7

Pattern analysis using convex optimisation

This chapter presents a number of algorithms for particular pattern analysis tasks such as novelty-detection, classification and regression. We consider criteria for choosing particular pattern functions, in many cases derived from stability analysis of the corresponding tasks they aim to solve. The optimisation of the derived criteria can be cast in the framework of convex optimization, either as linear or convex quadratic programs. This ensures that as with the algorithms of the last chapter the methods developed here do not suffer from the problem of local minima. They include such celebrated methods as support vector machines for both classification and regression.

We start, however, by describing how to find the smallest hypersphere containing the training data in the embedding space, together with the use and analysis of this algorithm for detecting anomalous or novel data. The techniques introduced for this problem are easily adapted to the task of finding the maximal margin hyperplane or support vector solution that separates two sets of points again possibly allowing some fraction of points to be exceptions. This in turn leads to algorithms for the case of regression.

An important feature of many of these systems is that, while enforcing the learning biases suggested by the stability analysis, they also produce 'sparse' dual representations of the hypothesis, resulting in efficient algorithms for both training and test point evaluation. This is a result of the Karush–Kuhn–Tucker conditions, which play a crucial role in the practical implementation and analysis of these algorithms.

7.1 The smallest enclosing hypersphere

In Chapter 1 novelty-detection was cited as one of the pattern analysis algorithms that we aimed to develop in the course of this book. A novelty-detection algorithm uses a training set to learn the support of the distribution of the 'normal' examples. Future test examples are then filtered by the resulting pattern function to identify any abnormal examples that appear not to have been generated from the same training distribution.

In Chapter 5 we developed a simple novelty-detection algorithm in a general kernel-defined feature space by estimating when new data is outside the hypersphere around the centre of mass of the distribution with radius large enough to contain all the training data. In this section we will further investigate the use of feature space hyperspheres as novelty detectors, where it is understood that new examples that lie outside the hypersphere are treated as 'abnormal' or 'novel'.

Clearly the smaller the hypersphere the more finely tuned the novelty-detection that it realises. Hence, our aim will be to define smaller hyperspheres for which we can still guarantee that with high probability they contain most of the support of the training distribution. There are two respects in which the novelty-detection hypersphere considered in Chapter 5 may be larger than is necessary. Firstly, the centre of the hypersphere was fixed at the centre of mass, or an estimate thereof, based on the training data. By allowing its centre to move it may be possible to find a smaller hypersphere that still contains all the training data. The second concern is that just one unlucky training example may force a much larger radius than should really be needed, implying that the solution is not robust. Ideally we would therefore like to find the smallest hypersphere that contains all but some small proportion of extreme training data.

Given a set of data embedded in a space, the problem of finding the smallest hypersphere containing a specified non-trivial fraction of the data is unfortunately NP-hard. Hence, there are no known algorithms to solve this problem exactly. It can, however, be solved exactly for the case when the hypersphere is required to include all of the data. We will therefore first tackle this problem. The solution is of interest in its own right, but the techniques developed will also indicate a route towards an approximate solution for the other case. Furthermore, the approach adopted for novelty-detection points the way towards a solution of the classification problem that we tackle in Section 7.2.

7.1.1 The smallest hypersphere containing a set of points

Let us assume that we are given a training set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ with an associated embedding $\boldsymbol{\phi}$ into a Euclidean feature space F with associated kernel κ satisfying

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z}) \rangle .$$

The centre of the smallest hypersphere containing S is the point $\mathbf{c}$ that minimises the distance r from the furthest datapoint, or more precisely

$$\mathbf{c}^* = \operatorname*{argmin}_{\mathbf{c}} \max_{1 \le i \le \ell} \|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\| ,$$

with R the value of the expression at the optimum. We have derived the following computation.

Computation 7.1 [Smallest enclosing hypersphere] Given a set of points

$$S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$$

the hypersphere $(\mathbf{c},r)$ that solves the optimisation problem

$$\begin{array}{ll} \min_{\mathbf{c},r} & r^2 \\ \text{subject to} & \|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\|^2 = (\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c})'(\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}) \le r^2 \\ & i = 1, \ldots, \ell, \end{array} \tag{7.1}$$

is the hypersphere containing S with smallest radius r. ■

We can solve constrained optimisation problems of this type by defining a Lagrangian involving one Lagrange multiplier $\alpha_i \ge 0$ for each constraint

$$L(\mathbf{c}, r, \boldsymbol{\alpha}) = r^2 + \sum_{i=1}^{\ell} \alpha_i \left[\|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\|^2 - r^2 \right] .$$

We then solve by setting the derivatives with respect to $\mathbf{c}$ and r equal to zero

$$\begin{aligned} \frac{\partial L(\mathbf{c}, r, \boldsymbol{\alpha})}{\partial \mathbf{c}} &= 2 \sum_{i=1}^{\ell} \alpha_i (\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}) = \mathbf{0}, \text{ and} \\ \frac{\partial L(\mathbf{c}, r, \boldsymbol{\alpha})}{\partial r} &= 2r \left(1 - \sum_{i=1}^{\ell} \alpha_i \right) = 0, \end{aligned}$$

giving the following equations

$$\sum_{i=1}^{\ell} \alpha_i = 1 \quad \text{and as a consequence} \quad \mathbf{c} = \sum_{i=1}^{\ell} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) .$$

The second equality implies that the centre of the smallest hypersphere containing the datapoints always lies in their span. This shows that the centre can be expressed in the dual representation. Furthermore, the first equality implies that the centre lies in the convex hull of the training set. Inserting these relations into the Lagrangian we obtain

$$\begin{aligned}
L(\mathbf{c}, r, \boldsymbol{\alpha}) &= r^2 + \sum_{i=1}^{\ell} \alpha_i \left[\|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\|^2 - r^2 \right] \\
&= \sum_{i=1}^{\ell} \alpha_i \langle \boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}, \boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c} \rangle \\
&= \sum_{i=1}^{\ell} \alpha_i \left(\kappa(\mathbf{x}_i, \mathbf{x}_i) + \sum_{k,j=1}^{\ell} \alpha_j \alpha_k \kappa(\mathbf{x}_j, \mathbf{x}_k) - 2 \sum_{j=1}^{\ell} \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right) \\
&= \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) + \sum_{k,j=1}^{\ell} \alpha_k \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_k) - 2 \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\
&= \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_k),
\end{aligned}$$

where we have used the relation $\sum_{i=1}^{\ell} \alpha_i = 1$ to obtain line 2 and to take the middle expression out of the brackets after line 3. The Lagrangian has now been expressed wholly in terms of the Lagrange parameters, something referred to as the *dual Lagrangian*. The solution is obtained by maximising the resulting expression. We have therefore shown the following algorithm, where we use $\mathcal{H}$ to denote the Heaviside function $\mathcal{H}(x) = 1$, if $x \geq 0$ and 0 otherwise.

Algorithm 7.2 [Smallest hypersphere enclosing data] The smallest hypersphere in a feature space defined by a kernel κ enclosing a dataset S is computed given in Code Fragment 7.1. ■

We have certainly achieved our goal of decreasing the size of the hypersphere since now we have located the hypersphere of minimal volume that contains the training data.

Remark 7.3 [On sparseness] The solution obtained here has an additional important property that results from a theorem of optimization known as the Kuhn-Tucker Theorem. This theorem states that the Lagrange parameters can be non-zero only if the corresponding inequality constraint is an

Input	training set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} \alpha_i = 1$ and $\alpha_i \geq 0$, $i = 1, \ldots, \ell$.
4	$r^* = \sqrt{W(\boldsymbol{\alpha}^*)}$
5	$D = \sum_{i,j=1}^{\ell} \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j) - r^{*2}$
6	$f(\mathbf{x}) = \mathcal{H}\left[\kappa(\mathbf{x}, \mathbf{x}) - 2\sum_{i=1}^{\ell} \alpha_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + D\right]$
7	$\mathbf{c}^* = \sum_{i=1}^{\ell} \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i)$
Output	centre of sphere $\mathbf{c}^*$ and/or function f testing for inclusion

Code Fragment 7.1. Pseudocode for computing the minimal hypersphere.

equality at the solution. These so-called Karush–Kuhn–Tucker (KKT) complementarity conditions are satisfied by the optimal solutions $\boldsymbol{\alpha}^*$, $(\mathbf{c}^*, r^*)$

$$\alpha_i^* \left[\|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}^*\|^2 - r^{*2} \right] = 0, \quad i = 1, \ldots, \ell.$$

This implies that only the training examples $\mathbf{x}_i$ that lie on the surface of the optimal hypersphere have their corresponding α_i^* non-zero. For the remaining examples, the corresponding parameter satisfies $\alpha_i^* = 0$. Hence, in the expression for the centre only the points on the surface are involved. It is for this reason that they are sometimes referred to as *support vectors*.

We will denote the set of indices of the support vectors with sv. Using this notation the pattern function becomes

$$f(\mathbf{x}) = \mathcal{H}\left[\kappa(\mathbf{x}, \mathbf{x}) - 2\sum_{i \in \text{sv}} \alpha_i^* \kappa(\mathbf{x}, \mathbf{x}_i) + D\right],$$

hence involving the evaluation of only $\#\,\text{sv}$ inner products rather than ℓ as was required for the hypersphere of Chapter 5. ■

Remark 7.4 [On convexity] In Chapter 3 we showed that for a kernel function the matrix with entries $(\kappa(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^{\ell}$ is positive semi-definite for all training sets, the so-called finitely positive semi-definite property. This in turn means that the optimisation problem of Algorithm 7.2 is always convex. Hence, the property required for a kernel function to define a feature space also ensures that the minimal hypersphere optimisation problem has a unique solution that can be found efficiently. This rules out the problem of encountering local minima. ■

Note that the function f output by Algorithm 7.2 outputs 1, if the new point lies outside the chosen sphere and so is considered novel, and 0 otherwise. The next section considers bounds on the probability that the novelty detector identifies a point as novel that has been generated by the original distribution, a situation that constitutes an erroneous output. Such examples will be false positives in the sense that they will be normal data identified by the algorithm as novel.

Data arising from novel conditions will be generated by a different distribution and hence we have no way of guaranteeing what output f will give. In this sense we have no way of bounding the negative positive rate. The intuition behind the approach is that the smaller the sphere used to define f, the more likely that novel data will fall outside and hence be detected as novel. Hence, in the subsequent development we will examine ways of shrinking the sphere, while still retaining control of the false positive rate.

7.1.2 Stability of novelty-detection

In the previous section we developed an algorithm for computing the smallest hypersphere enclosing a training sample and for testing whether a new point was contained in that hypersphere. It was suggested that the method could be used as a novelty-detection algorithm where points lying outside the hypersphere would be considered abnormal. But is there any guarantee that points from the same distribution will lie in the hypersphere? Even in the hypersphere based on the centre of gravity of the distribution we had to effectively leave some slack in its radius to allow for the inaccuracy in our estimation of their centre. But if we are to allow some slack in the radius of the minimal hypersphere, how much should it be?

In this section we will derive a stability analysis based on the techniques developed in Chapter 4 that will answer this question and give a novelty-detection algorithm with associated stability guarantees on its performance.

