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

Table 7.3 Performance on the USPS set, for three different types of classifier, constructed with the Support Vector algorithm by choosing different functions k in (7.25) and (7.29). Error rates on the test set are given; and for each of the ten-class-classifiers, we also show the average number of Support Vectors of the ten two-class-classifiers. The normalization factor of 256 is tailored to the dimensionality of the data, which is 16×16.

polynomial: $k(x, x') = \left(\langle x, x' \rangle / 256\right)^d$

d	1	2	3	4	5	6	7
raw error/%	8.9	4.7	4.0	4.2	4.5	4.5	4.7
av. # of SVs	282	237	274	321	374	422	491

RBF: $k(x, x') = \exp\left(-\|x - x'\|^2/(256\ c)\right)$

c	4.0	2.0	1.2	0.8	0.5	0.2	0.1
raw error/%	5.3	5.0	4.9	4.3	4.4	4.4	4.5
av. # of SVs	266	240	233	235	251	366	722

sigmoid: $k(x, x') = \tanh(2\langle x, x' \rangle / 256 + \Theta)$

$-\Theta$	0.8	0.9	1.0	1.1	1.2	1.3	1.4
raw error/%	6.3	4.8	4.1	4.3	4.3	4.4	4.8
av. # of SVs	206	242	254	267	278	289	296

well, we obtain identical performance. The experiments reported were carried out before the development of ν-SVC, and thus all use C-SVC code.

In the present study, we put particular emphasis on comparing different types of SV classifiers obtained by choosing different kernels. We report results for polynomial kernels (7.26), Gaussian radial basis function kernels (7.27), and sigmoid kernels (7.28), summarized in Table 7.3. In all three cases, error rates around 4% can be achieved.

Kernel Scaling

Note that in practical applications, it is usually helpful to scale the argument of the kernel, such that the numerical values do not get extremely small or large as the dimension of the data increases. This helps avoid large roundoff errors, and prevents over- and underflow. In the present case, the scaling was done by including the factor 256 in Table 7.3.

The results show that the Support Vector algorithm allows the construction of a range of learning machines, all of which perform well. The similar performance for the three different functions k suggests that among these cases, the choice of the set of decision functions is less important than capacity control in the chosen type of structure. This phenomenon is well-known for the Parzen window density estimator in $\mathbb{R}^N$ (e.g., [226])

$$p(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{\omega^N} k\left(\frac{x - x_i}{\omega}\right). \tag{7.67}$$

It is of great importance in this case to choose an appropriate value of the band-

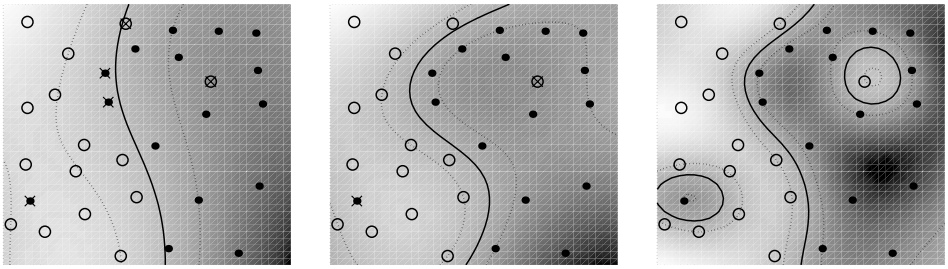

Figure 7.10 2D toy example of a binary classification problem solved using a soft margin SVC. In all cases, a Gaussian kernel (7.27) is used. From left to right, we decrease the kernel width. Note that for a large width, the decision boundary is almost linear, and the data set cannot be separated without error (see text). Solid lines represent decision boundaries; dotted lines depict the edge of the margin (where (7.34) becomes an equality with $\xi_i = 0$).

width parameter ω for a given amount of data. Similar parallels can be drawn to the solution of ill-posed problems; for a discussion, see [561].

Figure 7.10 shows a toy example using a Gaussian kernel (7.27), illustrating that it is crucial to pick the right kernel parameter. In all cases, the same value of C was used, but the kernel width c was varied. For large values of c, the classifier is almost linear, and it cannot separate the data set without errors. For a small width (right), the data set is practically *memorized*. For an intermediate width (middle), a trade-off is made between allowing *some* training errors and using a "simple" decision boundary.

