8 Single-Class Problems: Quantile Estimation and Novelty Detection

This chapter describes an SV approach to the problem of *novelty detection* and high-dimensional *quantile estimation* [475]. This is an *unsupervised* problem, which can be described as follows. Suppose we are given some dataset drawn from an underlying probability distribution P, and we want to estimate a "simple" subset S of input space, such that the probability that a test point drawn from P lies outside of S equals some a priori specified value between 0 and 1.

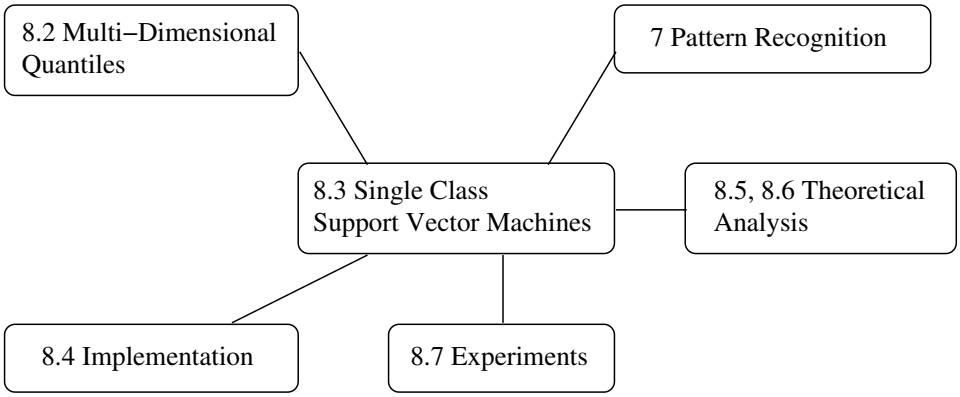
We approach the problem by trying to estimate a function f which is positive on S and negative on the complement. The functional form of f is given by a kernel expansion in terms of SVs; it is regularized by controlling the length of the weight vector in an associated feature space (or, equivalently, by maximizing a margin). The expansion coefficients are found by solving a quadratic programming problem, which can be done by carrying out sequential optimization over pairs of input patterns. We also state theoretical results concerning the statistical performance. The algorithm is a natural extension of the Support Vector classification algorithm, as described in the previous chapter, to the case of unlabelled data.

Overview

The chapter is organized as follows. After a review of some previous work in Section 8.2, taken from [475], we describe SV algorithms for single class problems. Section 8.4 gives details of the implementation of the optimization procedure, specifically for the case of single-class SVMs. Following this, we report theoretical results characterizing the present approach (Section 8.5). In Section 8.7, we deal with the application of the algorithm to artificial and real-world data.

Prerequisites

The prerequisites of the chapter are almost identical to the previous chapter. Those who have read Chapter 7, should be fine with the current chapter. Section 8.2 requires some knowledge of probability theory, as explained in Section B.1; readers who are only interested in the algorithms, however, can skip this slightly more technical section. Likewise, there are some technical parts of Section 8.5 which would benefit from knowledge of Chapter 5, but these can be skipped if desired.



8.1 Introduction

There have been a number of attempts to transfer the idea of using kernels to compute dot products in feature spaces to the domain of unsupervised learning. The problems in this domain are, however, less precisely specified. Generally, they can be characterized as estimating *functions* of the data which tell you something interesting about the underlying distributions. For instance, kernel PCA (Chapter 14) can be described as computing functions which on the training data produce unit variance outputs while having minimum norm in feature space. Another kernel-based unsupervised learning technique, regularized principal manifolds (Chapter 17), computes functions which give a mapping onto a lower-dimensional manifold minimizing a regularized quantization error. Clustering algorithms are further examples of unsupervised learning techniques which can be kernelized [480].

An extreme point of view is that unsupervised learning is about estimating the density of the distribution P generating the data. Knowledge of the density would then allow us to solve whatever problem can be solved on the basis of data *sampled* from that density.

The present chapter addresses an easier problem: it proposes an algorithm that computes a binary function which is supposed to capture regions in input space where the probability density is in some sense large (its support, or, more generally, quantiles); that is, a function which is nonzero in a region where most of the data are located. In doing so, this method is in line with Vapnik's principle never to solve a problem which is more general than the one we actually need to solve [561]. Moreover, it is also applicable in cases where the density of the data's distribution is not even well-defined, as can be the case if the distribution has singular components.

8.2 A Distribution's Support and Quantiles

Quantile Function

In order to describe previous work, it is convenient to introduce the following definition of a (multi-dimensional) quantile function [158]. Let $x_1, \ldots, x_m$ be an iid sample of a random experiment in a set $\mathcal{X}$ with distribution P. Let $\mathcal{C}$ be a class of measurable subsets of $\mathcal{X}$ and let λ be a real-valued function defined on $\mathcal{C}$. The *quantile function* with respect to $(\mathrm{P}, \lambda, \mathcal{C})$ is

$$U(\mu) = \inf\{\lambda(C) | \mathrm{P}(C) \geq \mu, C \in \mathcal{C}\}, \quad 0 < \mu \leq 1. \tag{8.1}$$

Loosely speaking, the quantile function measures how large a set one needs in order to capture a certain amount of probability mass of P.

An interesting special case is the *empirical quantile function*, where P is the empirical distribution

$$\mathrm{P}^m_{\mathrm{emp}}(C) = \frac{1}{m} \sum_{i=1}^{m} I_C(x_i), \tag{8.2}$$

which is the fraction of observations that fall into C.

We denote by $C_\lambda(\mu)$ and $C^m_\lambda(\mu)$ the (not necessarily unique) $C \in \mathcal{C}$ that attain the infimum (when it is achievable). Intuitively speaking, these are the smallest sets (where size is measured by λ) which contain a probability mass μ.

The most common choice of λ is Lebesgue measure (loosely speaking, the *volume* of the set C), in which case $C_\lambda(\mu)$ is the minimum volume set $C \in \mathcal{C}$ that contains at least a fraction μ of the probability mass. Estimators of the form $C^m_\lambda(\mu)$ are called *minimum volume estimators*. Of course, it is not sufficient that the estimated set have a small volume and contain a fraction μ of the training examples. In machine learning applications, we want to find a set that contains a fraction of *test* examples that is close to μ. This is where the complexity trade-off enters (see Figure 8.1), as with the methodology that we have already described in a number of learning scenarios. On the one hand, we want to use a large class $\mathcal{C}$, to ensure that it contains sets C which are very small yet can contain a fraction μ of training examples. On the other hand, if we allowed just *any* set, the chosen set C could consist of only the training points (we would then "memorize" the training points), and it would generalize poorly to test examples; in other words, it would not contain a large probability mass $\mathrm{P}(C)$. Therefore, we have to consider classes of sets which are suitably restricted. As we will see below, this can be achieved using an SVM regularizer.

Support of a Distribution

Observe that for $\mathcal{C}$ being all measurable sets, and λ being the Lebesgue measure, $C_\lambda(1)$ is the *support* of the density p corresponding to P, assuming it exists (note that $C_\lambda(1)$ is well defined even when p does not exist). For smaller classes $\mathcal{C}$, $C_\lambda(1)$ is the minimum volume $C \in \mathcal{C}$ containing the support of p. In the case where $\mu < 1$, it seems the first work was reported in [454, 229], in which $\mathcal{X} = \mathbb{R}^2$, with $\mathcal{C}$ being the class of closed convex sets in $\mathcal{X}$ (they actually considered density contour clusters; cf. [475] for a definition). Nolan [385] considered higher dimensions, with

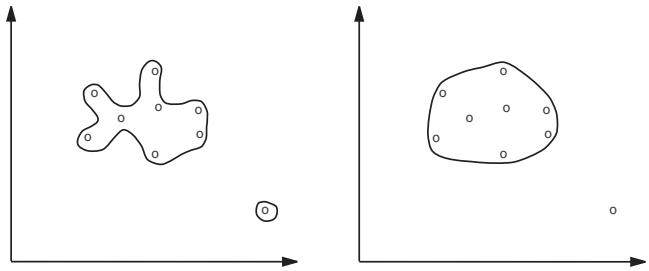


Figure 8.1 A single-class toy problem, with two different solutions. The left graph depicts a rather complex solution, which captures all training points (thus, P^m_{emp} of the estimated region equals 1, cf. (8.2)), while having small volume in $\mathbb{R}^2$. On the right, we show a solution which misses one training point (it does not capture all of P^m_{emp}), but since it is "simpler," it is conceivable that it will nevertheless capture more of the true underlying distribution P that is assumed to have generated the data. In the present context, a function λ is used to measure the simplicity of the estimated region. In the algorithm described below, λ is a SV style regularizer.