Theorem 7.5 *Fix $\gamma > 0$ and $\delta \in (0,1)$. Let $(\mathbf{c}, r)$ be the centre and radius of a hypersphere in a feature space determined by a kernel κ from a training sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ drawn randomly according to a probability distribution $\mathcal{D}$. Let $g(\mathbf{x})$ be the function defined by*

$$g(\mathbf{x}) = \begin{cases} 0, & \text{if } \|\mathbf{c} - \boldsymbol{\phi}(\mathbf{x})\| \leq r; \\ \left(\|\mathbf{c} - \boldsymbol{\phi}(\mathbf{x})\|^2 - r^2\right)/\gamma, & \text{if } r^2 \leq \|\mathbf{c} - \boldsymbol{\phi}(\mathbf{x})\|^2 \leq r^2 + \gamma; \\ 1, & \text{otherwise.} \end{cases}$$

Then with probability at least $1-\delta$ *over samples of size* ℓ *we have*

$$\mathbb{E}_{\mathcal{D}}\left[g(\mathbf{x})\right] \leq \frac{1}{\ell}\sum_{i=1}^{\ell} g\left(\mathbf{x}_i\right) + \frac{6R^2}{\gamma\sqrt{\ell}} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where R *is the radius of a ball in feature space centred at the origin containing the support of the distribution.*

Proof Consider the loss function $\mathcal{A}:\mathbb{R}\to[0,1]$, given by

$$\mathcal{A}(a) = \begin{cases} 0, & \text{if } R^2 a < r^2 - \|\mathbf{c}\|^2; \\ \left(R^2 a + \|\mathbf{c}\|^2 - r^2\right)/\gamma, & \text{if } r^2 - \|\mathbf{c}\|^2 \leq R^2 a \leq r^2 - \|\mathbf{c}\|^2 + \gamma; \\ 1, & \text{otherwise.} \end{cases}$$

Hence, we can write $g\left(\mathbf{x}\right) = \mathcal{A}\left(f\left(\mathbf{x}\right)\right)$, where

$$f\left(\mathbf{x}\right) = \left\|\mathbf{c} - \boldsymbol{\phi}(\mathbf{x})\right\|^2/R^2 - \left\|\mathbf{c}\right\|^2/R^2 = \left\|\boldsymbol{\phi}(\mathbf{x})\right\|^2/R^2 - 2\left\langle \mathbf{c}, \boldsymbol{\phi}(\mathbf{x})\right\rangle/R^2.$$

Hence, by Theorem 4.9 we have that

$$\mathbb{E}_{\mathcal{D}}\left[g(\mathbf{x})\right] \leq \hat{\mathbb{E}}\left[g(\mathbf{x})\right] + \hat{R}_\ell\left(\mathcal{A}\circ\left(\mathcal{F} + \left\|\boldsymbol{\phi}(\mathbf{x})\right\|^2/R^2\right)\right) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}, \quad (7.2)$$

where $\mathcal{F}$ is the class of linear functions with norm bounded by 1 with respect to the kernel

$$\hat{\kappa}\left(\mathbf{x}_i, \mathbf{x}_j\right) = 4\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)/R^2 = \left\langle 2\boldsymbol{\phi}(\mathbf{x}_i)/R, 2\boldsymbol{\phi}(\mathbf{x}_j)/R\right\rangle.$$

Since $\mathcal{A}\left(0\right) = 0$, we can apply part 4 of Theorem 4.15 with $L = R^2/\gamma$ to give

$$\hat{R}_\ell\left(\mathcal{A}\circ\left(\mathcal{F} + \left\|\boldsymbol{\phi}(\mathbf{x})\right\|^2/R^2\right)\right) \leq 2R^2\hat{R}_\ell\left(\mathcal{F} + \left\|\boldsymbol{\phi}(\mathbf{x})\right\|^2/R^2\right)/\gamma.$$

By part 5 of Theorem 4.15, we have

$$\begin{aligned} \hat{R}_\ell(\mathcal{F} + \left\|\boldsymbol{\phi}(\mathbf{x})\right\|^2/R^2) &\leq \hat{R}_\ell(\mathcal{F}) + 2\sqrt{\hat{\mathbb{E}}\left[\left\|\boldsymbol{\phi}(\mathbf{x})\right\|^4/R^4\right]/\ell} \\ &\leq \hat{R}_\ell(\mathcal{F}) + \frac{2}{\sqrt{\ell}}, \end{aligned}$$

while by Theorem 4.12 we have

$$\hat{R}_\ell(\mathcal{F}) = \frac{2}{\ell}\sqrt{\sum_{i=1}^{\ell}\hat{\kappa}(\mathbf{x}_i, \mathbf{x}_i)} = \frac{4}{R\ell}\sqrt{\sum_{i=1}^{\ell}\kappa(\mathbf{x}_i, \mathbf{x}_i)} = \frac{4}{\sqrt{\ell}}.$$

Putting the pieces into (7.2) gives the result. □

Consider applying Theorem 7.5 to the minimal hypersphere $(\mathbf{c}^*, r^*)$ containing the training data. The first term vanishes since

$$\frac{1}{\ell}\sum_{i=1}^{\ell} g\left(\mathbf{x}_i\right) = 0.$$

If a test point $\mathbf{x}$ lies outside the hypersphere of radius $r = \sqrt{r^{*2} + \gamma}$ with centre $\mathbf{c}^*$ it will satisfy $g\left(\mathbf{x}_i\right) = 1$. Hence with probability greater than $1 - \delta$ we can bound the probability p of such points by

$$\frac{6R^2}{\gamma\sqrt{\ell}} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

since their contribution to $\mathbb{E}_{\mathcal{D}}\left[g(\mathbf{x})\right]$ is p, implying that $p \leq \mathbb{E}_{\mathcal{D}}\left[g(\mathbf{x})\right]$. Since $\left(r^* + \sqrt{\gamma}\right)^2 = r^{*2} + 2r^*\sqrt{\gamma} + \gamma \geq r^{*2} + \gamma$ we also have that, with probability greater than $1 - \delta$, points from the training distribution will lie outside a hypersphere of radius $r^* + \sqrt{\gamma}$ centred at $\mathbf{c}^*$ with probability less than

$$\frac{6R^2}{\gamma\sqrt{\ell}} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}.$$

Hence, by choosing a radius slightly larger than r^* we can ensure that test data lying outside the hypersphere can be considered 'novel'.

Remark 7.6 [Size of the hypersphere] The results of this section formalise the intuition that small radius implies high sensitivity to novelties. If for a given kernel the radius is small we can hope for good novelty-detection. The next section will consider ways in which the radius of the ball can be reduced still further, while still retaining control of the sensitivity of the detector. ■

7.1.3 Hyperspheres containing most of the points

We have seen in the last section how enlarging the radius of the smallest hypersphere containing the data ensures that we can guarantee with high probability that it contains the support of most of the distribution. This still leaves unresolved the sensitivity of the solution to the position of just one point, something that undermines the reliability of the parameters, resulting in a pattern analysis system that is not robust.

Theorem 7.5 also suggests a solution to this problem. Since the bound also applies to hyperspheres that fail to contain some of the training data,

we can consider smaller hyperspheres provided we control the size of the term

$$\frac{1}{\ell}\sum_{i=1}^{\ell} g\left(\mathbf{x}_i\right) \leq \frac{1}{\gamma}\sum_{i=1}^{\ell}\left(\left\|\mathbf{c}-\boldsymbol{\phi}(\mathbf{x}_i)\right\|^2 - r^2\right)_+. \tag{7.3}$$

In this way we can consider hyperspheres that balance the loss incurred by missing a small number of points with the reduction in radius that results. These can potentially give rise to more sensitive novelty detectors.

In order to implement this strategy we introduce a notion of *slack variable* $\xi_i = \xi_i\left(\mathbf{c}, r, \mathbf{x}_i\right)$ defined as

$$\xi_i = \left(\left\|\mathbf{c}-\boldsymbol{\phi}(\mathbf{x}_i)\right\|^2 - r^2\right)_+,$$

which is zero for points inside the hypersphere and measures the degree to which the distance squared from the centre exceeds r^2 for points outside. Let $\boldsymbol{\xi}$ denote the vector with entries ξ_i, $i = 1, \ldots, \ell$. Using the upper bound of inequality (7.3), we now translate the bound of Theorem 7.5 into the objective of the optimisation problem (7.1) with a parameter C to control the trade-off between minimising the radius and controlling the slack variables.

Computation 7.7 [Soft minimal hypersphere] The sphere that optimises a trade off between equation (7.3) and the radius of the sphere is given as the solution of

$$\begin{array}{ll} \min_{\mathbf{c},r,\boldsymbol{\xi}} & r^2 + C\left\|\boldsymbol{\xi}\right\|_1 \\ \text{subject to} & \left\|\boldsymbol{\phi}(\mathbf{x}_i)-\mathbf{c}\right\|^2 = (\phi(\mathbf{x}_i)-\mathbf{c})'(\phi(\mathbf{x}_i)-\mathbf{c}) \leq r^2+\xi_i \\ & \xi_i \geq 0, \;\; i = 1, \ldots, \ell. \end{array} \tag{7.4}$$

We will refer to this approach as the *soft minimal hypersphere*. ■

An example of such a sphere obtained using a linear kernel is shown in Figure 7.1. Note how the centre of the sphere marked by a $\times$ obtained by the algorithm is now very close to the centre of the Gaussian distribution generating the data marked by a diamond.

Again introducing Lagrange multipliers we arrive at the Lagrangian

$$L(\mathbf{c}, r, \boldsymbol{\alpha}, \boldsymbol{\xi}) = r^2 + C\sum_{i=1}^{\ell}\xi_i + \sum_{i=1}^{\ell}\alpha_i\left[\left\|\boldsymbol{\phi}(\mathbf{x}_i)-\mathbf{c}\right\|^2 - r^2 - \xi_i\right] - \sum_{i=1}^{\ell}\beta_i\xi_i.$$

Differentiating with respect to the primal variables gives

$$\frac{\partial L(\mathbf{c}, r, \boldsymbol{\alpha}, \boldsymbol{\xi})}{\partial \mathbf{c}} \;=\; 2\sum_{i=1}^{\ell}\alpha_i(\boldsymbol{\phi}(\mathbf{x}_i)-\mathbf{c}) = \mathbf{0};$$

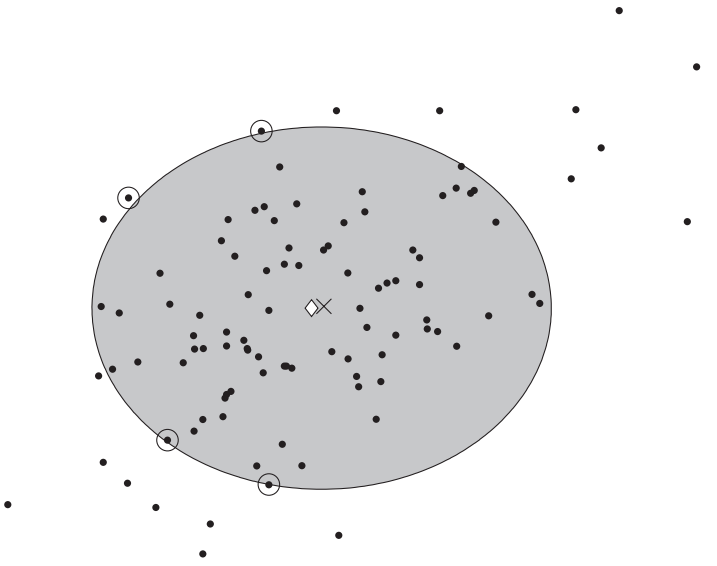

Fig. 7.1. The sphere found by Computation 7.7 using a linear kernel.

$$\begin{aligned}\frac{\partial L(\mathbf{c}, r, \boldsymbol{\alpha}, \boldsymbol{\xi})}{\partial r} &= 2r\left(1 - \sum_{i=1}^{\ell} \alpha_i\right) = 0;\\ \frac{\partial L(\mathbf{c}, r, \boldsymbol{\alpha}, \boldsymbol{\xi})}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0.\end{aligned}$$

The final equation implies that $\alpha_i \leq C$ since $\beta_i = C - \alpha_i \geq 0$. Substituting, we obtain

$$\begin{aligned}L(\mathbf{c}, r, \boldsymbol{\alpha}, \boldsymbol{\xi}) &= r^2 + C\sum_{i=1}^{\ell} \xi_i + \sum_{i=1}^{\ell} \alpha_i \left[\|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\|^2 - r^2 - \xi_i\right] - \sum_{i=1}^{\ell} \beta_i \xi_i\\ &= \sum_{i=1}^{\ell} \alpha_i \langle \boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}, \boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\rangle\\ &= \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j),\end{aligned}$$

which is the dual Lagrangian.