Parameter Choice

In practice, both the kernel parameters and the value of C (or ν) are often chosen using *cross validation*. To this end, we first split the data set into p parts of equal size, say, $p = 10$. We then perform ten training runs. Each time, we leave out one of the ten parts and use it as an independent validation set for optimizing the parameters. In the simplest case, we choose the parameters which work best, on average over the ten runs. It is common practice, however, to then train on the full training set, using these average parameters. There are some problems with this. First, it amounts to optimizing the parameters on the same set as the one used for training, which can lead to overfitting. Second, the optimal parameter settings for data sets of size m and $\frac{9}{10}m$, respectively, do not usually coincide. Typically, the smaller set will require a slightly stronger regularization. This could mean a wider Gaussian kernel, a smaller polynomial degree, a smaller C, or a larger ν. Even worse, it is theoretically possible that there is a so-called phase transition (e.g., [393]) in the learning curve between the two sample sizes. This means that the generalization error as a function of the sample size could change dramatically between $\frac{9}{10}m$ and m. Having said all this, practitioners often do not care about these theoretical precautions, and use the unchanged parameters with excellent results. For further detail, see Section 12.2.

In some cases, one can try to avoid the whole procedure by using an educated guess. Below, we list several methods.

- Use parameter settings that have worked well for similar problems. Here, some care has to be exercised in the scaling of kernel parameters. For instance, when using an RBF kernel, c must be rescaled to ensure that $\|x_i - x_j\|^2/c$ roughly lies in the same range, even if the scaling and dimension of the data are different.
- For many problems, there is some prior expectation regarding the typical error rate. Let us assume we are looking at an image classification task, and we have already tried three other approaches, all of which yielded around 5% test error. Using ν–SV classifiers, we can incorporate this knowledge by choosing a value for ν which is in that range, say $\nu = 5\%$. The reason for this guess is that we know (Proposition 7.5) that the margin error is then below 5%, which in turn implies that the training error is below 5%. The training error will typically be smaller than the test error, thus it is consistent that it should be upper bounded by the 5% test error.
- In a slightly less elegant way, one can try to mimic this procedure for C-SV classifiers. To this end, we start off with a large value of C, and reduce it until the number of Lagrange multipliers that are at the upper bound (in other words, the number of margin errors) is in a suitable range (say, somewhat below 5%). Compared to the above procedure for choosing ν, the disadvantage is that this entails a number of training runs. We can also monitor the number of actual training errors during the training runs, but since not every margin error is a training error, this is often less sensitive. Indeed, the difference between training error and test error can often be quite substantial. For instance, on the USPS set, most of the results reported here were obtained with systems that had essentially zero training error.
- One can put forward scaling arguments which indicate that $C \propto 1/R^2$, where R is a measure for the range of the data in feature space that scales like the length of the points in $\mathcal{H}$. Examples thereof are the standard deviation of the distance of the points to their mean, the radius of the smallest sphere containing the data (cf. (5.61) and (8.17)), or, in some cases, the maximum (or mean) length $k(x_i, x_i)$ over all data points (see Problem 7.25).
- Finally, we can use theoretical tools such as VC bounds (see, for instance, Figure 5.5) or leave-one-out bounds (Section 12.2).

Having seen that different types of SVCs lead to similar performance, the question arises as to how these performances compare with other approaches. Table 7.4 gives a summary of a number of results on the USPS set. Note that the best SVM result is 3.0%; it uses additional techniques that we shall explain in chapters 11 and 13. It is known that the USPS test set is rather difficult — the human error rate is 2.5% [79]. For a discussion, see [496]. Note, moreover, that some of the results reported in the literature for the USPS set were obtained with an enhanced training set: For instance, the study of Drucker et al. [148] used an enlarged training set of size 9709, containing some additional machine-printed digits, and found that this improves the accuracy on the test set. Similarly, Bottou and Vapnik [65] used a training set of size 9840. Since there are no machine-printed digits in the com-

Table 7.4 Summary of error rates on the USPS set. Note that two variants of this database are used in the literature; one of them (denoted by USPS$^+$) is enhanced by a set of machine-printed characters which have been found to improve the test error. Note that the virtual SV systems perform best out of all systems trained on the original USPS set.