$\mathcal{C}$ being the class of ellipsoids. Tsybakov [550] studied an estimator based on piecewise polynomial approximation of $C_\lambda(\mu)$ and showed it attains the asymptotically minimax rate for certain classes of densities. Polonik [417] studied the estimation of $C_\lambda(\mu)$ by $C^m_\lambda(\mu)$. He derived asymptotic rates of convergence in terms of various measures of richness of $\mathcal{C}$. More information on minimum volume estimators can be found in that work, and in [475].

Let us conclude this section with a short discussion of how the present work relates to the above. The present chapter describes an algorithm which finds regions close to $C_\lambda(\mu)$. Our class $\mathcal{C}$ is defined implicitly via a kernel k as the set of half-spaces in a SV feature space. We do not try to minimize the volume of C in input space. Instead, we minimize a SV style regularizer which, using a kernel, controls the smoothness of the estimated function describing C. In terms of multi-dimensional quantiles, the present approach employs $\lambda(C_{\mathbf{w}}) = ||\mathbf{w}||^2$, where $C_{\mathbf{w}} = \{x | f_{\mathbf{w}}(x) \geq \rho\}$, and $(\mathbf{w}, \rho)$ are respectively a weight vector and an offset parametrizing a hyperplane in the feature space associated with the kernel.

8.3 Algorithms

We consider unlabelled training data

$$X = \{x_1, \ldots, x_m\} \subset \mathcal{X}, \tag{8.3}$$

where $m \in \mathbb{N}$ is the number of observations, and $\mathcal{X}$ is some set. For simplicity, we think of it as a compact subset of $\mathbb{R}^N$. Let Φ be a feature map $\mathcal{X} \to \mathcal{H}$; in other words, a map into a dot product space $\mathcal{H}$ such that the dot product in the image

of Φ can be computed by evaluating some simple kernel (Chapters 2 and 4),

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle, \tag{8.4}$$

such as the Gaussian,

$$k(x, x') = e^{-\|x-x'\|^2/c}. \tag{8.5}$$

Indices i and j are understood to range over $1, \ldots, m$ (in compact notation: $i, j \in [m]$). Bold face Greek letters denote m-dimensional vectors whose components are labelled using normal face type.

In the remainder of this section, we shall describe an algorithm which returns a function f that takes the value $+1$ in a "small" region capturing most of the data points, and -1 elsewhere. The strategy, inspired by the previous chapter, is to map the data into the feature space corresponding to the kernel, and to separate them from the origin with maximum margin. For a new point x, the value $f(x)$ is determined by evaluating which side of the hyperplane it falls on, in feature space. Due to the freedom to utilize different types of kernel functions, this simple geometric picture corresponds to a variety of nonlinear estimators in input space.

To separate the data set from the origin, we solve the following quadratic program:

$$\underset{\mathbf{w}\in\mathcal{H}, \boldsymbol{\xi}\in\mathbb{R}^m, \rho\in\mathbb{R}}{\text{minimize}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu m}\sum_i \xi_i - \rho, \tag{8.6}$$

$$\text{subject to } \langle \mathbf{w}, \Phi(x_i) \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0. \tag{8.7}$$

Here, $\nu \in (0, 1]$ is a parameter which is introduced in close analogy to the ν-SV classification algorithm detailed in the previous chapter. Its meaning will become clear later.

Slack Variables

Since nonzero slack variables ξ_i are penalized in the objective function, we can expect that if $\mathbf{w}$ and ρ solve this problem, then the decision function,

$$f(x) = \operatorname{sgn}(\langle \mathbf{w}, \Phi(x) \rangle - \rho), \tag{8.8}$$

will equal 1 for most examples x_i contained in the training set,[1] while the regularization term $\|\mathbf{w}\|$ will still be small. For an illustration, see Figure 8.2. As in ν-SVC (Section 7.5), the trade-off between these two goals is controlled by a parameter ν.

Using multipliers $\alpha_i, \beta_i \geq 0$, we introduce a Lagrangian,

$$L(\mathbf{w}, \boldsymbol{\xi}, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu m}\sum_i \xi_i - \rho - \sum_i \alpha_i(\langle \mathbf{w}, \Phi(x_i) \rangle - \rho + \xi_i) - \sum_i \beta_i \xi_i, \tag{8.9}$$

and set the derivatives with respect to the primal variables $\mathbf{w}, \boldsymbol{\xi}, \rho$ equal to zero, yielding

$$\mathbf{w} = \sum_i \alpha_i \Phi(x_i), \tag{8.10}$$

1. We use the convention that $\operatorname{sgn}(z)$ equals 1 for $z \geq 0$ and -1 otherwise.

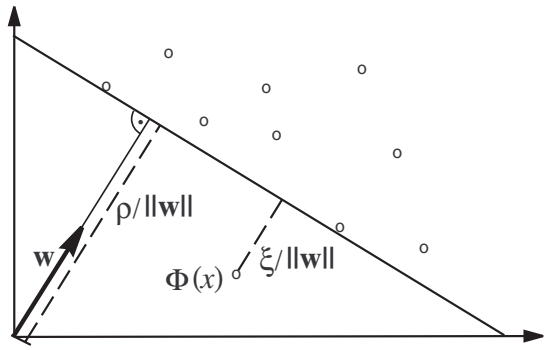


Figure 8.2 In the 2-D toy example depicted, the hyperplane $\langle \mathbf{w}, \Phi(x) \rangle = \rho$ (with normal vector $\mathbf{w}$ and offset ρ) separates all but one of the points from the origin. The outlier $\Phi(x)$ is associated with a slack variable ξ, which is penalized in the objective function (8.6). The distance from the outlier to the hyperplane is $\xi/\|\mathbf{w}\|$; the distance between hyperplane and origin is $\rho/\|\mathbf{w}\|$. The latter implies that a small $\|\mathbf{w}\|$ corresponds to a large margin of separation from the origin.

$$\alpha_i = \frac{1}{\nu m} - \beta_i \leq \frac{1}{\nu m}, \quad \sum_i \alpha_i = 1. \tag{8.11}$$

Eq. (8.10) is the familiar Support Vector expansion (cf. (7.15)). Together with (8.4), it transforms the decision function (8.8) into a kernel expansion,

$$f(x) = \operatorname{sgn}\left(\sum_i \alpha_i k(x_i, x) - \rho\right). \tag{8.12}$$

Single-Class Quadratic Program

Substituting (8.10)–(8.11) into L (8.9), and using (8.4), we obtain the dual problem:

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{minimize}} \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(x_i, x_j), \tag{8.13}$$

$$\text{subject to } 0 \leq \alpha_i \leq \frac{1}{\nu m}, \tag{8.14}$$

$$\sum_i \alpha_i = 1. \tag{8.15}$$

We can show that at the optimum, the two inequality constraints (8.7) become equalities if α_i and β_i are nonzero, which implies $0 < \alpha_i < 1/(\nu m)$ (KKT conditions). Therefore, we can recover ρ by exploiting that for any such α_i, the corresponding pattern x_i satisfies

$$\rho = \langle \mathbf{w}, \Phi(x_i) \rangle = \sum_j \alpha_j k(x_j, x_i). \tag{8.16}$$

Note that if ν approaches 0, the upper bounds on the Lagrange multipliers tend to infinity and the second inequality constraint in (8.14) becomes void. We then have a *hard margin* problem, since the penalization of errors becomes infinite, as can be

seen from the primal objective function (8.6). The problem is still feasible, since we have placed no restriction on the offset ρ, so ρ can become a large negative number in order to satisfy (8.7).

Parzen Windows

It is instructive to compare (8.13)–(8.15) to a Parzen windows estimator (cf. page 6). To this end, suppose we use a kernel which can be normalized as a density in input space, such as the Gaussian (8.5). If we use $\nu = 1$ in (8.14), then the two constraints only allow the solution $\alpha_1 = \ldots = \alpha_m = 1/m$. Thus the kernel expansion in (8.12) reduces to a Parzen windows estimate of the underlying density. For $\nu < 1$, the equality constraint (8.15) still ensures that the decision function is a thresholded density; in that case, however, the density will only be represented by a *subset* of training examples (the SVs) — those which are important for the decision (8.12) to be taken. Section 8.5 will explain the precise meaning of the parameter ν.