Hence, we obtain the following algorithm.

Algorithm 7.8 [Soft hypersphere minimisation] The hypersphere that optimises the soft bound of Computation 7.7 is computed in Code Fragment 7.2. ■

Input	training set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, $\delta > 0$, $\gamma > 0$, $C > 0$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i \leq C$, $i = 1, \ldots, \ell$.
4	choose i such that $0 < \alpha_i^* < C$
5	$r^* = \sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x}_i) + \sum_{i,j=1}^{\ell} \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j)}$
6	$D = \sum_{i,j=1}^{\ell} \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j) - (r^*)^2 - \gamma$
7	$f(\cdot) = \mathcal{H}\left[\kappa(\cdot, \cdot) - 2\sum_{i=1}^{\ell} \alpha_i^* \kappa(\mathbf{x}_i, \cdot) + D\right]$
8	$\|\boldsymbol{\xi}^*\|_1 = \left(W(\boldsymbol{\alpha}^*) - (r^*)^2\right)/C$
9	$\mathbf{c}^* = \sum_{i=1}^{\ell} \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i)$
Output	centre of sphere $\mathbf{c}^*$ and/or function f testing for containment sum of slacks $\|\boldsymbol{\xi}^*\|_1$, the radius r^*

Code Fragment 7.2. Pseudocode for soft hypersphere minimisation.

The function f again outputs 1 to indicate as novel new data falling outside the sphere. The size of the sphere has been reduced, hence increasing the chances that data generated by a different distribution will be identified as novel. The next theorem shows that this increase in sensitivity has not compromised the false positive rate, that is the probability of data generated according to the same distribution being incorrectly labelled as novel.

Theorem 7.9 *Fix $\delta > 0$ and $\gamma > 0$. Consider a training sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ drawn according to a distribution $\mathcal{D}$ and let $\mathbf{c}^*$, f and $\|\boldsymbol{\xi}^*\|_1$ be the output of Algorithm 7.8. Then the vector $\mathbf{c}^*$ is the centre of the soft minimal hypersphere that minimises the objective $r^2 + C\|\boldsymbol{\xi}\|_1$ for the image $\phi(S)$ of the set S in the feature space F defined by the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$. Furthermore, r^* is the radius of the hypersphere, while the sum of the slack variables is $\|\boldsymbol{\xi}^*\|_1$. The function f outputs 1 on test points $\mathbf{x} \in X$ drawn according to the distribution $\mathcal{D}$ with probability at most*

$$\frac{1}{\gamma\ell}\|\boldsymbol{\xi}^*\|_1 + \frac{6R^2}{\gamma\sqrt{\ell}} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}, \tag{7.5}$$

where R is the radius of a ball in feature space centred at the origin containing the support of the distribution.

Proof The first part of the theorem follows from the previous derivations.

The expression for the radius r^* follows from two applications of the Karush–Kuhn–Tucker conditions using the fact that $0 < \alpha_i^* < C$. Firstly, since $\beta_i^* = C - \alpha_i^* \neq 0$, we have that $\xi_i^* = 0$, while $\alpha_i^* \neq 0$ implies

$$0 = \|\mathbf{x}_i - \mathbf{c}^*\|^2 - (r^*)^2 - \xi_i^* = \|\mathbf{x}_i - \mathbf{c}^*\|^2 - (r^*)^2 .$$

The expression for $\|\boldsymbol{\xi}^*\|_1$ follows from the fact that

$$W(\boldsymbol{\alpha}^*) = (r^*)^2 + C \|\boldsymbol{\xi}^*\|_1 ,$$

while (7.5) follows from Theorem 7.5 and the fact that

$$\frac{1}{\ell} \sum_{i=1}^{\ell} g(\mathbf{x}_i) \leq \frac{1}{\gamma \ell} \|\boldsymbol{\xi}^*\|_1 ,$$

while

$$P_{\mathcal{D}}(f(\mathbf{x}) = 1) \leq \mathbb{E}_{\mathcal{D}}[g(\mathbf{x})] ,$$

where $g(\mathbf{x})$ is the function from Theorem 7.5 with $\mathbf{c} = \mathbf{c}^*$ and $r = r^*$. □

The algorithm is designed to optimise the bound on the probability of new points lying outside the hypersphere. Despite this there may be any number of training points excluded. We are also not guaranteed to obtain the smallest hypersphere that excludes the given number of points.

ν-Formulation There is an alternative way of parametrising the problem that allows us to exert some control over the fraction of points that are excluded from the hypersphere. Note that in Theorem 7.9 the parameter C must be chose larger than $1/\ell$, since otherwise the constraint

$$\sum_{i=1}^{\ell} \alpha_i = 1$$

cannot be satisfied.

Computation 7.10 [ν-soft minimal hypersphere] If we consider setting the parameter $C = 1/(\nu\ell)$, as C varies between $1/\ell$ and ∞ in Theorem 7.9, the same solutions are obtained as the parameter ν varies between 0 and 1 in the optimisation problem

$$\begin{array}{ll} \min_{\mathbf{c},r,\boldsymbol{\xi}} & \frac{1}{\ell}\|\boldsymbol{\xi}\|_1 + \nu r^2 \\ \text{subject to} & \|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbf{c}\|^2 = (\phi(\mathbf{x}_i) - \mathbf{c})'(\phi(\mathbf{x}_i) - \mathbf{c}) \leq r^2 + \xi_i \\ & \xi_i \geq 0, \;\; i = 1, \ldots, \ell. \end{array} \tag{7.6}$$

The solutions clearly correspond since this is just a rescaling of the objective. This approach will be referred to as the *ν-soft minimal hypersphere*. ■

An example of novelty-detection using a radial basis function kernel is given in Figure 7.2. Note how the area of the region has again reduced though since the distribution is a circular Gaussian the performance has probably not improved.

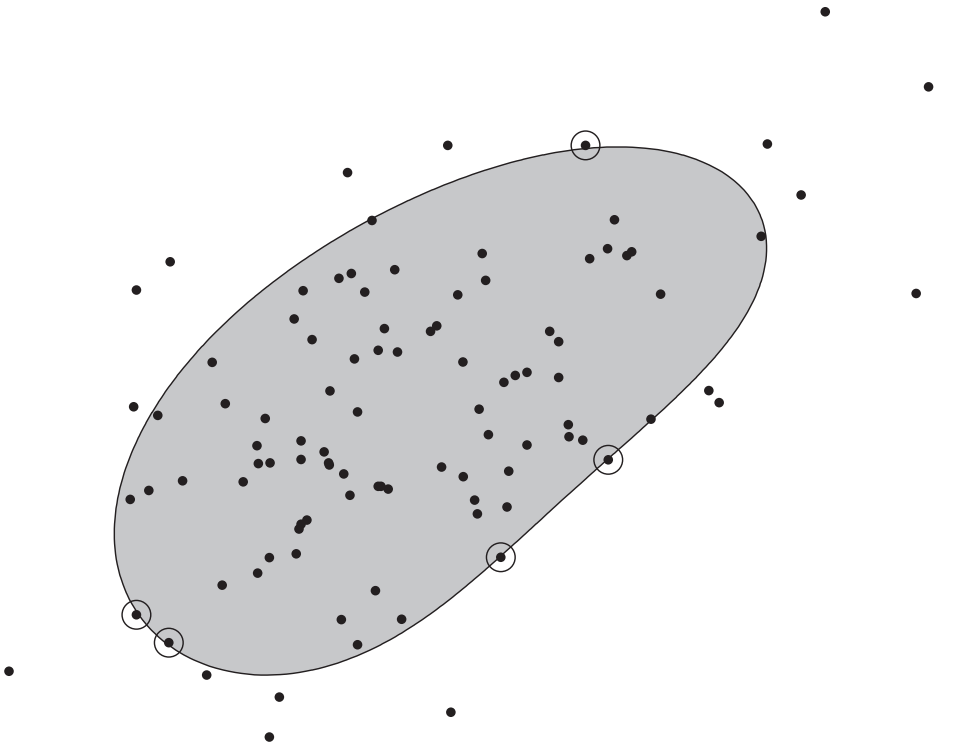

Fig. 7.2. Novelty detection in a kernel defined feature space.

The analysis for the soft hypersphere is identical to the ν-soft minimal hypersphere with an appropriate redefinition of the parameters. Using this fact we obtain the following algorithm.

Algorithm 7.11 [ν-soft minimal hypersphere] The hypersphere that optimises the ν-soft bound is computed in Code Fragment 7.3. ■

As with the previous novelty-detection algorithms, the function f indicates as novel points for which its output is 1. The next theorem is again concerned with bounding the false positive rate, but also indicates the role of the parameter ν.

Input	training set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, $\delta > 0$, $\gamma > 0$, $0 < \nu < 1$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i \leq 1/(\nu\ell)$, $i = 1, \ldots, \ell$.
4	choose i such that $0 < \alpha_i^* < 1/(\nu\ell)$
5	$r^* = \sqrt{\kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x}_i) + \sum_{i,j=1}^{\ell} \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j)}$
6	$D = \sum_{i,j=1}^{\ell} \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j) - (r^*)^2 - \gamma$
7	$f(\cdot) = \mathcal{H}\left[\kappa(\cdot, \cdot) - 2\sum_{i=1}^{\ell} \alpha_i^* \kappa(\mathbf{x}_i, \cdot) + D\right]$
8	$\|\boldsymbol{\xi}^*\|_1 = \nu\ell\left(W(\boldsymbol{\alpha}^*) - (r^*)^2\right)$
9	$\mathbf{c}^* = \sum_{i=1}^{\ell} \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i)$
Output	centre of sphere $\mathbf{c}^*$ and/or function f testing for containment sum of slacks $\|\boldsymbol{\xi}^*\|_1$, the radius r^*

Code Fragment 7.3. Pseudocode for the soft hypersphere.