Classifier	Training set	Test error	Reference
Linear SVM	USPS	8.9%	[470]
Relevance Vector Machine	USPS	5.1%	Chapter 16
Hard margin SVM	USPS	4.6%	[62]
SVM	USPS	4.0%	[470]
Hyperplane on KPCA features	USPS	4.0%	Chapter 14
KFD	USPS	3.7%	Chapter 15
Virtual SVM	USPS	3.2%	Chapter 11
Virtual SVM, local kernel	USPS	3.0%	Chapter 13
Nearest neighbor	USPS$^+$	5.9%	[496]
LeNet1	USPS$^+$	5.0%	[318]
Local learning Approach	USPS$^+$	3.3%	[65]
Boosted Neural Net	USPS$^+$	2.6%	[148]
Tangent distance	USPS$^+$	2.6%	[496]
Human error rate	—	2.5%	[79]

monly used test set (size 2007), this addition distorts the original learning problem to a situation where results become somewhat hard to interpret. For our experiments, we only had the original 7291 training examples at our disposal. Of all the systems trained on this original set, the SVM system of Chapter 13 performs best.

7.8.2 Universality of the Support Vector Set

In the present section, we report empirical evidence that the SV set contains all the information necessary to solve a given classification task: Using the Support Vector algorithm to train three different types of handwritten digit classifiers, we observe that these types of classifiers construct their decision surface from small, strongly overlapping subsets of the database.

Overlap of SV Sets

To study the Support Vector sets for three different types of SV classifiers, we use the optimal kernel parameters on the USPS set according to Table 7.3. Table 7.5 shows that all three classifiers use around 250 Support Vectors per two-class-classifier (less than 4% of the training set), of which there are 10. The *total* number of different Support Vectors of the ten-class-classifiers is around 1600. It is less than 2500 (10 times the above 250), since for instance a particular vector that has been used as a positive SV (i.e., $y_i = +1$ in (7.25)) for digit 7, might at the same time be a negative SV ($y_i = -1$) for digit 1.

Table 7.6 shows that the SV sets of the different classifiers have about 90% overlap. This surprising result has been reproduced on the MNIST OCR set [467].

Table 7.5 First row: Total number of different SVs in three different ten-class-classifiers (i.e., number of elements of the union of the ten two-class-classifier SV sets), obtained by choosing different functions k in (7.25) and (7.29); Second row: Average number of SVs per two-class-classifier (USPS database size: 7291) (from [470]).

	Polynomial	RBF	Sigmoid
total # of SVs	1677	1498	1611
average # of SVs	274	235	254

Table 7.6 Percentage of the SV set of [column] contained in the SV set of [row]; for ten-class classifiers *(top)*, and binary recognizers for digit class 7 *(bottom)* (USPS set) (from [470]).

	Polynomial	RBF	Sigmoid
Polynomial	100	93	94
RBF	83	100	87
Sigmoid	90	93	100

	Polynomial	RBF	Sigmoid
Polynomial	100	84	93
RBF	89	100	92
Sigmoid	93	86	100

Using a leave-one-out procedure similar to Proposition 7.4, Vapnik and Watkins have put forward a theoretical argument for shared SVs. We state it in the following form: If the SV set of three SV classifiers had no overlap, we could obtain a fourth classifier which has zero test error.

Voting Argument for Shared SVs

To see why this is the case, note that if a pattern is left out of the training set, it will always be classified correctly by voting between the three SV classifiers trained on the remaining examples: Otherwise, it would have been an SV of at least two of them, if kept in the training set. The expectation of the number of patterns which are SVs of at least two of the three classifiers, divided by the training set size, thus forms an upper bound on the expected test error of the voting system. Regarding error rates, it would thus in fact be desirable to be able to construct classifiers with different SV sets. An alternative explanation, studying the effect of the input density on the kernel, was recently proposed by Williams [597]. Finally, we add that the result is also plausible in view of the similar regularization characteristics of the different kernels that were used (see Chapter 4).