To conclude this section, we note that *balls* can also be used to describe the data in feature space, close in spirit to the algorithms described in [470], with hard boundaries, and [535], with "soft margins" (cf. also the algorithm described in Section 5.6). Again, we try to put *most of* the data into a small ball by solving

$$\begin{aligned} &\underset{R\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^m,c\in\mathcal{H}}{\text{minimize}} \; R^2 + \frac{1}{\nu m}\sum_i \xi_i \text{ for } 0 < \nu \leq 1 \\ &\text{subject to } \; \|\Phi(x_i) - c\|^2 \leq R^2 + \xi_i \text{ and } \xi_i \geq 0 \text{ for } i \in [m]. \end{aligned} \tag{8.17}$$

This leads to the dual,

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) - \sum_i \alpha_i k(x_i, x_i), \tag{8.18}$$

$$\text{subject to } \; 0 \leq \alpha_i \leq \frac{1}{\nu m} \text{ and } \sum_i \alpha_i = 1, \tag{8.19}$$

and the solution

$$c = \sum_i \alpha_i \Phi(x_i), \tag{8.20}$$

corresponding to a decision function of the form

$$f(x) = \operatorname{sgn}\left(R^2 - \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) + 2\sum_i \alpha_i k(x_i, x) - k(x, x)\right). \tag{8.21}$$

As above, R^2 is computed such that for any x_i with $0 < \alpha_i < 1/(\nu m)$ the argument of the sgn is zero.

For kernels $k(x, x')$ which only depend on $x - x'$ (the translation invariant kernels, such as RBF kernels), $k(x, x)$ is constant. In this case, the equality constraint implies that the linear term in the dual target function (8.18) is constant, and the problem (8.18–8.19) turns out to be equivalent to (8.13–8.15). It can be shown that the same holds true for the decision function, hence the two algorithms coincide in this case. This is geometrically plausible: for constant $k(x, x)$, all mapped patterns lie on a sphere in feature space. Therefore, finding the smallest ball containing the

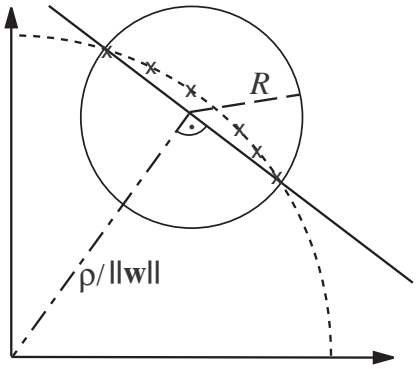


Figure 8.3 For RBF kernels, which depend only on $x - x'$, $k(x, x)$ is constant, and the mapped data points thus lie on a hypersphere in feature space. In this case, finding the smallest sphere enclosing the data is equivalent to maximizing the margin of separation from the origin (cf. Figure 8.2).

points amounts to finding the smallest segment of the sphere on which the data lie. This segment, however, can be found in a straightforward way by simply intersecting the data sphere with a hyperplane — the hyperplane with maximum margin of separation from the origin cuts off the smallest segment (Figure 8.3).

8.4 Optimization

The previous section formulated quadratic programs (QPs) for computing regions that capture a certain fraction of the data. These constrained optimization problems can be solved via an off-the-shelf QP package (cf. Chapter 6). In the present section, however, we describe an algorithm which takes advantage of the precise form of the QPs [475], which is an adaptation of the SMO (Sequential Minimal Optimization) algorithm [409]. Although most of the material on implementations is dealt with in Chapter 10, we will spend a few moments to describe the single class algorithm here. Further information on SMO in general can be found in Section 10.5; additional information on *single-class* SVM implementations, including variants which work in an online setting, can be found in Section 10.6.3.

The SMO algorithm has been reported to work well in C-SV classification. The latter has a structure resembling the present optimization problem: just as the dual of C-SV classification (7.37), the present dual also has only one equality constraint (8.15).[2]

The strategy of SMO is to break up the constrained minimization of (8.13) into the smallest optimization steps possible. Note that it is not possible to modify variables α_i *individually* without violating the sum constraint (8.15). We therefore resort to optimizing over pairs of variables.

2. The ν-SV classification algorithm (7.49), on the other hand, has two equality constraints. Therefore, it is not directly amenable to an SMO approach, unless we remove the equality constraint arising from the offset b, as done in [98].

Elementary SMO Optimization Step

Thus, consider optimizing over two variables α_i and α_j with all other variables fixed. Using the shorthand $K_{ij} := k(x_i, x_j)$, (8.13)–(8.15)) then reduce to (up to a constant)

$$\begin{aligned} \underset{\alpha_i,\alpha_j}{\text{minimize}} \quad & \frac{1}{2}\left[\alpha_i^2 K_{ii} + \alpha_j^2 K_{jj} + 2\alpha_i\alpha_j K_{ij}\right] + c_i\alpha_i + c_j\alpha_j \\ \text{subject to} \quad & \alpha_i + \alpha_j = \gamma \\ & 0 \leq \alpha_i, \alpha_j \leq \tfrac{1}{\nu m} \end{aligned} \tag{8.22}$$

in analogy to (10.63) below. Here the constants c_i, c_j, and γ are defined as follows;

$$c_i := \sum_{l\neq i,j}^{m} \alpha_l K_{il},\ c_j := \sum_{l\neq i,j}^{m} \alpha_l K_{jl}, \text{ and } \gamma = 1 - \sum_{l\neq i,j}^{m} \alpha_l. \tag{8.23}$$

To find the minimum, we use $\alpha_i + \alpha_j = \gamma$. This allows us to obtain a constrained optimization problem in α_i alone by elimination of α_j. For convenience we introduce $\chi := K_{ii} + K_{jj} - 2K_{ij}$.

$$\begin{aligned} \underset{\alpha_i}{\text{minimize}} \quad & \alpha_i^2\chi + \alpha_i\left(c_i - c_j + 2\gamma(K_{ij} - K_{jj})\right) \\ \text{subject to} \quad & L \leq \alpha_i \leq H, \text{ where } L = \max(0, \gamma - 1/(\nu m)) \text{ and } H = \min(1/(\nu m), \gamma). \end{aligned}$$

Without going into details (a similar calculation can be found in Section 10.5.1) the minimizer α_i of this optimization problem is given by

$$\alpha_i = \min(\max(L, \tilde{\alpha}_i), H). \tag{8.24}$$

where $\tilde{\alpha}_i$, the unconstrained solution, is given by

$$\tilde{\alpha}_i = \alpha_i^{\text{old}} + \chi^{-1}\left(c_j - c_i + K_{jj}\alpha_j^{\text{old}} + K_{ij}\left(\alpha_j^{\text{old}} - \alpha_i^{\text{old}}\right) - K_{ii}\alpha_i^{\text{old}}\right) \tag{8.25}$$

$$= \alpha_i^{\text{old}} + \chi^{-1}\left(f^{\text{old}}(x_j) - f^{\text{old}}(x_i)\right). \tag{8.26}$$

Finally, α_j can be obtained via $\alpha_j = \gamma - \alpha_i$. Eq. (8.26) tells us that the change in α_i will depend on the difference between the values $f(x_i)$ and $f(x_j)$. The less close these values are, i.e., the larger the difference in the distances to the hyperplane, the larger the possible change in the set of variables. Note, however, that there is no guarantee that the actual change in α_i will indeed be large, since α_i has to satisfy the constraint $L \leq \alpha_i \leq H$. Finally, the size of χ plays an important role, too (for the case of $\chi = 0$ see Lemma 10.3). The larger it is, the smaller the likely change in α_i.

We next briefly describe how to do the overall optimization.

Initialization of the Algorithm We start by setting a random fraction ν of all α_i to $1/(\nu m)$. If νm is not an integer, then one of the examples is set to a value in $(0, 1/(\nu m))$ to ensure that $\sum_i \alpha_i = 1$. Furthermore, we set the initial ρ to $\max\{f(x_i) | i \in [m],\ \alpha_i > 0\}$.

Optimization Algorithm We then select the first variable for the elementary optimization step in one of two following ways. Here, we use the shorthand SV_{nb} for

the indices of variables which are not at bound (see also Section 10.5.5 for a more detailed description of such a strategy),

$$SV_{\text{nb}} := \{i | i \in [m],\ 0 < \alpha_i < 1/(\nu m)\}. \tag{8.27}$$

These correspond to points which will sit exactly on the hyperplane once optimization is complete, and which will therefore have a strong influence on its precise position.

(i) We scan over the entire data set[3] until we find a variable that violates a KKT condition (Section 6.3.1); in other words, a point such that $(O_i - \rho) \cdot \alpha_i > 0$ or $(\rho - O_i) \cdot (1/(\nu m) - \alpha_i) > 0$. Calling this variable α_i, we pick α_j according to

$$j = \operatorname*{argmax}_{n \in SV_{\text{nb}}} |O_i - O_n|. \tag{8.28}$$

(ii) Same as (i), but the scan is only performed over SV_{nb}.

In practice, one scan of type (i) is followed by multiple scans of type (ii), until there are no KKT violators in SV_{nb}, whereupon the optimization goes back to a single scan of type (i). If the type (i) scan finds no KKT violators, the optimization algorithm terminates.