Theorem 7.12 *Fix $\delta > 0$ and $\gamma > 0$. Consider a training sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ drawn according to a distribution $\mathcal{D}$ and let $\mathbf{c}^*$, f and $\|\boldsymbol{\xi}^*\|_1$ be the output of Algorithm 7.11. Then the vector $\mathbf{c}^*$ is the centre of the soft minimal hypersphere that minimises the objective $r^2 + \|\boldsymbol{\xi}\|_1/(\nu\ell)$ for the image $\phi(S)$ of the set S in the feature space F defined by the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$. Furthermore, r^* is the radius of the hypersphere, while the sum of the slack variables is $\|\boldsymbol{\xi}^*\|_1$ and there are at most $\nu\ell$ training points outside the hypersphere centred at $\mathbf{c}^*$ with radius r^*, while at least $\nu\ell$ of the training points do not lie in the interior of the hypersphere. The function f outputs 1 on test points $\mathbf{x} \in X$ drawn according to the distribution $\mathcal{D}$ with probability at most*

$$\frac{1}{\gamma\ell}\|\boldsymbol{\xi}^*\|_1 + \frac{6R^2}{\gamma\sqrt{\ell}} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where R is the radius of a ball in feature space centred at the origin containing the support of the distribution.

Proof Apart from the observations about the number of training points lying inside and outside the hypersphere the result follows directly from an application of Theorem 7.9 using the fact that the objective can be scaled by ν to give $\nu r^{*2} + \ell^{-1}\|\boldsymbol{\xi}^*\|_1$. For a point $\mathbf{x}_i$ lying outside the hypersphere

we have $\xi_i^* > 0$ implying that $\beta_i^* = 0$, so that $\alpha_i^* = 1/(\nu\ell)$. Since

$$\sum_{i=1}^{\ell} \alpha_i^* = 1,$$

there can be at most $\nu\ell$ such points. Furthermore this equation together with the upper bound on α_i^* implies that at least $\nu\ell$ training points do not lie in the interior of the hypersphere, since for points inside the hypersphere $\alpha_i^* = 0$. □

Remark 7.13 [Varying γ] Theorem 7.12 applies for a fixed value of γ. In practice we would like to choose γ based on the performance of the algorithm. This can be achieved by applying the theorem for a set of k values of γ with the value of δ set to δ/k. This ensures that with probability $1-\delta$ the bound holds for all k choices of γ. Hence, we can apply the most useful value for the given situation at the cost of a slight weakening of the bound. The penalty is an additional $\frac{\ln(k)}{\ell}$ under the square root in the probability bound. We omit this derivation as it is rather technical without giving any additional insight. We will, however, assume that we can choose γ in response to the training data in the corollary below. ■

Theorem 7.12 shows how ν places a lower bound on the fraction of points that fail to be in the interior of the hypersphere and an equal upper bound on those lying strictly outside the hypersphere. Hence, modulo the points lying on the surface of the hypersphere, ν determines the fraction of points not enclosed in the hypersphere. This gives a more intuitive parametrisation of the problem than that given by the parameter C in Theorem 7.9. This is further demonstrated by the following appealing corollary relating the choice of ν to the false positive error rate.

Corollary 7.14 *If we wish to fix the probability bound of Theorem 7.12 to be*

$$p + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}} = \frac{1}{\gamma\ell}\left\|\boldsymbol{\xi}^*\right\|_1 + \frac{6R^2}{\gamma\sqrt{\ell}} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}} \tag{7.7}$$

for some $0 < p < 1$, and can choose γ accordingly, we will minimise the volume of the corresponding test hypersphere obtained by choosing $\nu = p$.

Proof Using the freedom to choose γ, it follows from equation (7.7) that

$$\gamma = \frac{1}{p}\left(\frac{1}{\ell}\left\|\boldsymbol{\xi}^*\right\|_1 + \frac{6R^2}{\sqrt{\ell}}\right)$$

so that the radius squared of the test hypersphere is

$$\begin{aligned} r^{*2} + \gamma &= r^{*2} + \frac{1}{p}\left(\frac{1}{\ell}\left\|\boldsymbol{\xi}^*\right\|_1 + \frac{6R^2}{\sqrt{\ell}}\right) \\ &= r^{*2} + \frac{1}{p\ell}\left\|\boldsymbol{\xi}^*\right\|_1 + \frac{6R^2}{p\sqrt{\ell}}, \end{aligned}$$

implying that p times the volume is

$$pr^{*2} + \frac{1}{\ell}\left\|\boldsymbol{\xi}^*\right\|_1 + \frac{6R^2}{\sqrt{\ell}},$$

which is equivalent to the objective of Computation 7.10 if $\nu = p$. □

Remark 7.15 [Combining with PCA] During this section we have restricted our consideration to hyperspheres. If the data lies in a subspace of the feature space the hypersphere will significantly overestimate the support of the distribution in directions that are perpendicular to the subspace. In such cases we could further reduce the volume of the estimation by performing kernel PCA and applying Theorem 6.14 with δ set to $\delta/2$ to rule out points outside a thin slab around the k-dimensional subspace determined by the first k principal axes. Combining this with Theorem 7.12 also with δ set to $\delta/2$ results in a region estimated by the intersection of the hypersphere with the slab. ■

Remark 7.16 [Alternative approach] If the data is normalised it can be viewed as lying on the surface of a hypersphere in the feature space. In this case there is a correspondence between hyperspheres in the feature space and hyperplanes, since the decision boundary determined by the intersection of the two hyperspheres can equally well be described by the intersection of a hyperplane with the unit hypersphere. The weight vector of the hyperplane is that of the centre of the hypersphere containing the data. This follows immediately from the form of the test function if we assume that $\kappa(\mathbf{x}, \mathbf{x}) = 1$, since

$$\begin{aligned} f(\mathbf{x}) &= \mathcal{H}\left[\kappa(\mathbf{x}, \mathbf{x}) - 2\sum_{i=1}^{\ell} \alpha_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + D\right] \\ &= \mathcal{H}\left[-2\sum_{i=1}^{\ell} \alpha_i^* \kappa(\mathbf{x}_i, \mathbf{x}) + D + 1\right]. \end{aligned}$$

This suggests that an alternative strategy could be to search for a hyperplane that maximally separates the data from the origin with an appropriately

adjusted threshold. For normalised data this will result in exactly the same solution, but for data that is not normalised it will result in the slightly different optimisation problem. The approach taken for classification in the next section parallels this idea. ■

7.2 Support vector machines for classification

In this section we turn our attention to the problem of classification. For novelty-detection we have seen how the stability analysis of Theorem 7.5 guides Computation 7.7 for the soft minimal hypersphere. Such an approach gives a principled way of choosing a pattern function for a particular pattern analysis task. We have already obtained a stability bound for classification in Theorem 4.17 of Chapter 4. This gives a bound on the test misclassification error or generalisation error of a linear function $g(\mathbf{x})$ with norm 1 in a kernel-defined feature space of

$$P_{\mathcal{D}}\left(y \neq g(\mathbf{x})\right) \leq \frac{1}{\ell\gamma}\sum_{i=1}^{\ell}\xi_i + \frac{4}{\ell\gamma}\sqrt{\mathrm{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}, \tag{7.8}$$

where $\mathbf{K}$ is the kernel matrix for the training set and $\xi_i = \xi\left((\mathbf{x}_i, y_i), \gamma, g\right) = \left(\gamma - y_i g(\mathbf{x}_i)\right)_+$. We now use this bound to guide the choice of linear function returned by the learning algorithm. As with the (soft) minimal hyperspheres this leads to a quadratic optimisation problem though with some slight additional complications. Despite these we will follow a similar route to that outlined above starting with separating hyperplanes and moving to soft solutions and eventually to ν-soft solutions.

Remark 7.17 [Choosing γ and the threshold] Again as with the bound for the stability of novelty-detection, strictly speaking the bound of (7.8) only applies if we have chosen γ a priori, while in practice we will choose γ after running the learning algorithm. A similar strategy to that described above involving the application of Theorem 4.17 for a range of values of γ ensures that we can use the bound for approximately the observed value at the cost of a small penalty under the square root. We will again omit these technical details for the sake of readability and treat (7.8) as if it held for all choices of γ. Similarly, the bound was proven for $g(\mathbf{x})$ a simple linear function, while below we will consider the additional freedom of choosing a threshold without adapting the bound to take this into account. ■

7.2.1 The maximal margin classifier

Let us assume initially that for a given training set

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\},$$

there exists a norm 1 linear function

$$g(\mathbf{x}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b$$

determined by a weight vector $\mathbf{w}$ and threshold b and that there exists $\gamma > 0$, such that $\xi_i = (\gamma - y_i g(\mathbf{x}_i))_+ = 0$ for $1 \leq i \leq \ell$. This implies that the first term on the right-hand side of (7.8) vanishes. In the terminology of Chapter 4 it implies that the margin $m(S, g)$ of the training set S satisfies

$$m(S, g) = \min_{1 \leq i \leq \ell} y_i g(\mathbf{x}_i) \geq \gamma.$$

Informally, this implies that the two classes of data can be separated by a hyperplane with a margin of γ as shown in Figure 7.3. We will call such

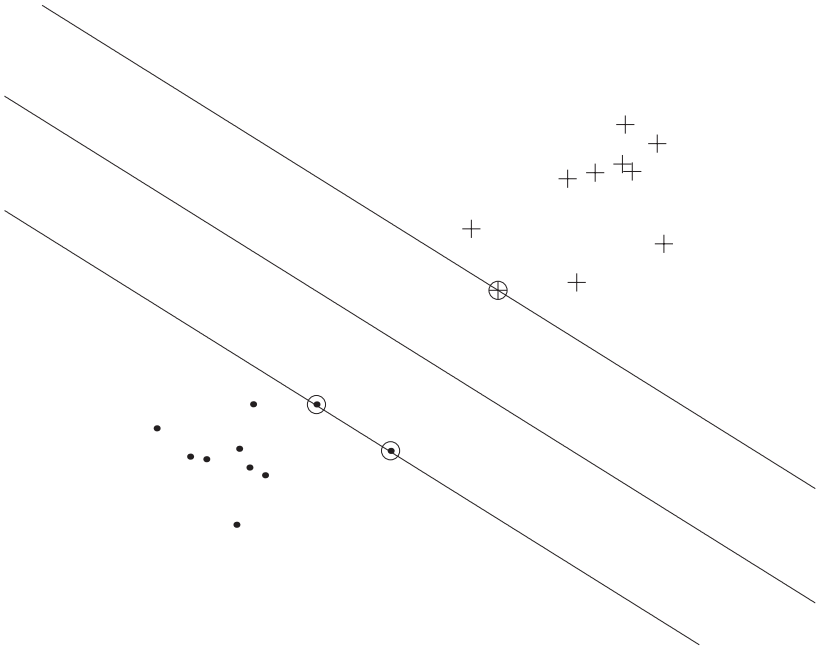

Fig. 7.3. Example of large margin hyperplane with support vectors circled.

a training set *separable* or more precisely *linearly separable* with margin γ. More generally a classifier is called *consistent* if it correctly classifies all of the training set.

Since the function $\mathbf{w}$ has norm 1 the expression $\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle$ measures the length of the perpendicular projection of the point $\boldsymbol{\phi}(\mathbf{x}_i)$ onto the ray determined by $\mathbf{w}$ and so

$$y_i g(\mathbf{x}_i) = y_i \left(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b \right)$$

measures how far the point $\boldsymbol{\phi}(\mathbf{x}_i)$ is from the boundary hyperplane, given by

$$\{ \mathbf{x} : g(\mathbf{x}) = 0 \},$$

measuring positively in the direction of correct classification. For this reason we refer to the functional margin of a linear function with norm 1 as the *geometric margin* of the associated classifier. Hence $m(S, g) \geq \gamma$ implies that S is correctly classified by g with a geometric margin of at least γ.