Training on SV Sets

As described in Section 7.3, the Support Vector set contains all the information a given classifier needs for constructing the decision function. Due to the overlap in the Support Vector sets of different classifiers, we can even train classifiers on the Support Vector set of *another* classifier; the latter having a different kernel to the former. Table 7.7 shows that this leads to results comparable to those after training

Table 7.7 Training classifiers on the Support Vector sets of other classifiers, leads to performances on the test set (USPS problem) which are as good as the results for training on the full database (number of errors on the 2007-element test set are shown, for two-class classifiers separating digit 7 from the rest). Additionally, the results for training on a random subset of the database of size 200 are displayed.

kernel	trained on: *size:*	poly-SVs *178*	rbf-SVs *189*	tanh-SVs *177*	full db *7291*	rnd. subs. *200*
Poly		13	13	12	13	23
RBF		17	13	17	15	27
tanh		15	13	13	15	25

on the whole database. In Section 11.3, we will use this finding as a motivation for a method to make SVMs transformation invariant, to obtain *virtual SV* machines.

What do these results concerning the nature of Support Vectors tell us? Learning can be viewed as inferring regularities from a set of training examples. Much research has been devoted to the study of various learning algorithms, which allow the extraction of these underlying regularities. No matter how different the outward appearance of these algorithms is, they must all rely on intrinsic regularities of the data. If the learning has been successful, these intrinsic regularities are captured in the values of certain parameters of a learning machine; for a polynomial classifier, these parameters are the coefficients of a polynomial, for a neural network, they are weights, biases, and gains, and for a radial basis function classifier, they are weights, centers, and widths. This variety of different representations of the intrinsic regularities, however, conceals the fact that they all stem from a common root. This is why SVMs with different kernel functions identify the same subset of the training examples as crucial for the regularity to be learned.

7.8.3 Other Applications

SVMs have been successfully applied in other computer vision tasks, which relate to the OCR problems discussed above. Examples include object and face detection and recognition, as well as image retrieval [57, 467, 399, 419, 237, 438, 99, 75].

Another area where SVMs have been used with success is that of *text categorization*. Being a high-dimensional problem, text categorization has been found to be well suited for SVMs. A popular benchmark is the Reuters-22173 text corpus. The news agency Reuters collected 21450 news stories from 1997, and partitioned and indexed them into 135 different categories. The feature typically used to classify Reuters documents are 10^4-dimensional vectors containing word frequencies within a document (sometimes called the "bag-of-words" representation of texts, as it completely discards the information on word ordering). Using this coding, SVMs have led to excellent results, see [155, 265, 267, 150, 333, 542, 149, 326].

Since the use of classification techniques is ubiquitous throughout technology,

we cannot give an exhaustive listing of all successful SVM applications. We thus conclude the list with some of the more exotic applications, such as in High-Energy-Physics [19, 558], in the monitoring of household appliances [390], in protein secondary structure prediction [249], and, with rather intriguing results, in the design of decision feedback equalizers (DFE) in telephony [105].

7.9 Summary

This chapter introduced SV pattern recognition algorithms. The crucial idea is to use kernels to reduce a complex classification task to one that can be solved with separating hyperplanes. We discussed what kind of hyperplane should be constructed in order to get good generalization performance, leading to the idea of large margins. It turns out that the concept of large margins can be justified in a number of different ways, including arguments based on statistical learning theory, and compression schemes. We described in detail how the optimal margin hyperplane can be obtained as the solution of a quadratic programming problem. We started with the linear case, where the hyperplane is constructed in the space of the inputs, and then moved on to the case where we use a kernel function to compute dot products, in order to compute the hyperplane in a feature space.

Two further extensions greatly increase the applicability of the approach. First, to deal with noisy data, we introduced so-called slack variables in the optimization problem. Second, for problems that have more than just two classes, we described a number of generalizations of the binary SV classifiers described initially.

Finally, we reported applications and benchmark comparisons for the widely used USPS handwritten digit task. SVMs turn out to work very well in this field, as well as in a variety of other domains mentioned briefly.

7.10 Problems

7.1 (Weight Vector Scaling •) *Show that instead of the "*1*" on the right hand side of the separation constraint (7.11), we can use any positive number* $\gamma > 0$*, without changing the optimal margin hyperplane solution. What changes in the soft margin case?*

7.2 (Dual Perceptron Algorithm [175] ••) *Kernelize the perceptron algorithm described in footnote 1. Which of the patterns will appear in the expansion of the solution?*

7.3 (Margin of Optimal Margin Hyperplanes [62] ••) *Prove that the geometric margin* ρ *of the optimal margin hyperplane can be computed from the solution* $\boldsymbol{\alpha}$ *via*

$$\rho^{-2} = \sum_{i=1}^{m} \alpha_i. \tag{7.68}$$