In unusual circumstances, the choice heuristic (8.28) cannot make positive progress. Therefore, a hierarchy of other choice heuristics is applied to ensure positive progress. These other heuristics are the same as in the case of pattern recognition, cf. Chapter 10 and [409], and were found to work well in the experiments reported below.

Tolerance in the Margin

We conclude this section by stating a trick which is of importance in implementations. In practice, we must use a nonzero accuracy tolerance in tests for equality of numerical quantities. In particular, comparisons of this type are used in determining whether a point lies on the margin. Since we want the final decision function to return 1 for points which lie *on* the margin, we need to subtract this tolerance from ρ at the end.

In the next section, it will be argued that subtracting something from ρ is also advisable from a *statistical* point of view.

8.5 Theory

We now analyze the algorithm theoretically, starting with the uniqueness of the hyperplane (Proposition 8.1). We describe the connection to pattern recognition (Proposition 8.2), and show that the parameter ν characterizes the fractions of SVs and *outliers*. The latter term refers to points which are on the wrong side of

Outlier

3. This scan can be accelerated by not checking patterns which are on the correct side of the hyperplane by a large margin, using the method of [266].

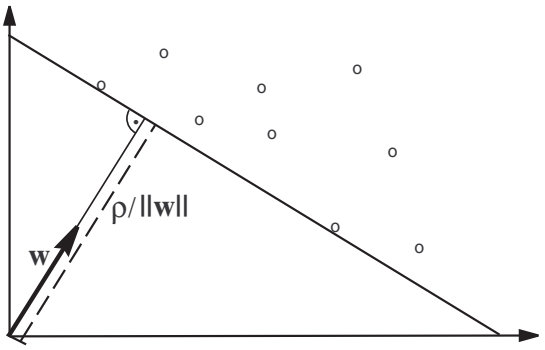


Figure 8.4 A separable data set, with the unique supporting hyperplane separating the data from the origin with maximum margin.

the hyperplane (Proposition 8.3). Following this, we give a robustness result for the soft margin (Proposition 8.4) and we present a theoretical result regarding the generalization error (Theorem 8.6).

As in some of the earlier chapters, we will use boldface letters to denote the feature space images of the corresponding patterns in input space,

$$\mathbf{x}_i := \Phi(x_i). \tag{8.29}$$

We will call a data set

$$\mathbf{X} := \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \tag{8.30}$$

Separable Dataset

separable if there exists some $\mathbf{w} \in \mathcal{H}$ such that $\langle \mathbf{w}, \mathbf{x}_i \rangle > 0$ for $i \in [m]$ (see also Lemma 6.24). If we use a Gaussian kernel (8.5), then any data set $x_1, \ldots, x_m$ is separable after it is mapped into feature space, since in this case, all patterns lie inside the same orthant and have unit length (Section 2.3).

The following proposition is illustrated in Figure 8.4.

Supporting Hyperplane

Proposition 8.1 (Supporting Hyperplane) *If the data set* $\mathbf{X}$ *is separable, then there exists a unique* supporting hyperplane *with the properties that (1) it separates all data from the origin, and (2) its distance to the origin is maximal among all such hyperplanes. For any* $\rho > 0$*, the supporting hyperplane is given by*

$$\underset{\mathbf{w} \in \mathcal{H}}{\text{minimize}} \frac{1}{2}\|\mathbf{w}\|^2 \quad \textit{subject to} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle \geq \rho, \quad i \in [m]. \tag{8.31}$$

Proof Due to the separability, the convex hull of the data does not contain the origin. The existence and uniqueness of the hyperplane then follows from the supporting hyperplane theorem [45, e.g.].

In addition, separability implies that there actually exists some $\rho > 0$ and $\mathbf{w} \in \mathcal{H}$ such that $\langle \mathbf{w}, \mathbf{x}_i \rangle \geq \rho$ for $i \in [m]$ (by rescaling $\mathbf{w}$, this can be seen to work for arbitrarily large ρ). The distance from the hyperplane $\{\mathbf{z} \in \mathcal{H} : \langle \mathbf{w}, \mathbf{z} \rangle = \rho\}$ to the origin is $\rho/\|\mathbf{w}\|$. Therefore the optimal hyperplane is obtained by minimizing $\|\mathbf{w}\|$ subject to these constraints; that is, by the solution of (8.31). ■

The following result elucidates the relationship between single-class classification

and binary classification.

Proposition 8.2 (Connection to Pattern Recognition) *(i) Suppose* $(\mathbf{w}, \rho)$ *parametrizes the supporting hyperplane for the data* $\mathbf{X}$*. Then* $(\mathbf{w}, 0)$ *parametrizes the optimal separating hyperplane for the labelled data set*

$$\{(\mathbf{x}_1, 1), \ldots, (\mathbf{x}_m, 1), (-\mathbf{x}_1, -1), \ldots, (-\mathbf{x}_m, -1)\}. \tag{8.32}$$

(ii) Suppose $(\mathbf{w}, 0)$ *parametrizes the optimal separating hyperplane passing through the origin for a labelled data set*

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}, \quad (y_i \in \{\pm 1\} \text{ for } i \in [m]), \tag{8.33}$$

aligned such that $\langle \mathbf{w}, \mathbf{x}_i \rangle$ *is positive for* $y_i = 1$*. Suppose, moreover, that* $\rho / ||\mathbf{w}||$ *is the margin of the optimal hyperplane. Then* $(\mathbf{w}, \rho)$ *parametrizes the supporting hyperplane for the unlabelled data set* $\mathbf{X}' = \{y_1 \mathbf{x}_1, \ldots, y_m \mathbf{x}_m\}$.

Proof *Ad (i).* By construction, the separation of $\mathbf{X}'$ is a point-symmetric problem. Hence, the optimal separating hyperplane passes through the origin, as if it did not, we could obtain another optimal separating hyperplane by reflecting the first one with respect to the origin — this would contradict the uniqueness of the optimal separating hyperplane.

Next, observe that $(-\mathbf{w}, \rho)$ parametrizes the supporting hyperplane for the data set reflected through the origin, and that it is parallel to that given by $(\mathbf{w}, \rho)$. This provides an optimal separation of the two sets, with distance $2\rho / ||\mathbf{w}||$, and a separating hyperplane $(\mathbf{w}, 0)$.

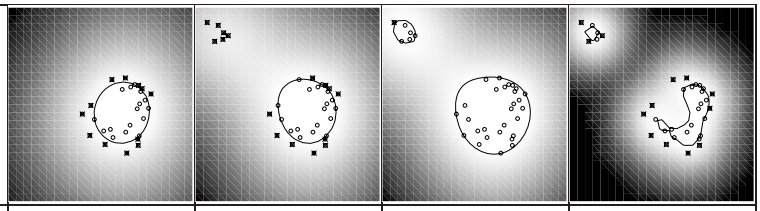
Ad (ii). By assumption, $\mathbf{w}$ is the shortest vector satisfying $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq \rho$ (note that the offset is 0). Hence, equivalently, it is also the shortest vector satisfying $\langle \mathbf{w}, y_i \mathbf{x}_i \rangle \geq \rho$ for $i \in [m]$. ■

Note that the relationship is similar for nonseparable problems. In this case, *margin errors* in binary classification (points which are either on the wrong side of the separating hyperplane or which fall inside the margin) translate into *outliers* in single-class classification, which are points that fall on the wrong side of the hyperplane. Proposition 8.2 then holds, cum grano salis, for the training sets with margin errors and outliers, respectively, removed.

The utility of Proposition 8.2 lies in the fact that it allows us to recycle certain results from binary classification (Chapter 7) for use in the single-class scenario. The following property, which explains the significance of the parameter ν, is such a case.

ν-Property

Proposition 8.3 (ν-Property) *Assume the solution of (8.6),(8.7) satisfies* $\rho \neq 0$*. The following statements hold:*
(i) ν *is an upper bound on the fraction of outliers.*
(ii) ν *is a lower bound on the fraction of SVs.*
(iii) Suppose the data $\mathbf{X}$ *were generated independently from a distribution* $\mathrm{P}(x)$ *which does not contain discrete components. Suppose, moreover, that the kernel is analytic and non-*



ν, width c	0.5, 0.5	0.5, 0.5	0.1, 0.5	0.5, 0.1
frac. SVs/OLs	0.54, 0.43	0.59, 0.47	0.24, 0.03	0.65, 0.38
margin $\rho/\|\mathbf{w}\|$	0.84	0.70	0.62	0.48

Figure 8.5 *First two pictures:* A single-class SVM applied to two toy problems; $\nu = c = 0.5$, domain: $[-1,1]^2$. Note how in both cases, at least a fraction $1-\nu$ of all examples is in the estimated region (cf. table). The large value of ν causes the additional data points in the upper left corner to have almost no influence on the decision function. For smaller values of ν, such as 0.1 *(third picture)*, these points can no longer be ignored. Alternatively, we can force the algorithm to take these 'outliers' (OLs) into account by changing the kernel width (8.5): in the *fourth picture*, using $c = 0.1, \nu = 0.5$, the data are effectively analyzed on a different length scale, which leads the algorithm to consider the outliers as meaningful points.

constant. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of outliers.

