gous manner to the case of pattern recognition. Introducing Lagrange multipliers, one arrives at the following optimization problem (for $C, \varepsilon \geq 0$ chosen a priori):

$$\underset{\boldsymbol{\alpha},\boldsymbol{\alpha}^* \in \mathbb{R}^m}{\text{maximize}} \quad W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\varepsilon \sum_{i=1}^{m} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) y_i - \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) k(x_i, x_j) \tag{1.51}$$

$$\text{subject to} \quad 0 \leq \alpha_i, \alpha_i^* \leq C \text{ for all } i = 1, \ldots, m, \text{ and } \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0. \tag{1.52}$$

Regression Function

The regression estimate takes the form

$$f(x) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) k(x_i, x) + b, \tag{1.53}$$

where b is computed using the fact that (1.48) becomes an equality with $\xi_i = 0$ if $0 < \alpha_i < C$, and (1.49) becomes an equality with $\xi_i^* = 0$ if $0 < \alpha_i^* < C$ (for details, see Chapter 9). The solution thus looks quite similar to the pattern recognition case (cf. (1.35) and Figure 1.9).

A number of extensions of this algorithm are possible. From an abstract point of view, we just need some target function which depends on $(\mathbf{w}, \boldsymbol{\xi})$ (cf. (1.47)). There are multiple degrees of freedom for constructing it, including some freedom how to penalize, or regularize. For instance, more general loss functions can be used for $\boldsymbol{\xi}$, leading to problems that can still be solved efficiently ([512, 515], cf. Chapter 9). Moreover, norms other than the 2-norm $\|.\|$ can be used to regularize the solution (see Sections 4.9 and 9.4).

ν-SV Regression

Finally, the algorithm can be modified such that ε need not be specified a priori. Instead, one specifies an upper bound $0 \leq \nu \leq 1$ on the fraction of points allowed to lie outside the tube (asymptotically, the number of SVs) and the corresponding ε is computed automatically. This is achieved by using as primal objective function

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\left(\nu m \varepsilon + \sum_{i=1}^{m} |y_i - f(\mathbf{x}_i)|_\varepsilon\right) \tag{1.54}$$

instead of (1.46), and treating $\varepsilon \geq 0$ as a parameter over which we minimize. For more detail, cf. Section 9.3.

1.7 Kernel Principal Component Analysis

The kernel method for computing dot products in feature spaces is not restricted to SVMs. Indeed, it has been pointed out that it can be used to develop nonlinear generalizations of any algorithm that can be cast in terms of dot products, such as principal component analysis (PCA) [480].

Principal component analysis is perhaps the most common feature extraction algorithm; for details, see Chapter 14. The term *feature extraction* commonly refers to procedures for extracting (real) numbers from patterns which in some sense represent the crucial information contained in these patterns.

PCA in feature space leads to an algorithm called *kernel PCA*. By solving an eigenvalue problem, the algorithm computes nonlinear feature extraction functions

$$f_n(x) = \sum_{i=1}^{m} \alpha_i^n k(x_i, x), \tag{1.55}$$

where, up to a normalizing constant, the α_i^n are the components of the nth eigenvector of the kernel matrix $K_{ij} := (k(x_i, x_j))$.

In a nutshell, this can be understood as follows. To do PCA in $\mathcal{H}$, we wish to find eigenvectors $\mathbf{v}$ and eigenvalues λ of the so-called *covariance matrix* $\mathbf{C}$ in the feature space, where

$$\mathbf{C} := \frac{1}{m} \sum_{i=1}^{m} \Phi(x_i)\Phi(x_i)^\top. \tag{1.56}$$

Here, $\Phi(x_i)^\top$ denotes the transpose of $\Phi(x_i)$ (see Section B.2.1). In the case when $\mathcal{H}$ is very high dimensional, the computational costs of doing this directly are prohibitive. Fortunately, one can show that all solutions to

$$\mathbf{Cv} = \lambda \mathbf{v} \tag{1.57}$$

with $\lambda \neq 0$ must lie in the span of Φ-images of the training data. Thus, we may expand the solution $\mathbf{v}$ as

$$\mathbf{v} = \sum_{i=1}^{m} \alpha_i \Phi(x_i), \tag{1.58}$$

thereby reducing the problem to that of finding the α_i. It turns out that this leads to a dual eigenvalue problem for the expansion coefficients,

Kernel PCA Eigenvalue Problem

$$m\lambda \boldsymbol{\alpha} = K\boldsymbol{\alpha}, \tag{1.59}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^\top$.

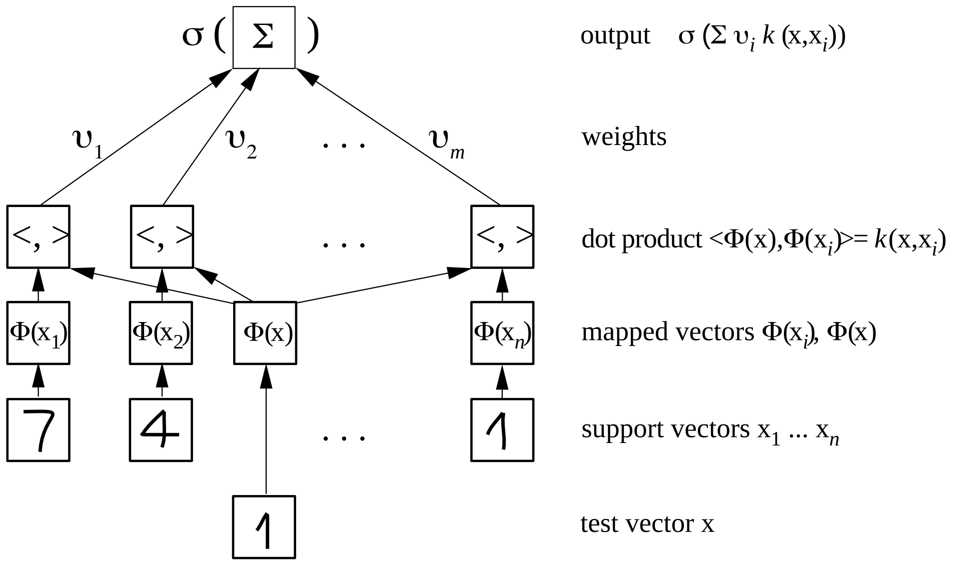
To extract nonlinear features from a test point x, we compute the dot product between $\Phi(x)$ and the nth normalized eigenvector in feature space,

Feature Extraction

$$\langle \mathbf{v}^n, \Phi(x) \rangle = \sum_{i=1}^{m} \alpha_i^n k(x_i, x). \tag{1.60}$$

Usually, this will be computationally far less expensive than taking the dot product in the feature space explicitly.

A toy example is given in Chapter 14 (Figure 14.4). As in the case of SVMs, the architecture can be visualized by Figure 1.9.



σ(Σ)
output σ (Σ υ_i k (x,x_i))
υ_1
υ_2
υ_m
weights
<, >
dot product <Φ(x),Φ(x_i)>= k(x,x_i)
Φ(x_1)
Φ(x_2)
Φ(x)
Φ(x_n)
mapped vectors Φ(x_i), Φ(x)
7
4
1
support vectors x_1 ... x_n
1
test vector x

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 υ_i, found by solving a quadratic program (in pattern recognition, $\upsilon_i = y_i\alpha_i$; in regression estimation, $\upsilon_i = \alpha_i^* - \alpha_i$) 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