMs, less apparent in OCR applications which are characterized by low noise levels, was that the size of the quadratic programming problem (Chapter 10) scaled with the number of support vectors. This was due to the fact that in (1.36), the quadratic part contained at least all SVs — the common practice was to extract the SVs by going through the training data in chunks while regularly testing for the possibility that patterns initially not identified as SVs become SVs at a later stage. This procedure is referred to as *chunking*; note that without chunking, the size of the matrix in the quadratic part of the objective function would be $m \times m$, where m is the number of all training examples.

What happens if we have a high-noise problem? In this case, many of the slack variables ξ_i become nonzero, and all the corresponding examples become SVs. For this case, decomposition algorithms were proposed [398, 409], based on the observation that not only can we leave out the non-SV examples (the x_i with $\alpha_i = 0$) from the current chunk, but also some of the SVs, especially those that hit the upper boundary ($\alpha_i = C$). The chunks are usually dealt with using quadratic optimizers. Among the optimizers used for SVMs are LOQO [555], MINOS [380], and variants of conjugate gradient descent, such as the optimizers of Bottou [459] and Burges [85]. Several public domain SV packages and optimizers are listed on the web page http://www.kernel-machines.org. For more details on implementations, see Chapter 10.

Once the SV algorithm had been generalized to regression, researchers started applying it to various problems of estimating real-valued functions. Very good results were obtained on the Boston housing benchmark [529], and on problems of times series prediction (see [376, 371, 351]). Moreover, the SV method was applied to the solution of inverse function estimation problems ([572]; cf. [563, 589]). For overviews, the interested reader is referred to [85, 472, 504, 125].