The proof can be found in [475]. The result also applies to the soft margin ball algorithm of [535], provided that it is stated in the ν-parametrization given in (8.17). Figure 8.5 displays a 2-D toy example, illustrating how the choice of ν and the kernel width influence the solution.

Resistance

Proposition 8.4 (Resistance) *Local movements of outliers parallel to* $\mathbf{w}$ *do not change the hyperplane.*

Proof **(Proposition 8.4)** Suppose $\mathbf{x}_o$ is an outlier, for which $\xi_o > 0$; hence by the KKT conditions (Chapter 6) $\alpha_o = 1/(\nu m)$. Transforming it into $\mathbf{x}'_o := \mathbf{x}_o + \delta \cdot \mathbf{w}$, where $|\delta| < \xi_o/\|\mathbf{w}\|$, leads to a slack which is still nonzero, $\xi'_o > 0$, hence we still have $\alpha_o = 1/(\nu m)$. Therefore, $\boldsymbol{\alpha}' = \boldsymbol{\alpha}$ is still feasible, as is the primal solution $(\mathbf{w}', \boldsymbol{\xi}', \rho')$. Here, we use $\xi'_i = (1 + \delta \cdot \alpha_o)\xi_i$ for $i \neq o$, $\mathbf{w}' = (1 + \delta \cdot \alpha_o)\mathbf{w}$, and ρ' as computed from (8.16). Finally, the KKT conditions are still satisfied, as $\alpha'_o = 1/(\nu m)$ still holds. Thus (Section 6.3), $\boldsymbol{\alpha}$ remains the solution. ■

Note that although the hyperplane does not change, its parametrization in $\mathbf{w}$ and ρ is different. In single-class SVMs, the hyperplane is not constrained to be in canonical form as it was in SV classifiers (Definition 7.1).

Generalization Error

We now move on to the subject of *generalization*. The goal is to bound the probability that a novel point drawn from the same underlying distribution lies outside of the estimated region. As in the case of pattern recognition, it turns out that the *margin* plays a central role. In the single-class case there is no margin

between the two classes, for the simple reason that there is just one class. We can nevertheless introduce a "safety margin" and make a conservative statement about a slightly larger region than the one estimated. In a sense, this is not so different from pattern recognition: in Theorem 7.3, we try to separate the training data into two half-spaces separated by a margin, and then make a statement about the actual test error (rather than the margin test error); that is, about the probability that a new point will be misclassified, no matter whether it falls inside the margin or not. Just as in the single-class case, the statement regarding the test error thus refers to a region slightly larger than the one in which we try to put the training data.

Definition 8.5 *Let f be a real-valued function on a space $\mathcal{X}$. Fix $\theta \in \mathbb{R}$. For $x \in \mathcal{X}$ let $d(x, f, \theta) = \max\{0, \theta - f(x)\}$. Similarly, for a training sequence $X := (x_1, \ldots, x_m)$, we define*

$$\mathcal{D}(X, f, \theta) = \sum_{x \in X} d(x, f, \theta). \tag{8.34}$$

Theorem 8.6 (Generalization Error Bound) *Suppose we are given a set of m examples $X \in \mathcal{X}^m$, generated from an unknown distribution P that does not have discrete components. Suppose, moreover, that we solve the optimization problem (8.6),(8.7) (or equivalently (8.13)–(8.15)) and obtain a solution $f_{\mathbf{w},\rho}$ given explicitly by (8.12). Let $R_{\mathbf{w},\rho} := \{x | f_{\mathbf{w}}(x) \geq \rho\}$ denote the induced decision region. With probability $1 - \delta$ over the draw of the random sample $X \in \mathcal{X}^m$, for any $\gamma > 0$,*

$$\mathrm{P}\{x' | x' \notin R_{\mathbf{w},\rho-\gamma}\} \leq \frac{2}{m}\left(k + \log_2 \frac{m^2}{2\delta}\right), \tag{8.35}$$

where

$$k = \frac{c_1 \log_2(c_2 \hat{\gamma}^2 m)}{\hat{\gamma}^2} + \frac{2\mathcal{D}}{\hat{\gamma}} \log_2\left(e\left(\frac{(2m-1)\hat{\gamma}}{2\mathcal{D}} + 1\right)\right) + 2, \tag{8.36}$$

$c_1 = 16c^2$, $c_2 = \ln(2)/(4c^2)$, $c = 103$, $\hat{\gamma} = \gamma/\|\mathbf{w}\|$, $\mathcal{D} = \mathcal{D}(X, f_{\mathbf{w},0}, \rho) = \mathcal{D}(X, f_{\mathbf{w},\rho}, 0)$, and ρ is given by (8.16).

The proof can be found in [475].

The training sample X defines (via the algorithm) the decision region $R_{\mathbf{w},\rho}$. We expect that new points generated according to P will lie in $R_{\mathbf{w},\rho}$. The theorem gives a probabilistic guarantee that new points lie in the larger region $R_{\mathbf{w},\rho-\gamma}$.

The parameter ν can be adjusted when running the algorithm to trade off incorporating outliers against minimizing the "size" of $R_{\mathbf{w},\rho}$. Adjusting ν changes the value of $\mathcal{D}$. Note that since $\mathcal{D}$ is measured with respect to ρ while the bound applies to $\rho - \gamma$, any point which is outside of the region to which the bound applies will make a contribution to $\mathcal{D}$ that is bounded away from 0. Therefore, (8.35) does *not* imply that asymptotically, we will always estimate the complete support.

The parameter γ allows us to trade off the confidence with which we wish the assertion of the theorem to hold against the size of the predictive region $R_{\mathbf{w},\rho-\gamma}$:

we can see from (8.36) that k, and hence the rhs of (8.35), scales inversely with γ. In fact, it scales inversely with $\hat{\gamma}$; in other words, it increases with $\mathbf{w}$. This justifies measuring the complexity of the estimated region by the size of $\mathbf{w}$, and minimizing $||\mathbf{w}||^2$ in order to find a region that generalizes well. In addition, the theorem suggests not to use the offset ρ returned by the algorithm, which corresponds to $\gamma = 0$, but a smaller value $\rho - \gamma$ (with $\gamma > 0$).

In the present form, Theorem 8.6 is not a practical means to determine the parameters ν and γ explicitly. It is loose in several ways. The constant c used is far from its smallest possible value. Furthermore, no account is taken of the smoothness of the kernel. If that were done (by using refined bounds on the covering numbers of the induced class of functions, as in Chapter 12), then the first term in (8.36) would increase much more slowly when decreasing γ. The fact that the second term would not change indicates a different trade-off point. Nevertheless, the theorem provides some confidence that ν and γ are suitable parameters to adjust.

8.6 Discussion

Before we move on to experiments, it is worthwhile to discuss some aspects of the algorithm described. As mentioned in the introduction, we could view it as being in line with Vapnik's principle never to solve a problem which is more general than the one that we are actually interested in [561]. For instance, in situations where one is only interested in detecting *novelty*, it is not always necessary to estimate a full density model of the data. Indeed, density estimation is more difficult than what we are doing, in several respects.

Vapnik's Principle

Mathematically speaking, a density only exists if the underlying probability measure possesses an absolutely continuous distribution function. The general problem of estimating the measure for a large class of sets, say the sets measurable in Borel's sense, is not solvable, however (for a discussion, see [562]). Therefore we need to restrict ourselves to making a statement about the measure of *some* sets. Given a small class of sets, the simplest estimator which accomplishes this task is the empirical measure, which simply looks at how many training points fall into the region of interest. The present algorithm does the opposite. It starts with the number of training points that are supposed to fall into the region, and then estimates a region with the desired property. Often, there will be many such regions — the solution becomes unique only by applying a regularizer, which in the SV case enforces that the region be small in a feature space associated with the kernel.

Existence of a Density

Therefore, we must keep in mind that the measure of smallness in this sense depends on the kernel used, in a way that is no different to any other method that regularizes in a feature space. A similar problem, however, already appears in density estimation when done in input space. Let p denote a density on $\mathcal{X}$. If we perform a (nonlinear) coordinate transformation in the input domain $\mathcal{X}$,

then the density values *change*; loosely speaking, what remains constant is $p(x)\,dx$, where dx is also transformed. When directly estimating the probability *measure* of regions, we are not faced with this problem, as the regions automatically change accordingly.