For such cases we consider optimising the bound of (7.8) over all functions g for which such a γ exists. Clearly, the larger the value of γ the smaller the bound. Hence, we optimise the bound by maximising the margin $m(S, g)$.

Remark 7.18 [Robustness of the maximal margin] Although the stability of the resulting hyperplane is guaranteed provided $m(S, g) = \gamma$ is large, the solution is not robust in the sense that a single additional *training* point can reduce the value of γ very significantly potentially even rendering the training set non-separable. ■

In view of the above criterion our task is to find the linear function that maximises the geometric margin. This function is often referred to as the *maximal margin hyperplane* or the *hard margin support vector machine*.

Computation 7.19 [Hard margin SVM] Hence, the choice of hyperplane should be made to solve the following optimisation problem

$$\begin{array}{ll} \max_{\mathbf{w}, b, \gamma} & \gamma \\ \text{subject to} & y_i \left(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b \right) \geq \gamma, \; i = 1, \ldots, \ell, \\ & \text{and } \|\mathbf{w}\|^2 = 1. \end{array} \tag{7.9}$$

■

Remark 7.20 [Canonical hyperplanes] The traditional way of formulating the optimisation problem makes use of the observation that rescaling the weight vector and threshold does not change the classification function. Hence we can fix the functional margin to be 1 and minimise the norm of the weight vector. We have chosen to use the more direct method here as it

follows more readily from the novelty detector of the previous section and leads more directly to the ν-support vector machine discussed later. ■

For the purposes of conversion to the dual it is better to treat the optimisation as minimising $-\gamma$. As with the novelty-detection optimisation we derive a Lagrangian in order to arrive at the dual optimisation problem. Introducing Lagrange multipliers we obtain

$$L(\mathbf{w}, b, \gamma, \boldsymbol{\alpha}, \lambda) = -\gamma - \sum_{i=1}^{\ell} \alpha_i \left[y_i \left(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + b \right) - \gamma \right] + \lambda \left(\|\mathbf{w}\|^2 - 1 \right).$$

Differentiating with respect to the primal variables gives

$$\begin{aligned}
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\alpha}, \lambda)}{\partial \mathbf{w}} &= -\sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i) + 2\lambda \mathbf{w} = \mathbf{0}, \\
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\alpha}, \lambda)}{\partial \gamma} &= -1 + \sum_{i=1}^{\ell} \alpha_i = 0, \text{ and} \\
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\alpha}, \lambda)}{\partial b} &= -\sum_{i=1}^{\ell} \alpha_i y_i = 0. \qquad (7.10)
\end{aligned}$$

Substituting we obtain

$$\begin{aligned}
L(\mathbf{w}, b, \gamma, \boldsymbol{\alpha}, \lambda) &= -\sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i) \rangle + \lambda \|\mathbf{w}\|^2 - \lambda \\
&= \left(-\frac{1}{2\lambda} + \frac{1}{4\lambda} \right) \sum_{i,j=1}^{\ell} \alpha_i y_i \alpha_j y_j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle - \lambda \\
&= -\frac{1}{4\lambda} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) - \lambda.
\end{aligned}$$

Finally, optimising the choice of λ gives

$$\lambda = \frac{1}{2} \left(\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right)^{1/2},$$

resulting in

$$L(\boldsymbol{\alpha}) = -\left(\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right)^{1/2}, \qquad (7.11)$$

which we call the dual Lagrangian. We have therefore derived the following algorithm.

Algorithm 7.21 [Hard margin SVM] The hard margin support vector machine is implemented in Code Fragment 7.4. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = -\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$, $\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i$, $i = 1, \ldots, \ell$.
4	$\gamma^* = \sqrt{-W(\boldsymbol{\alpha}^*)}$
5	choose i such that $0 < \alpha_i^*$
6	$b = y_i (\gamma^*)^2 - \sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \mathbf{x}_i)$
7	$f(\cdot) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \cdot) + b\right)$;
8	$\mathbf{w} = \sum_{j=1}^{\ell} y_j \alpha_j^* \phi(\mathbf{x}_j)$
Output	weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin γ^* and function f implementing the decision rule represented by the hyperplane

Code Fragment 7.4. Pseudocode for the hard margin SVM.

The following theorem characterises the output and analyses the statistical stability of Algorithm 7.21.

Theorem 7.22 *Fix $\delta > 0$. Suppose that a training sample*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\},$$

is drawn according to a distribution $\mathcal{D}$ is linearly separable in the feature space implicitly defined by the kernel κ and suppose Algorithm 7.21 outputs $\mathbf{w}$, $\boldsymbol{\alpha}^*$, γ^* and the function f. Then the function f realises the hard margin support vector machine in the feature space defined by κ with geometric margin γ^*. Furthermore, with probability $1 - \delta$, the generalisation error of the resulting classifier is bounded by

$$\frac{4}{\ell \gamma^*} \sqrt{\operatorname{tr}(\mathbf{K})} + 3 \sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where $\mathbf{K}$ is the corresponding kernel matrix.

Proof The solution of the optimisation problem in Algorithm 7.21 clearly

optimises (7.11) subject to the constraints of (7.10). Hence, the optimisation of $W(\boldsymbol{\alpha})$ will result in the same solution vector $\boldsymbol{\alpha}^*$. It follows that

$$\gamma^* = -L(\mathbf{w}^*, b^*, \gamma^*, \boldsymbol{\alpha}^*, \lambda^*) = \sqrt{-W(\boldsymbol{\alpha}^*)}.$$

The result follows from these observations and the fact that $\mathbf{w}$ is a simple rescaling of the solution vector $\mathbf{w}^*$ by twice the Lagrange multiplier λ^*. Furthermore

$$2\lambda^* = \frac{2}{2}\left(\sum_{i,j=1}^{\ell} \alpha_i^* \alpha_j^* y_i y_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)^{1/2} = \sqrt{-W(\boldsymbol{\alpha}^*)}.$$

If $\mathbf{w}$ is the solution given by Algorithm 7.21, it is a rescaled version of the optimal solution $\mathbf{w}^*$. Since the weight vector $\mathbf{w}$ has norm and geometric margin equal to $\sqrt{-W(\boldsymbol{\alpha}^*)}$, its functional margin is $-W(\boldsymbol{\alpha}^*) = \gamma^*$, while the vectors with non-zero α_i^* have margin equal to the functional margin – see Remark 7.23 – this gives the formula for b. □

Remark 7.23 [On sparseness] The Karush–Kuhn–Tucker complementarity conditions provide useful information about the structure of the solution. The conditions state that the optimal solutions $\boldsymbol{\alpha}^*$, $(\mathbf{w}^*, b^*)$ must satisfy

$$\alpha_i^* \left[y_i \left(\langle \mathbf{w}^*, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\rangle + b^*\right) - \gamma^*\right] = 0, \qquad i = 1, \ldots, \ell.$$

This implies that only for inputs $\mathbf{x}_i$ for which the geometric margin is γ^*, and that therefore lie closest to the hyperplane, are the corresponding α_i^* non-zero. All the other parameters α_i^* are zero. This is a similar situation to that encountered in the novelty-detection algorithm of Section 7.1. For the same reason the inputs with non-zero α_i^* are called *support vectors* (see Figure 7.3) and again we will denote the set of indices of the support vectors with sv. ■

Remark 7.24 [On convexity] Note that the requirement that κ is a kernel means that the optimisation problem of Algorithm 7.21 is convex since the matrix $\mathbf{G} = \left(y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)\right)_{i,j=1}^{\ell}$ is also positive semi-definite, as the following computation shows

$$\begin{aligned}
\boldsymbol{\beta}'\mathbf{G}\boldsymbol{\beta} &= \sum_{i,j=1}^{\ell} \beta_i \beta_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \sum_{i=1}^{\ell} \beta_i y_i \boldsymbol{\phi}(\mathbf{x}_i), \sum_{j=1}^{\ell} \beta_j y_j \boldsymbol{\phi}(\mathbf{x}_j) \right\rangle \\
&= \left\| \sum_{i=1}^{\ell} \beta_i y_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|^2 \geq 0.
\end{aligned}$$

Hence, the property required of a kernel function to define a feature space also ensures that the maximal margin optimisation problem has a unique solution that can be found efficiently. This rules out the problem of local minima often encountered in for example training neural networks. ■

Remark 7.25 [Duality gap] An important result from optimisation theory states that throughout the feasible regions of the primal and dual problems the primal objective is always bigger than the dual objective, when the primal is a minimisation. This is also indicated by the fact that we are minimising the primal and maximising the dual. Since the problems we are considering satisfy the conditions of strong duality, there is no duality gap at the optimal solution. We can therefore use any difference between the primal and dual objectives as an indicator of convergence. We will call this difference the duality gap. Let $\hat{\boldsymbol{\alpha}}$ be the current value of the dual variables. The possibly still negative margin can be calculated as

$$\hat{\gamma} = \frac{\min_{y_i=1}\left(\langle \hat{\mathbf{w}}, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\rangle\right) - \max_{y_i=-1}\left(\langle \hat{\mathbf{w}}, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\rangle\right)}{2},$$

where the current value of the weight vector is $\hat{\mathbf{w}}$. Hence, the duality gap can be computed as

$$-\sqrt{-W(\hat{\boldsymbol{\alpha}})} + \hat{\gamma}.$$

■

Alternative formulation There is an alternative way of defining the maximal margin optimisation by constraining the functional margin to be 1 and minimising the norm of the weight vector that achieves this. Since the resulting classification is invariant to rescalings this delivers the same classifier. We can arrive at this formulation directly from the dual optimisation problem (7.10) if we use a Lagrange multiplier to incorporate the constraint

$$\sum_{i=1}^{\ell} \alpha_i = 1$$

into the optimisation. Again using the invariance to rescaling we can elect to fix the corresponding Lagrange variable to a value of 2. This gives the following algorithm.

Algorithm 7.26 [Alternative hard margin SVM] The alternative hard margin support vector machine is implemented in Code Fragment 7.5. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$ and $0 \leq \alpha_i$, $i = 1, \ldots, \ell$.
4	$\gamma^* = \left(\sum_{i \in \text{sv}} \alpha_i^*\right)^{-1/2}$
5	choose i such that $0 < \alpha_i^*$
6	$b = y_i - \sum_{j \in \text{sv}} \alpha_j^* y_j \kappa(\mathbf{x}_j, \mathbf{x}_i)$
7	$f(\cdot) = \text{sgn}\left(\sum_{j \in \text{sv}} \alpha_j^* y_j \kappa(\mathbf{x}_j, \cdot) + b\right)$;
8	$\mathbf{w} = \sum_{j \in \text{sv}} y_j \alpha_j^* \phi(\mathbf{x}_j)$
Output	weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin γ^* and function f implementing the decision rule represented by the hyperplane

Code Fragment 7.5. Pseudocode for the alternative version of the hard SVM.

The following theorem characterises the output and analyses the stability of Algorithm 7.26.

Theorem 7.27 *Fix $\delta > 0$. Suppose that a training sample*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\},$$

is drawn according to a distribution $\mathcal{D}$, is linearly separable in the feature space implicitly defined by the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, and suppose Algorithm 7.26 outputs $\mathbf{w}$, $\boldsymbol{\alpha}^*$, γ^* and the function f. Then the function f realises the hard margin support vector machine in the feature space defined by κ with geometric margin γ^*. Furthermore, with probability $1 - \delta$, the generalisation error is bounded by

$$\frac{4}{\ell \gamma^*} \sqrt{\text{tr}(\mathbf{K})} + 3 \sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where $\mathbf{K}$ is the corresponding kernel matrix.

