where $m \in \{1, \ldots, M\} \setminus y_i$, and $y_i \in \{1, \ldots, M\}$ is the multi-class label of the pattern $\mathbf{x}_i$ (cf. Problem 7.17).

In terms of accuracy, the results obtained with this approach are comparable to those obtained with the widely used one-versus-the-rest approach. Unfortunately, the optimization problem is such that it has to deal with *all* SVs at the same time. In the other approaches, the individual binary classifiers usually have much smaller SV sets, with beneficial effects on the training time. For further multiclass approaches, see [160, 323]. Generalizations to *multi-label* problems, where patterns are allowed to belong to several classes at the same time, are discussed in [162].

Overall, it is fair to say that there is probably no multi-class approach that generally outperforms the others. For practical problems, the choice of approach will depend on constraints at hand. Relevant factors include the required accuracy, the time available for development and training, and the nature of the classification problem (e.g., for a problem with very many classes, it would not be wise to use (7.59)). That said, a simple one-against-the-rest approach often produces acceptable results.

7.7 Variations on a Theme

There are a number of variations of the standard SV classification algorithm, such as the elegant *leave-one-out machine* [589, 592] (see also Section 12.2.2 below), the idea of *Bayes point machines* [451, 239, 453, 545, 392], and extensions to *feature selection* [70, 224, 590]. Due to lack of space, we only describe one of the variations; namely, *linear programming machines*.

Linear Programming Machines

As we have seen above, the SVM approach automatically leads to a decision function of the form (7.25). Let us rewrite it as $f(x) = \operatorname{sgn}(g(x))$, with

$$g(x) = \sum_{i=1}^{m} v_i k(x, x_i) + b. \tag{7.61}$$

In Chapter 4, we showed that this form of the solution is essentially a consequence of the form of the regularizer $\|\mathbf{w}\|^2$ (Theorem 4.2). The idea of linear programming (LP) machines is to use the kernel expansion as an ansatz for the solution, but to use a different regularizer, namely the ℓ_1 norm of the coefficient vector [343, 344, 74, 184, 352, 37, 591, 593, 39]. The main motivation for this is that this regularizer is known to induce sparse expansions (see Chapter 4).

ℓ_1 Regularizer

This amounts to the objective function

$$R_{\text{reg}}[g] := \frac{1}{m}\|\boldsymbol{v}\|_1 + C\, R_{\text{emp}}[g], \tag{7.62}$$

where $\|\boldsymbol{v}\|_1 = \sum_{i=1}^{m} |v_i|$ denotes the ℓ_1 norm in coefficient space, using the soft margin empirical risk,

$$R_{\text{emp}}[g] = \frac{1}{m} \sum_i \xi_i, \tag{7.63}$$

with slack terms

$$\xi_i = \max\{1 - y_i g(x_i), 0\}. \tag{7.64}$$

We thus obtain a linear programming problem;

$$\begin{array}{ll} \underset{\boldsymbol{\alpha},\boldsymbol{\xi}\in\mathbb{R}^m,b\in\mathbb{R}}{\text{minimize}} & \frac{1}{m}\sum_{i=1}^{m}(\alpha_i+\alpha_i^*)+C\sum_{i=1}^{m}\xi_i, \\ \text{subject to} & y_i g(x_i) \geq 1-\xi_i, \\ & \alpha_i, \alpha_i^*, \xi_i \geq 0. \end{array} \tag{7.65}$$

Here, we have dealt with the ℓ_1-norm by splitting each component v_i into its positive and negative part: $v_i = \alpha_i - \alpha_i^*$ in (7.61). The solution differs from (7.25) in that it is no longer necessarily the case that each expansion pattern has a weight $\alpha_i y_i$, whose sign equals its class label. This property would have to be enforced separately (Problem 7.19). Moreover, it is also no longer the case that the expansion patterns lie on or beyond the margin — in LP machines, they can basically be anywhere.

ν-LPMs

LP machines can also benefit from the ν-trick. In this case, the programming problem can be shown to take the following form [212]:

$$\begin{array}{ll} \underset{\boldsymbol{\alpha},\boldsymbol{\xi}\in\mathbb{R}^m,b,\rho\in\mathbb{R}}{\text{minimize}} & \frac{1}{m}\sum_{i=1}^{m}\xi_i - \nu\rho, \\ \text{subject to} & \frac{1}{m}\sum_{i=1}^{m}(\alpha_i+\alpha_i^*)=1, \\ & y_i g(\mathbf{x}_i) \geq \rho - \xi_i, \\ & \alpha_i, \alpha_i^*, \xi_i, \rho \geq 0. \end{array} \tag{7.66}$$

We will not go into further detail at this point. Additional information on linear programming machines from a regularization point of view is given in Section 4.9.2.

7.8 Experiments

7.8.1 Digit Recognition Using Different Kernels

Handwritten digit recognition has long served as a test bed for evaluating and benchmarking classifiers [318, 64, 319]. Thus, it was imperative in the early days of SVM research to evaluate the SV method on widely used digit recognition tasks. In this section we report results on the US Postal Service (USPS) database (described in Section A.1). We shall return to the character recognition problem in Chapter 11, where we consider the larger MNIST database.

As described above, the difference between C-SVC and ν-SVC lies only in the fact that we have to select a different parameter a priori. If we are able to do this