Regularization Interpretation

An attractive property of the measure of smallness that the present algorithm uses is that it can also be placed in the context of regularization theory, leading to an interpretation of the solution as maximally smooth in a sense which depends on the specific kernel used. More specifically, if k is a Green's function of $\Upsilon^*\Upsilon$ for an operator Υ mapping into some dot product space (cf. Section 4.3), then the dual objective function that we minimize equals

$$\sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) = ||\Upsilon f||^2, \tag{8.37}$$

using $f(x) = \sum_i \alpha_i k(x_i, x)$. In addition, we show in Chapter 4 that the regularization operators of common kernels can be shown to correspond to derivative operators — therefore, minimizing the dual objective function has the effect of maximizing the smoothness of the function f (which is, up to a thresholding operation, the function we estimate). This, in turn, is related to a prior with density $p(f) \propto e^{-||\Upsilon f||^2}$ on the function space (cf. Chapter 16).

Interestingly, as the minimization of the dual objective function also corresponds to a maximization of the margin in feature space, an equivalent interpretation is in terms of a prior on the distribution of the unknown other class (the "novel" class in a novelty detection problem) — trying to separate the data from the origin amounts to assuming that the novel examples lie around the origin.

Earlier Work

The main inspiration for the approach described stems from the earliest work of Vapnik and collaborators. In 1962, they proposed an algorithm for characterizing a set of unlabelled data points by separating it from the origin using a hyperplane [573, 570]. However, they quickly moved on to two-class classification problems, both in terms of algorithms and in terms of the theoretical development of statistical learning theory which originated in those days.

From an algorithmic point of view, we can identify two shortcomings of the original approach, which may have caused research in this direction to stop for more than three decades. First, the original algorithm [570] was limited to linear decision rules in input space; second, there was no way of dealing with outliers. In conjunction, these restrictions are indeed severe — a generic dataset need not be separable from the origin by a hyperplane in input space. The two modifications that the single-class SVM incorporates dispose of these shortcomings. First, the kernel trick allows for a much larger class of functions by nonlinearly mapping into a high-dimensional feature space, and thereby increases the chances of a separation from the origin being possible. In particular, using a Gaussian kernel (8.5), such a separation is always possible, as shown in Section 8.5. The second modification directly allows for the possibility of outliers. This 'softness' of the decision rule is incorporated using the ν-trick, which we have already seen in the classification case (Section 7.5), leading to a direct handle on the fraction of

outliers.

Combinatorial Problem

Given $\nu \in (0,1]$, the resulting algorithm computes (8.6) subject to (8.7), and thereby constructs a region R such that for $OL = \{i : x_i \notin R\}$, we have $\frac{|OL|}{m} \leq \nu$. The "$\leq$" is *sharp* in the sense that if we multiply the solution $\mathbf{w}$ by $(1-\epsilon)$ (with $\epsilon > 0$), it becomes a "$>$." The algorithm does *not* solve the following *combinatorial* problem, however: given $\nu \in (0,1]$, compute

$$\begin{aligned} &\underset{\mathbf{w}\in\mathcal{H}, OL\subset[m]}{\text{minimize}} \ \frac{1}{2}\|\mathbf{w}\|^2, \\ &\text{subject to} \ \ \langle \mathbf{w}, \Phi(x_i)\rangle \geq 1 \text{ for } i \in [m] \setminus OL \text{ and } \frac{|OL|}{m} = \nu. \end{aligned} \tag{8.38}$$

Ben-David et al. [31] analyze a problem related to (8.38): they consider a sphere (which for some feature spaces is equivalent to a half-space, as shown in Section 8.3), fix its radius, and attempt to find its center such that it encloses as many points as possible. They prove that it is already NP hard to approximate the maximal number to within a factor smaller than $3/418$.

Kernel-Based Vector Quantization

We conclude this section by mentioning another kernel-based algorithm that has recently been proposed for the use on unlabelled data [541]. This algorithm applies to vector quantization, a standard process which finds a codebook such that the training set can be approximated by elements of the codebook with small error. Vector quantization is briefly described in Example 17.2 below; for further detail, see [195].

Given some metric d, the kernel-based approach of [541] uses a kernel that indicates whether two points lie within a distance $R \geq 0$ of each other,

$$k(x, x') = I_{\{(x,x')\in\mathcal{X}\times\mathcal{X}: d(x,x')\leq R\}}. \tag{8.39}$$

Let Φ_m be the empirical kernel map (2.56) with respect to the training set. The main idea is that if we can find a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ such that

$$\boldsymbol{\alpha}^\top \Phi_m(x_i) > 0 \tag{8.40}$$

holds true for all $i = 1, \ldots, m$, then each point x_i lies within a distance R of some point x_j which has a positive weight $w_j > 0$. To see this, note that otherwise all nonzero components of $\boldsymbol{\alpha}$ would get multiplied by components of Φ_m which are 0, and the dot product in (8.40) would equal 0.

To perform vector quantization, we can thus use optimization techniques, which produce a vector $\boldsymbol{\alpha}$ that satisfies (8.40) while being sparse. As in Section 7.7, this can be done using linear programming techniques. Once optimization is complete, the nonzero entries of $\boldsymbol{\alpha}$ indicate the codebook vectors.

8.7 Experiments

We apply the method to artificial and real-world data. Figure 8.6 shows a comparison with a Parzen windows estimator on a 2-D problem, along with a family of

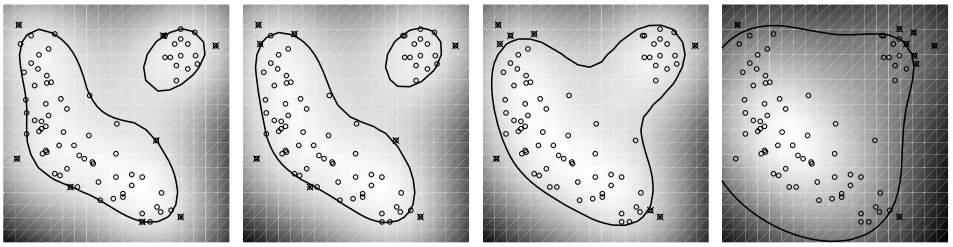


Figure 8.6 A single-class SVM applied to a toy problem; $c = 0.5$, domain: $[-1,1]^2$, for various settings of the offset ρ. As discussed in Section 8.3, $\nu = 1$ yields a Parzen windows expansion. However, to get a Parzen windows estimator of the distribution's support, we must not use the offset returned by the algorithm (which would allow *all* points to lie outside the estimated region). Therefore, in this experiment, we adjusted the offset such that a fraction $\nu' = 0.1$ of patterns lie outside. From left to right, we show the results for $\nu \in \{0.1, 0.2, 0.4, 1\}$. The rightmost picture corresponds to the Parzen estimator which utilizes all kernels; the other estimators use roughly a fraction ν of kernels. Note that as a result of the averaging over all kernels, the Parzen windows estimate does not model the shape of the distribution very well for the chosen parameters.

estimators which lie "in between" the present one and the Parzen one.

Figure 8.7 shows a plot of the outputs $\langle \mathbf{w}, \Phi(x) \rangle$ on training and test sets of the USPS database of handwritten digits (Section A.1). We used a Gaussian kernel (8.5), which has the advantage that the data are always separable from the origin in feature space (Section 2.3). For the kernel parameter c, we used $0.5 \cdot 256$. This value was chosen a priori, and is a common value for SVM classifiers on that data set, cf. Chapter 7.[4] The algorithm was given only the training instances of digit 0. Testing was done on both digit 0 and on all other digits. We present results for two values of ν, one large, one small; for values in between, the results are qualitatively similar. In the first experiment, we used $\nu = 50\%$, thus aiming for a description of "0-ness" which only captures half of all zeros in the training set. As shown in figure 8.7, this leads to *zero* false positives (i.e., even though the learning machine has not seen any non-0s during training, it correctly identifies all non-0s as such), while still recognizing 44% of the digits 0 in the *test* set. Higher recognition rates can be achieved using smaller values of ν. For $\nu = 5\%$, we get 91% correct recognition of digits 0 in the test set, with a fairly moderate false positive rate of 7%.

Although leading to encouraging results, this experiment does not really address the actual task the algorithm was designed for. Therefore, we next focus on a problem of *novelty detection*. Again, we utilized the USPS set; this time, however, we trained the algorithm on the test set and used it to identify outliers — it is well known that the USPS test set (Figure 8.8) contains a number of patterns which

4. In [236], the following procedure is used to determine a value of c. For small c, all training points become SVs — the algorithm just memorizes the data, and will not generalize well. As c increases, the number of SVs drops. As a simple heuristic, we can thus start with a small value of c and increase it until the number of SVs does not decrease any further.