Proof The generalisation follows from the equivalence of the two classifiers. It therefore only remains to show that the expression for γ^* correctly computes the geometric margin. Since we know that the solution is just a scaling of the solution of problem (7.10) we can seek the solution by optimising μ, where

$$\boldsymbol{\alpha} = \mu \boldsymbol{\alpha}^\dagger$$

and $\boldsymbol{\alpha}^\dagger$ is the solution to problem (7.10). Hence, μ is chosen to maximise

$$\mu \sum_{i=1}^{\ell} \alpha_i^\dagger - \frac{\mu^2}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i^\dagger \alpha_j^\dagger \kappa(\mathbf{x}_i, \mathbf{x}_j) = \mu - \frac{\mu^2}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i^\dagger \alpha_j^\dagger \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

giving

$$\mu^* = \left(\sum_{i,j=1}^{\ell} y_i y_j \alpha_i^\dagger \alpha_j^\dagger \kappa(\mathbf{x}_i, \mathbf{x}_j) \right)^{-1} = -W(\boldsymbol{\alpha}^\dagger)^{-1} = (\gamma^*)^{-2},$$

implying

$$\gamma^* = (\mu^*)^{-1/2} = \left(\mu^* \sum_{i=1}^{\ell} \alpha_i^\dagger \right)^{-1/2} = \left(\sum_{i=1}^{\ell} \alpha_i^* \right)^{-1/2},$$

as required. □

An example using the Gaussian kernel is shown in Figure 7.4.

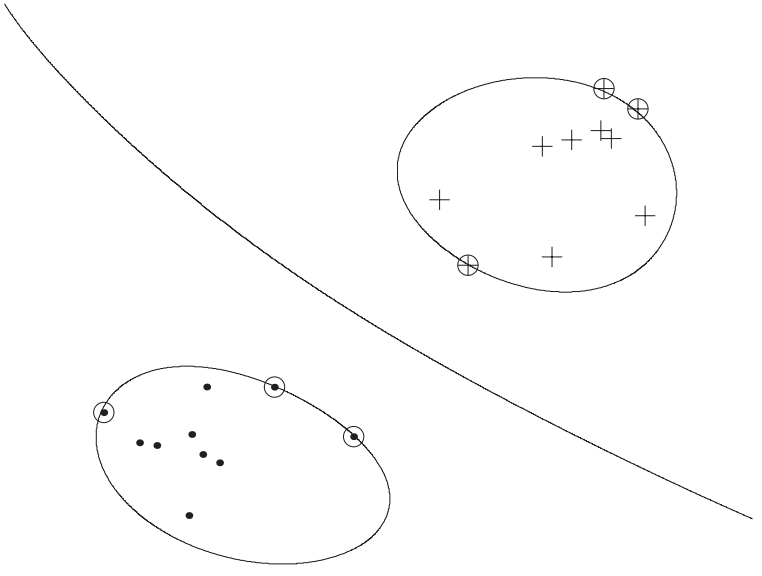

Fig. 7.4. Decision boundary and support vectors when using a gaussian kernel.

7.2.2 Soft margin classifiers

The maximal margin classifier is an important concept, but it can only be used if the data are separable. For this reason it is not applicable in many

real-world problems where the data are frequently noisy. If we are to ensure linear separation in the feature space in such cases, we will need very complex kernels that may result in overfitting. Since the hard margin support vector machine always produces a consistent hypothesis, it is extremely sensitive to noise in the training data. The dependence on a quantity like the margin opens the system up to the danger of being very sensitive to a few points. For real data this will result in a non-robust estimator.

This problem motivates the development of more robust versions that can tolerate some noise and outliers in the training set without drastically altering the resulting solution. The motivation of the maximal margin hyperplane was the bound given in (7.8) together with the assumption that the first term vanishes. It is the second assumption that led to the requirement that the data be linearly separable. Hence, if we relax this assumption and just attempt to optimise the complete bound we will be able to tolerate some misclassification of the training data. Exactly as with the novelty detector we must optimise a combination of the margin and 1-norm of the vector $\boldsymbol{\xi}$, where $\xi_i = \xi\left((y_i, \mathbf{x}_i), \gamma, g\right) = \left(\gamma - y_i g(\mathbf{x}_i)\right)_+$. Introducing this vector into the optimisation criterion results in an optimisation problem with what are known as *slack variables* that allow the margin constraints to be violated. For this reason we often refer to the vector $\boldsymbol{\xi}$ as the *margin slack vector*.

Computation 7.28 [1-norm soft margin SVM] The 1-norm soft margin support vector machine is given by the computation

$$\begin{array}{ll} \min_{\mathbf{w},b,\gamma,\boldsymbol{\xi}} & -\gamma + C\sum_{i=1}^{\ell}\xi_i \\ \text{subject to} & y_i\left(\langle \mathbf{w}, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\rangle + b\right) \geq \gamma - \xi_i,\ \xi_i \geq 0, \\ & i = 1, \ldots, \ell, \text{ and } \|\mathbf{w}\|^2 = 1. \end{array} \tag{7.12}$$

■

The parameter C controls the trade-off between the margin and the size of the slack variables. The optimisation problem (7.12) controls the 1-norm of the margin slack vector. It is possible to replace the 1-norm with the square of the 2-norm. The generalisation analysis for this case is almost identical except for the use of the alternative squared loss function

$$\mathcal{A}(a) = \begin{cases} 1, & \text{if } a < 0; \\ \left(1 - a/\gamma\right)^2, & \text{if } 0 \leq a \leq \gamma; \\ 0, & \text{otherwise.} \end{cases}$$

The resulting difference when compared to Theorem 4.17 is that the empirical loss involves $1/\gamma^2$ rather than $1/\gamma$ and the Lipschitz constant is $2/\gamma$ in

place of $1/\gamma$. Hence, the bound becomes

$$P_{\mathcal{D}}\left(y \neq g(\mathbf{x})\right) \leq \frac{1}{\ell\gamma^2}\sum_{i=1}^{\ell}\xi_i^2 + \frac{8}{\ell\gamma}\sqrt{\mathrm{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}. \tag{7.13}$$

In the next section we look at optimising the 1-norm bound and, following that, turn our attention to the case of the 2-norm of the slack variables.

1-Norm soft margin – the box constraint The corresponding Lagrangian for the 1-norm soft margin optimisation problem is

$$\begin{aligned} L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \lambda) &= -\gamma + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i\left[y_i(\langle\boldsymbol{\phi}\left(\mathbf{x}_i\right), \mathbf{w}\rangle + b) - \gamma + \xi_i\right] \\ &\quad - \sum_{i=1}^{\ell}\beta_i\xi_i + \lambda\left(\|\mathbf{w}\|^2 - 1\right) \end{aligned}$$

with $\alpha_i \geq 0$ and $\beta_i \geq 0$. The corresponding dual is found by differentiating with respect to $\mathbf{w}$, $\boldsymbol{\xi}$, γ and b, and imposing stationarity:

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \lambda)}{\partial \mathbf{w}} &= 2\lambda\mathbf{w} - \sum_{i=1}^{\ell} y_i\alpha_i\boldsymbol{\phi}\left(\mathbf{x}_i\right) = \mathbf{0}, \\ \frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \lambda)}{\partial \xi_i} &= C - \alpha_i - \beta_i = 0, \\ \frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \lambda)}{\partial b} &= \sum_{i=1}^{\ell} y_i\alpha_i = 0, \\ \frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \lambda)}{\partial \gamma} &= 1 - \sum_{i=1}^{\ell}\alpha_i = 0. \end{aligned}$$

Resubstituting the relations obtained into the primal, we obtain the following adaptation of the dual objective function

$$L(\boldsymbol{\alpha}, \lambda) = -\frac{1}{4\lambda}\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) - \lambda,$$

which, again optimising with respect to λ, gives

$$\lambda^* = \frac{1}{2}\left(\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)^{1/2} \tag{7.14}$$

resulting in

$$L(\boldsymbol{\alpha}) = -\left(\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)^{1/2}.$$

This is identical to that for the maximal margin, the only difference being that the constraint $C-\alpha_i-\beta_i=0$, together with $\beta_i \geq 0$ enforces $\alpha_i \leq C$. The KKT complementarity conditions are therefore

$$\begin{aligned} &\alpha_i \left[y_i(\langle \boldsymbol{\phi}\left(\mathbf{x}_i\right), \mathbf{w}\rangle + b) - \gamma + \xi_i\right] = 0, \quad i = 1, \ldots, \ell, \\ &\xi_i \left(\alpha_i - C\right) = 0, \qquad\qquad\qquad\qquad\quad i = 1, \ldots, \ell. \end{aligned}$$

Notice that the KKT conditions imply that non-zero slack variables can only occur when $\alpha_i = C$. The computation of b^* and γ^* from the optimal solution $\boldsymbol{\alpha}^*$ can be made from two points $\mathbf{x}_i$ and $\mathbf{x}_j$ satisfying $y_i = -1$, $y_j = +1$ and $C > \alpha_i^*, \alpha_j^* > 0$. It follows from the KKT conditions that

$$y_i(\langle \boldsymbol{\phi}\left(\mathbf{x}_i\right), \mathbf{w}^*\rangle + b^*) - \gamma^* = 0 = y_j(\langle \boldsymbol{\phi}\left(\mathbf{x}_j\right), \mathbf{w}^*\rangle + b^*) - \gamma^*$$

implying that

$$\begin{aligned} -\langle \boldsymbol{\phi}\left(\mathbf{x}_i\right), \mathbf{w}^*\rangle - b^* - \gamma^* &= \langle \boldsymbol{\phi}\left(\mathbf{x}_j\right), \mathbf{w}^*\rangle + b^* - \gamma^* \\ \text{or } b^* &= -0.5\left(\langle \boldsymbol{\phi}\left(\mathbf{x}_i\right), \mathbf{w}^*\rangle + \langle \boldsymbol{\phi}\left(\mathbf{x}_j\right), \mathbf{w}^*\rangle\right) \quad (7.15) \\ \text{while } \gamma^* &= \langle \boldsymbol{\phi}\left(\mathbf{x}_j\right), \mathbf{w}^*\rangle + b^*. \quad (7.16) \end{aligned}$$

We therefore have the following algorithm.

Algorithm 7.29 [1-norm soft margin support vector machine] The 1-norm soft margin support vector machine is implemented in Code Fragment 7.6. ■

The following theorem characterises the output and statistical stability of Algorithm 7.29.