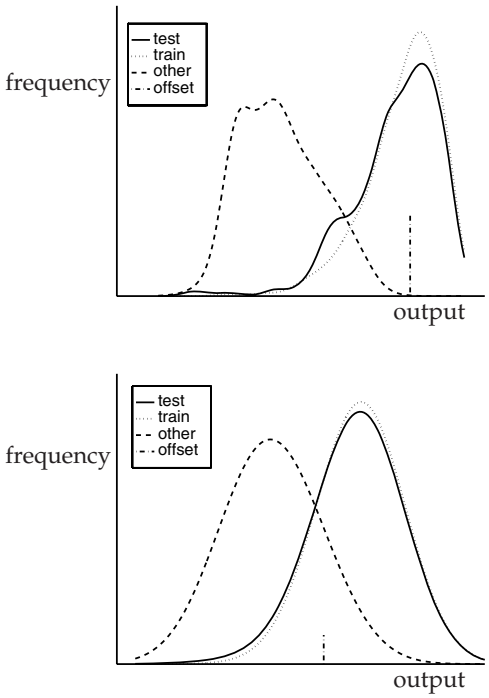


Figure 8.7 Experiments using the USPS OCR dataset. Recognizer for digit 0; output histogram for the exemplars of 0 in the training/test set, and on test exemplars of other digits. The x-axis gives the output values; in other words, the argument of the sgn function in (8.12). For $\nu = 50\%$ *(top)*, we get 50% SVs and 49% outliers (consistent with Proposition 8.3), 44% true positive test examples, and zero false positives from the "other" class. For $\nu = 5\%$ *(bottom)*, we get 6% and 4% for SVs and outliers, respectively. In that case, the true positive rate is improved to 91%, while the false positive rate increases to 7%. The offset ρ is marked in the graphs.
Note, finally, that the plots show a Parzen windows density estimate of the output histograms. In reality, many examples sit exactly at the offset value (the non-bound SVs). Since this peak is smoothed out by the estimator, the fractions of outliers in the training set appear slightly larger than they should be.

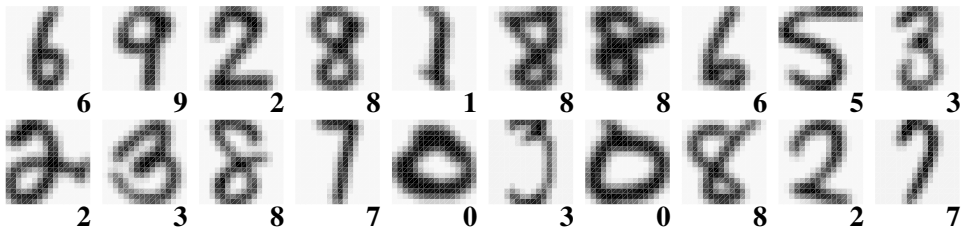


Figure 8.8 20 examples randomly drawn from the USPS test set, with class labels.

are hard or impossible to classify, due to segmentation errors or mislabelling. In this experiment, we augmented the input patterns by ten extra dimensions corresponding to the class labels of the digits. The rationale for this is that if we disregard the labels, there would be no hope of identifying *mislabelled* patterns as outliers. With the labels, the algorithm has the chance to identify both unusual patterns *and* usual patterns with unusual labels. Figure 8.9 shows the 20 worst outliers for the USPS test set, respectively. Note that the algorithm indeed extracts patterns which are very hard to assign to their respective classes. In the experiment, we used the same kernel width as above, and a ν value of 5%. The latter was chosen to roughly reflect the expectation as to how many "bad" patterns there are in the test set: most good learning algorithms achieve error rates of 3 - 5% on the USPS benchmark (for a list of results, cf. Table 7.4).

Table 8.1 Experimental results for various values of the outlier control constant ν, USPS test set, size $m = 2007$. Note that ν bounds the fractions of outliers and Support Vectors from above and below, respectively (cf. Proposition 8.3). As we are not in the asymptotic regime, there is some slack in the bounds; nevertheless, ν can be used to control these fractions. Note, moreover, that training times (CPU time in seconds on a Pentium II running at 450 MHz) increase as ν approaches 1. This is related to the fact that almost all Lagrange multipliers are at the upper bound in that case (cf. Section 8.4). The system used in the outlier detection experiments is shown in boldface.

ν	fraction of OLs	fraction of SVs	training time
1%	0.0%	10.0%	36
3%	0.1%	10.0%	31
5%	**1.4%**	**10.6%**	**36**
10%	6.2%	13.7%	65
30%	27.5%	31.8%	269
50%	47.4%	51.2%	1284
70%	68.3%	70.7%	1512
90%	89.4%	90.1%	2349

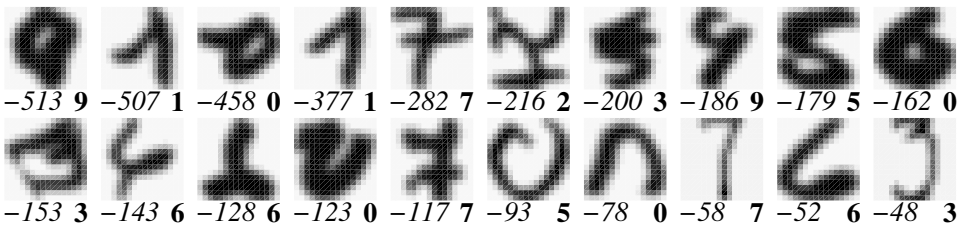


Figure 8.9 Outliers identified by the proposed algorithm, ranked by the negative output of the SVM (the argument of (8.12)). The outputs (for convenience in units of 10^{-5}) are written underneath each image in italics, the (alleged) class labels are given in bold face. Note that most of the examples are "difficult" in that they are either atypical or mislabelled.

In the last experiment, we tested the runtime scaling behavior of the SMO solver used for training (Figure 8.10). Performance was found to depend on ν. For the small values of ν which are typically used in outlier detection tasks, the algorithm scales very well to larger data sets, with a dependency of training times on the sample size which is at most quadratic.

In addition to the experiments reported above, the present algorithm has since been applied in several other domains, such as the modelling of parameter regimes for the control of walking robots [528], condition monitoring of jet engines [236], and hierarchical clustering problems [35].

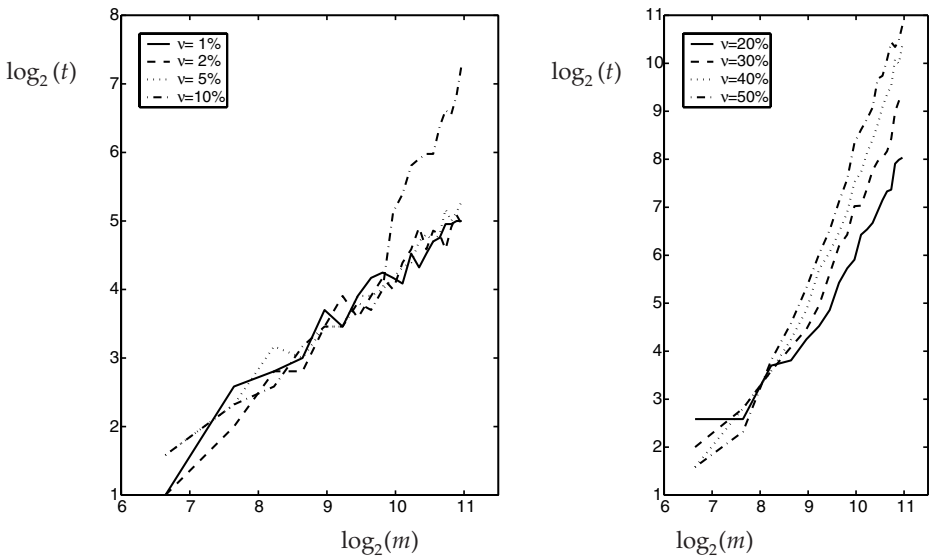


Figure 8.10 Training times t (in seconds) vs. data set sizes m (both axes depict logs at base 2; CPU time in seconds on a Pentium II running at 450 MHz, training on subsets of the USPS test set); $c = 0.5 \cdot 256$. As in Table 8.1, it can be seen that larger values of ν generally lead to longer training times (note that the plots use different y-axis ranges). Times also differ in their scaling with the sample size, however. The exponents can be directly read from the slope of the graphs, as they are plotted in log scale with equal axis spacing: for small values of ν ($\leq 5\%$), the training times are approximately linear in the training set size (left plot). The scaling gets worse as ν increases. For large values of ν, training times are roughly proportional to the sample size raised to the power of 2.5 (right plot). The results should be taken only as an indication of what is going on: they were obtained using fairly small training sets; the largest being 2007, the size of the USPS test set. As a consequence, they are fairly noisy, and strictly speaking, they refer only to the examined regime. Encouragingly, the scaling is better than the cubic scaling that we would expect when solving the optimization problem using all patterns at once, cf. Section 8.4. Moreover, for *small* values of ν, which are typically used in outlier detection (in Figure 8.9, we used $\nu = 5\%$), the algorithm is particularly efficient.

8.8 Summary

In the present chapter, we described SV algorithms that can be applied to unlabelled data. Statistically speaking, these are a solution to the problem of multidimensional quantile estimation. Practically speaking, they provide a "description" of the dataset that can thought of as a "single-class" classifier, which can tell whether a new point is likely to have been generated by the same regularity as the training data. This makes them applicable to problems such as novelty detection.

We began with a discussion of the quantile estimation problem, which led us to a quantile estimation algorithm, which is very similar to the SV classification algorithm in that it uses the same type of large margin regularizer. To deal with outliers in the data, we made use of the ν-trick introduced in the previous chapter. We described an SMO-style optimization problem to compute the unique solution

of the algorithm, and we provided a theoretical analysis of a number of aspects, such as robustness with respect to outliers, and generalization to unseen data.

Finally, we described outlier detection experiments using real-world data. It is worthwhile to note at this point that single-class approaches have abundant practical applications. Suggestions include information retrieval, problems in medical diagnosis [534], marketing [33], condition monitoring of machines [138], estimating manufacturing yields [531], econometrics and generalized nonlinear principal curves [550, 308], regression and spectral analysis [417], tests for multimodality and clustering [416] and others [374]. Many of these papers, in particular those with a theoretical slant [417, 114, e.g.], do not go all the way in devising practical algorithms that work on high-dimensional real-world-problems, a notable exception being the work of Tarassenko et al. [534]. The single-class SVM described in this chapter constitutes a practical algorithm with well-behaved computational complexity (convex quadratic programming) for these problems.

8.9 Problems

8.1 (Uniqueness of the Supporting Hyperplane ••) *Using Lemma 6.24 (cf. Figure 6.7), prove that if the convex hull of the training data does not contain the origin, then there exists a unique hyperplane with the properties that (1) it separates all data from the origin, and (2) its distance to the origin is maximal among all such hyperplanes. (cf. Proposition 8.1).*

Hint: argue that you can limit yourself to finding the maximum margin of separation over all weight vectors of length $||\mathbf{w}|| \leq 1$*; that is, over a convex set.*

8.2 (Soft Ball Algorithm [535] ••) *Extend the hard ball algorithm described in Section 5.6 to the soft margin case; in other words, derive (8.18) and (8.19) from (8.17).*

8.3 (Hard Margin Limit for Positive Offsets •) *Show that if we require* $\rho \geq 0$ *in the primal problem, we end up with the constraint* $\sum_i \alpha_i \geq 1$ *instead of* $\sum_i \alpha_i = 1$ *(see (8.15)). Consider the hard margin limit* $\nu \to 0$ *and argue that in this case, the problem can become infeasible, and the multipliers* α_i *can diverge during the optimization. Give a geometric interpretation of why this happens for* $\rho \geq 0$*, but not if* ρ *is free.*

8.4 (Positivity of ρ ••) *Prove that the solution of (8.6) satisfies* $\rho \geq 0$*.*

8.5 (Hard Margin Identities ••) *Consider the hard margin optimization problem, consisting of minimizing* $||\mathbf{w}||^2$ *subject to* $\langle \mathbf{w}, \mathbf{x}_i \rangle \geq \rho$ *for* $i \in [m]$ *(here,* $\rho > 0$ *is a constant). Prove that the following identities hold true for the solution* $\boldsymbol{\alpha}$*:*

$$\frac{\rho^2}{d^2} = ||\mathbf{w}||^2 = 2W(\boldsymbol{\alpha}) \tag{8.41}$$

Here, $W(\boldsymbol{\alpha})$ *denotes the value of the dual objective function, and* d *is the distance of the hyperplane to the origin. Hint: use the KKT conditions and the primal constraints.*

8.6 (ν-Property for Single-Class SVMs [475] ••) *Prove Proposition 8.3, using the techniques from the proof of Proposition 7.5.*

8.7 (Graphical Proof of ν-Property •) *Give a graphical proof of the first two statements of Proposition 8.3 along the lines of Figure 9.5.*

8.8 (ν-Property for the Soft Ball Algorithm ••) *Prove Proposition 8.3 for the soft ball algorithm (8.18), (8.19).*

8.9 (Multi-class SV Classification ••) *Implement the single-class algorithm and use it to solve a multi-class classification problem by training a single-class recognizer on each class. Discuss how to best compare the outputs of the individual recognizers in order to get to a multi-class decision.*

8.10 (Negative Data [476] ••) *Derive a variant of the single-class algorithm that can handle negative data by reflecting the negative points at the origin in feature space, and then solving the usual one-class problem. Show that this leads to the following problem:*

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{minimize}} \frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j k(x_i,x_j), \tag{8.42}$$

$$\text{subject to } 0 \le \alpha_i \le \frac{1}{\nu m}, \tag{8.43}$$

$$\sum_i \alpha_i = 1 \tag{8.44}$$

Argue, moreover, that the decision function takes the form

$$f(x) = \operatorname{sgn}\left(\sum_i \alpha_i y_i k(x_i,x) - \rho\right), \tag{8.45}$$

and that ρ can be computed by exploiting that for any x_i such that $\alpha_i \in (0, 1/(\nu m))$,

$$\rho = \langle \mathbf{w}, \Phi(x_i)\rangle = \sum_j \alpha_j y_j k(x_j, x_i). \tag{8.46}$$

Show that the algorithm (8.13) is a special case of the above one. Discuss the connection to the SVC algorithm (Chapter 7), in particular with regard to how the two algorithms deal with unbalanced data sets.

8.11 (Separation from General Points [476] ••) *Derive a variant of the single-class algorithm that, rather than separating the points from the origin in feature space, separates them from the mean of some other set of points $\Phi(z_1), \ldots, \Phi(z_t)$. Argue that this lets you take into account the unknown "other" class in a "weak" sense. Hint: modify the first constraint in (8.7) to*

$$\left\langle \mathbf{w}, \Phi(x_i) - \frac{1}{t}\sum_{n=1}^{t}\Phi(z_n)\right\rangle \ge \rho - \xi_i, \tag{8.47}$$

and the decision function to

$$f(x) = \operatorname{sgn}\left(\left\langle \mathbf{w}, \Phi(x) - \frac{1}{t}\sum_{n=1}^{t}\Phi(z_n)\right\rangle - \rho\right). \tag{8.48}$$

Prove that the dual problem takes the following form:

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{minimize}}\ \frac{1}{2}\sum_{ij}\alpha_i\alpha_j\left(k(x_i,x_j)+q-q_j-q_i\right), \tag{8.49}$$

$$\text{subject to}\ \ 0 \le \alpha_i \le \frac{1}{\nu m}, \tag{8.50}$$

$$\sum_i \alpha_i = 1, \tag{8.51}$$

where $q = \frac{1}{t^2}\sum_{np} k(z_n, z_p)$ *and* $q_j = \frac{1}{t}\sum_n k(x_j, z_n)$.

Discuss the special case where you try to separate a data set from its own *mean and argue that this provides a method for computing one-sided quantiles with large margin.*

8.12 (Cross-Validation •) *Discuss the question of how to validate whether a single-class SVM generalizes well. What are the main differences with the problem of pattern recognition?*

8.13 (Leave-One-Out Bound ••) *Using Theorem 12.9, prove that the generalization error of single-class SVMs trained on samples of size* $m - 1$ *is bounded by the number of SVs divided by* m.

Note that this bound makes a statement about the case where $\gamma = 0$ *(cf. Theorem 8.6). Argue that this can cause the bound to be loose. Compare the case of pattern recognition, and argue that the usual leave-one-out bound is also loose there, since it makes a statement about the probability of a test point being misclassified* or *lying inside the margin.*

8.14 (Kernel-Dependent Generalization Error Bounds ○○○) *Modify Theorem 8.6 to take into account properties of the kernel along the lines of the entropy number methods described in Chapter 12.*

8.15 (Model Selection ○○○) *Try to come up with principled model selection methods for single-class SVMs. How would you recommend to choose* ν *and the kernel parameter? How would you choose* γ *(Theorem 8.6)?*