Theorem 7.30 *Fix $\delta > 0$ and $C \in [1/\ell, \infty)$. Suppose that a training sample*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

is drawn according to a distribution $\mathcal{D}$ and suppose Algorithm 7.29 outputs $\mathbf{w}$, $\boldsymbol{\alpha}^*$, γ^* and the function f. Then the function f realises the 1-norm soft margin support vector machine in the feature space defined by κ. Furthermore, with probability $1-\delta$, the generalisation error is bounded by

$$\frac{1}{C\ell} - \frac{\sqrt{-W(\boldsymbol{\alpha}^*)}}{C\ell\gamma^*} + \frac{4}{\ell\gamma^*}\sqrt{\mathrm{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$, $C \in [1/\ell, \infty)$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = -\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$, $\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i \leq C$, $i = 1, \ldots, \ell$.
4	$\lambda^* = \frac{1}{2}\left(\sum_{i,j=1}^{\ell} y_i y_j \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j)\right)^{1/2}$
5	choose i, j such that $-C < \alpha_i^* y_i < 0 < \alpha_j^* y_j < C$
6	$b^* = -\lambda^*(\sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_i) + \sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_j))$
7	$\gamma^* = 2\lambda^* \sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_j) + b^*$
8	$f(\cdot) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \cdot) + b^*\right)$;
9	$\mathbf{w} = \sum_{j=1}^{\ell} y_j \alpha_j^* \phi(\mathbf{x}_j)$
Output	weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin γ^* and function f implementing the decision rule represented by the hyperplane

Code Fragment 7.6. Pseudocode for 1-norm soft margin SVM.

where $\mathbf{K}$ *is the corresponding kernel matrix.*

Proof Note that the rescaling of b^* is required since the function $f(\mathbf{x})$ corresponds to the weight vector

$$\mathbf{w} = 2\lambda^* \mathbf{w}^* = \sum_{i=1}^{\ell} y_i \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i).$$

All that remains to show is that the error bound can be derived from the general formula

$$P_{\mathcal{D}}(y \neq g(\mathbf{x})) \leq \frac{1}{\ell\gamma} \sum_{i=1}^{\ell} \xi_i + \frac{4}{\ell\gamma} \sqrt{\operatorname{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}.$$

We need to compute the sum of the slack variables. Note that at the optimum we have

$$L(\mathbf{w}^*, b^*, \gamma^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \lambda^*) = -\sqrt{-W(\boldsymbol{\alpha}^*)} = -\gamma^* + C \sum_{i=1}^{\ell} \xi_i^*$$

and so

$$\sum_{i=1}^{\ell} \xi_i^* = \frac{\gamma^* - \sqrt{-W(\boldsymbol{\alpha}^*)}}{C}.$$

Substituting into the bound gives the result. □

An example of the soft margin support vector solution using a Gaussian kernel is shown in Figure 7.5. The support vectors with zero slack variables are circled, though there are other support vectors that fall outside the positive and negative region corresponding to their having non-zero slack variables.

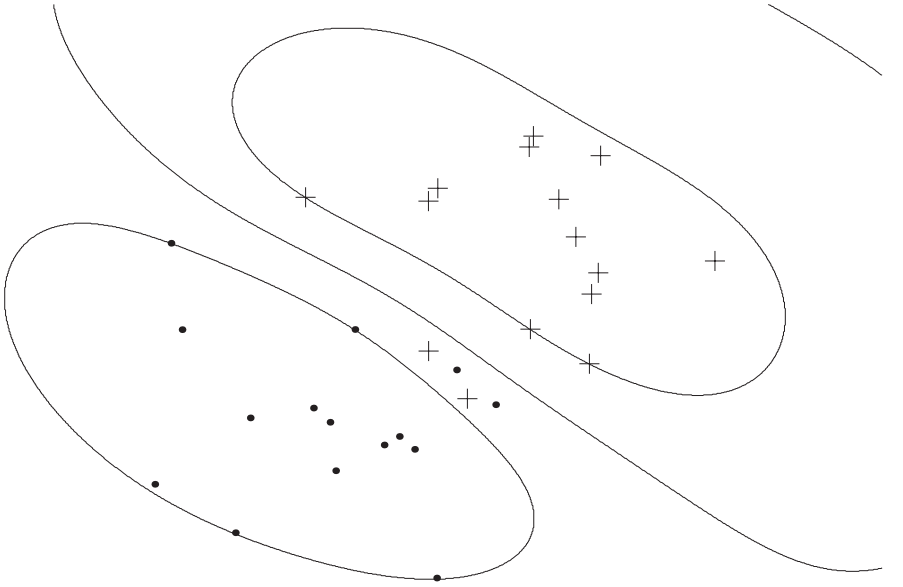

Fig. 7.5. Decision boundary for a soft margin support vector machine using a gaussian kernel.

Surprisingly the algorithm is equivalent to the maximal margin hyperplane, with the additional constraint that all the α_i are upper bounded by C. This gives rise to the name *box constraint* that is frequently used to refer to this formulation, since the vector $\boldsymbol{\alpha}$ is constrained to lie inside the box with side length C in the positive orthant. The trade-off parameter between accuracy and regularisation directly controls the size of the α_i. This makes sense intuitively as the box constraints limit the influence of outliers, which would otherwise have large Lagrange multipliers. The constraint also ensures that the feasible region is bounded and hence that the primal always has a non-empty feasible region.

Remark 7.31 [Tuning the parameter C] In practice the parameter C is varied through a wide range of values and the optimal performance assessed

using a separate validation set or a technique known as cross-validation for verifying performance using only the training set. As the parameter C runs through a range of values, the margin γ^* varies smoothly through a corresponding range. Hence, for a given problem, choosing a particular value for C corresponds to choosing a value for γ^*, and then minimising $\|\boldsymbol{\xi}\|_1$ for that size of margin. ∎

As with novelty-detection, the parameter C has no intuitive meaning. However, the same restrictions on the value of C, namely that $C \geq 1/\ell$, that applied for the novelty-detection optimisation apply here. Again this suggests using

$$C = 1/(\nu\ell),$$

with $\nu \in (0, 1]$ as this leads to a similar control on the number of outliers in a way made explicit in the following theorem. This form of the support vector machine is known as the ν-support vector machine or new support vector machine.

Algorithm 7.32 [ν-support vector machine] The ν-support vector machine is implemented in Code Fragment 7.7. ∎

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$, $\nu \in (0, 1]$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = -\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$, $\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i \leq 1/(\nu\ell)$, $i = 1, \ldots, \ell$.
4	$\lambda^* = \frac{1}{2}\left(\sum_{i,j=1}^{\ell} y_i y_j \alpha_i^* \alpha_j^* \kappa(\mathbf{x}_i, \mathbf{x}_j)\right)^{1/2}$
5	choose i, j such that $-1/(\nu\ell) < \alpha_i^* y_i < 0 < \alpha_j^* y_j < 1/(\nu\ell)$
6	$b^* = -\lambda^*(\sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_i) + \sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_j))$
7	$\gamma^* = 2\lambda^* \sum_{k=1}^{\ell} \alpha_k^* y_k \kappa(\mathbf{x}_k, \mathbf{x}_j) + b^*$
8	$f(\cdot) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j^* y_j \kappa(\mathbf{x}_j, \cdot) + b^*\right)$;
9	$\mathbf{w} = \sum_{j=1}^{\ell} y_j \alpha_j^* \phi(\mathbf{x}_j)$
Output	weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin γ^* and function f implementing the decision rule represented by the hyperplane

Code Fragment 7.7. Pseudocode for the soft margin SVM.

The following theorem characterises the output and analyses the statistical stability of Algorithm 7.32, while at the same time elucidating the role of the parameter ν.

Theorem 7.33 *Fix $\delta > 0$ and $\nu \in (0, 1]$. Suppose that a training sample*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

is drawn according to a distribution $\mathcal{D}$ and suppose Algorithm 7.32 outputs $\mathbf{w}$, $\boldsymbol{\alpha}^*$, γ^* and the function f. Then the function f realises the ν-support vector machine in the feature space defined by κ. Furthermore, with probability $1 - \delta$, the generalisation error of the resulting classifier is bounded by

$$\nu - \frac{\nu\sqrt{-W(\boldsymbol{\alpha}^*)}}{\gamma^*} + \frac{4}{\ell\gamma^*}\sqrt{\mathrm{tr}(\mathbf{K})} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}, \tag{7.17}$$

where $\mathbf{K}$ is the corresponding kernel matrix. Furthermore, there are at most $\nu\ell$ training points that fail to achieve a margin γ^*, while at least $\nu\ell$ of the training points have margin at most γ^*.

Proof This is a direct restatement of Proposition 7.30 with $C = 1/(\nu\ell)$. It remains only to show the bounds on the number of training points failing to achieve the margin γ^* and having margin at most γ^*. The first bound follows from the fact that points failing to achieve margin γ^* have a non-zero slack variable and hence $\alpha_i = 1/(\nu\ell)$. Since

$$\sum_{i=1}^{\ell} \alpha_i = 1,$$

it follows there can be at most $\nu\ell$ such points. Since $\alpha_i \leq 1/(\nu\ell)$ it similarly follows that at least $\nu\ell$ points have non-zero α_i implying that they have margin at most γ^*. □

Remark 7.34 [Tuning ν] The form of the generalisation error bound in Proposition 7.33 gives a good intuition about the role of the parameter ν. It corresponds to the noise level inherent in the data, a value that imposes a lower bound on the generalisation error achievable by any learning algorithm.

We can of course use the bound of (7.17) to guide the best choice of the parameter ν, though strictly speaking we should apply the bound for a range of values of ν, in order to work with the bound with non-fixed ν. This

would lead to an additional $\log(\ell)/\ell$ factor under the final square root, but for simplicity we again omit these details. ■

Remark 7.35 [Duality gap] In the case of the 1-norm support vector machine the feasibility gap can again be computed since the ξ_i, γ, and b are not specified when moving to the dual and so can be chosen to ensure that the primary problem is feasible. If we choose them to minimise the primal we can compute the difference between primal and dual objective functions. This can be used to detect convergence to the optimal solution. ■

2-Norm soft margin – weighting the diagonal In order to minimise the bound (7.13) we can again formulate an optimisation problem, this time involving γ and the 2-norm of the margin slack vector

$$\begin{array}{ll} \min_{\mathbf{w},b,\gamma,\boldsymbol{\xi}} & -\gamma + C\sum_{i=1}^{\ell}\xi_i^2 \\ \text{subject to} & y_i\left(\langle \mathbf{w}, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\rangle + b\right) \geq \gamma - \xi_i,\ \xi_i \geq 0, \\ & i = 1,\ldots,\ell, \text{ and } \|\mathbf{w}\|^2 = 1. \end{array} \tag{7.18}$$

Notice that if $\xi_i < 0$, then the first constraint will still hold if we set $\xi_i = 0$, while this change will reduce the value of the objective function. Hence, the optimal solution for the problem obtained by removing the positivity constraint on ξ_i will coincide with the optimal solution of (7.18). Hence we obtain the solution to (7.18) by solving the following computation.

Computation 7.36 [2-norm soft margin SVM] The 2-norm soft margin support vector machine is given by the optimisation:

$$\begin{array}{ll} \min_{\mathbf{w},b,\gamma,\boldsymbol{\xi}} & -\gamma + C\sum_{i=1}^{\ell}\xi_i^2 \\ \text{subject to} & y_i\left(\langle \mathbf{w}, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\rangle + b\right) \geq \gamma - \xi_i, \\ & i = 1,\ldots,\ell, \text{ and } \|\mathbf{w}\|^2 = 1. \end{array} \tag{7.19}$$

■

The Lagrangian for problem (7.19) of Computation 7.36 is

$$\begin{aligned} L(\mathbf{w},b,\gamma,\boldsymbol{\xi},\boldsymbol{\alpha},\lambda) &= -\gamma + C\sum_{i=1}^{\ell}\xi_i^2 - \sum_{i=1}^{\ell}\alpha_i\left[y_i(\langle\boldsymbol{\phi}\left(\mathbf{x}_i\right),\mathbf{w}\rangle + b) - \gamma + \xi_i\right] \\ &\quad + \lambda\left(\|\mathbf{w}\|^2 - 1\right) \end{aligned}$$

with $\alpha_i \geq 0$. The corresponding dual is found by differentiating with respect to $\mathbf{w}$, $\boldsymbol{\xi}$, γ and b, imposing stationarity

$$\begin{aligned}
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \lambda)}{\partial \mathbf{w}} &= 2\lambda \mathbf{w} - \sum_{i=1}^{\ell} y_i \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) = \mathbf{0}, \\
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \lambda)}{\partial \xi_i} &= 2C\xi_i - \alpha_i = 0, \\
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \lambda)}{\partial b} &= \sum_{i=1}^{\ell} y_i \alpha_i = 0, \\
\frac{\partial L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \lambda)}{\partial \gamma} &= 1 - \sum_{i=1}^{\ell} \alpha_i = 0.
\end{aligned}$$

Resubstituting the relations obtained into the primal, we obtain the following adaptation of the dual objective function

$$L(\mathbf{w}, b, \gamma, \boldsymbol{\xi}, \boldsymbol{\alpha}, \lambda) = -\frac{1}{4C} \sum_{i=1}^{\ell} \alpha_i^2 - \frac{1}{4\lambda} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) - \lambda,$$

which, again optimising with respect to λ, gives

$$\lambda^* = \frac{1}{2} \left(\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right)^{1/2} \tag{7.20}$$

resulting in

$$L(\boldsymbol{\alpha}, \lambda) = -\frac{1}{4C} \sum_{i=1}^{\ell} \alpha_i^2 - \left(\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \right)^{1/2}.$$

We can see that adding the 2-norm regularisation of the slack variables in the primal corresponds to regularising the dual with the 2-norm of the Lagrange multipliers. As C is varied, the size of this 2-norm squared will vary from a minimum of $1/\ell$ corresponding to a uniform allocation of

$$\alpha_i = \frac{1}{\ell}$$

to a maximum of 0.5 when exactly one positive and one negative example each get weight 0.5. Maximising the above objective over $\boldsymbol{\alpha}$ for a particular value C is equivalent to maximising

$$W(\boldsymbol{\alpha}) = -\mu \sum_{i=1}^{\ell} \alpha_i^2 - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

$$= \quad -\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j \left(\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) + \mu\delta_{ij}\right),$$

for some value of $\mu = \mu(C)$, where δ_{ij} is the Kronecker δ defined to be 1 if $i = j$ and 0 otherwise. But this is just the objective of Algorithm 7.21 with the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ replaced by $(\kappa(\mathbf{x}_i, \mathbf{x}_j) + \mu\delta_{ij})$.

Hence, we have the following algorithm.

Algorithm 7.37 [2-norm soft margin SVM] The 2-norm soft margin support vector machine is implemented in Code Fragment 7.8. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\delta > 0$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
maximise	$W(\boldsymbol{\alpha}) = -\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \left(\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) + \mu\delta_{ij}\right)$
subject to	$\sum_{i=1}^{\ell} y_i \alpha_i = 0$, $\sum_{i=1}^{\ell} \alpha_i = 1$ and $0 \leq \alpha_i$, $i = 1, \ldots, \ell$.
4	$\gamma^* = \sqrt{-W(\boldsymbol{\alpha}^*)}$
5	choose i such that $0 < \alpha_i^*$
6	$b = y_i \left(\gamma^*\right)^2 - \sum_{j=1}^{\ell} \alpha_j^* y_j \left(\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) + \mu\delta_{ij}\right)$
7	$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j^* y_j \kappa\left(\mathbf{x}_j, \mathbf{x}\right) + b\right)$;
8	$\mathbf{w} = \sum_{j=1}^{\ell} y_j \alpha_j^* \phi(\mathbf{x}_j)$
Output	weight vector $\mathbf{w}$, dual solution $\boldsymbol{\alpha}^*$, margin γ^* and function f implementing the decision rule represented by the hyperplane

Code Fragment 7.8. Pseudocode for the 2-norm SVM.

The following theorem characterises the output and analyses the statistical stability of Algorithm 7.37.

Theorem 7.38 *Fix $\delta > 0$. Suppose that a training sample*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

drawn according to a distribution $\mathcal{D}$ in the feature space implicitly defined by the kernel κ and suppose Algorithm 7.37 outputs $\mathbf{w}$, $\boldsymbol{\alpha}^*$, γ^* and the function f. Then the function f realises the hard margin support vector machine in the feature space defined by $(\kappa(\mathbf{x}_i, \mathbf{x}_j) + \mu\delta_{ij})$ with geometric margin γ^*. This is equivalent to minimising the expression $-\gamma + C\sum_{i=1}^{\ell} \xi_i^2$ involving the 2-norm of the slack variables for some value of C, hence realising the

2-norm support vector machine. Furthermore, with probability $1-\delta$*, the generalisation error of the resulting classifier is bounded by*

$$\min\left(\frac{\mu\left\|\boldsymbol{\alpha}^*\right\|^2}{\ell\gamma^{*4}}+\frac{8\sqrt{\mathrm{tr}(\mathbf{K})}}{\ell\gamma^*}+3\sqrt{\frac{\ln(4/\delta)}{2\ell}}, \frac{4\sqrt{\mathrm{tr}(\mathbf{K})+\ell\mu}}{\ell\gamma^*}+3\sqrt{\frac{\ln(4/\delta)}{2\ell}}\right),$$

where $\mathbf{K}$ *is the corresponding kernel matrix.*

Proof The value of the slack variable ξ_i^* can be computed by observing that the contribution to the functional output of the $\mu\delta_{ij}$ term is $\mu\alpha_i^*$ for the unnormalised weight vector $\mathbf{w}$ whose norm is given by

$$\|\mathbf{w}\|^2 = -W(\boldsymbol{\alpha}^*) = \gamma^{*2}.$$

Hence, for the normalised weight vector its value is $\mu\alpha_i^*/\gamma^*$. Plugging this into the bound (7.13) for the 2-norm case shows that the first term of the minimum holds with probability $1-(\delta/2)$. The second term of the minimum holds with probability $1-(\delta/2)$ through an application of the hard margin bound in the feature space defined by the kernel

$$\left(\kappa\left(\mathbf{x}_i,\mathbf{x}_j\right)+\mu\delta_{ij}\right).$$

□

The 2-norm soft margin algorithm reduces to the hard margin case with an extra constant added to the diagonal. In this sense it is reminiscent of the ridge regression algorithm. Unlike ridge regression the 2-norm soft margin algorithm does not lose the sparsity property that is so important for practical applications. We now return to give a more detailed consideration of ridge regression including a strategy for introducing sparsity.

7.3 Support vector machines for regression

We have already discussed the problem of learning a real-valued function in both Chapters 2 and 6. The partial least squares algorithm described in Section 6.7.1 can be used for learning functions whose output is in any Euclidean space, so that the 1-dimensional output of a real-valued function can be seen as a special case. The term regression is generally used to refer to such real-valued learning. Chapter 2 used the ridge regression algorithm to introduce the dual representation of a linear function. We were not, however, in a position to discuss the stability of regression or extensions to the basic algorithm at that stage. We therefore begin this section by redressing this shortcoming of our earlier presentation. Following that we will give a fuller description of ridge regression and other support vector regression methods.

7.3.1 Stability of regression

In order to assess the stability of ridge regression we must choose a pattern function similar to that used for classification functions, namely a measure of discrepancy between the generated output and the desired output. The most common choice is to take the squared loss function between prediction and true output

$$f(\mathbf{z}) = f(\mathbf{x}, y) = \mathcal{L}(y, g(\mathbf{x})) = (y - g(\mathbf{x}))^2.$$

The function g is here the output of the ridge regression algorithm with the form

$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}),$$

where $\boldsymbol{\alpha}$ is given by

$$\boldsymbol{\alpha} = (\mathbf{K} + \boldsymbol{\lambda}\mathbf{I}_\ell)^{-1}\mathbf{y}.$$

We can now apply Theorem 4.9 to this function to obtain the following result.

Theorem 7.39 *Fix $B > 0$ and $\delta \in (0,1)$. Let $\mathcal{F}_B$ be the class of linear functions with norm at most B, mapping from a feature space defined by the kernel κ over a space X. Let*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

be drawn independently according to a probability distribution $\mathcal{D}$ on $X \times \mathbb{R}$, the image of whose support in the feature space is contained in a ball of radius R about the origin, while the support of the output value y lies in the interval $[-BR, BR]$. Then with probability at least $1 - \delta$ over the random draw of S, we have, for all $g \in \mathcal{F}_B$

$$\mathbb{E}_{\mathcal{D}}\left[(y - g(\mathbf{x}))^2\right] \leq \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - g(\mathbf{x}_i))^2 + \frac{16RB}{\ell}\left(B\sqrt{\mathrm{tr}(\mathbf{K})} + \|\mathbf{y}\|_2\right) + 12(RB)^2\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where $\mathbf{K}$ is the kernel matrix of the training set S.

Proof We define the loss function class $\mathcal{L}_{\mathcal{F},h,2}$ to be

$$\mathcal{L}_{\mathcal{F},h,2} = \left\{(g - h)^2 \,\middle|\, g \in \mathcal{F}\right\}.$$

We will apply Theorem 4.9 to the function $(y - g(\mathbf{x}))^2 / (2RB)^2 \in \mathcal{L}_{\mathcal{F},h,2}$ with $\mathcal{F} = \mathcal{F}_{B/(2RB)} = \mathcal{F}_{1/(2R)}$ and $h(\mathbf{x}, y) = y/(2RB)$. Since this ensures that in the support of the distribution the class is bounded in the interval $[0, 1]$, we have

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[(y - g(\mathbf{x}))^2 / (2RB)^2\right] &\leq \hat{\mathbb{E}}\left[(y - g(\mathbf{x}))^2 / (2RB)^2\right] \\ &\quad + \hat{R}_\ell(\mathcal{L}_{\mathcal{F},h,2}) + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}.\end{aligned}$$

Multiplying through by $(2RB)^2$ gives

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[(y - g(\mathbf{x}))^2\right] &\leq \hat{\mathbb{E}}\left[(y - g(\mathbf{x}))^2\right] + (2RB)^2 \hat{R}_\ell(\mathcal{L}_{\mathcal{F},h,2}) \\ &\quad + 12(RB)^2 \sqrt{\frac{\ln(2/\delta)}{2\ell}}.\end{aligned}$$

The first term on the right-hand side is simply the empirical squared loss. By part (vi) of Proposition 4.15 we have

$$\hat{R}_\ell(\mathcal{L}_{\mathcal{F},h,2}) \leq 4\left(\hat{R}_\ell(\mathcal{F}_{1/(2R)}) + 2\sqrt{\hat{\mathbb{E}}\left[y^2/(2RB)^2\right]/\ell}\right).$$

This together with Theorem 4.12 gives the result. □

7.3.2 Ridge regression

Theorem 7.39 shows that the expected value of the squared loss can be bounded by its empirical value together with a term that involves the trace of the kernel matrix and the 2-norm of the output values, but involving a bound on the norm of the weight vector of the linear functions. It therefore suggests that we can optimise the off-training set performance by solving the computation:

Computation 7.40 [Ridge regression optimisation] The ridge regression optimisation is achieved by solving

$$\begin{array}{ll}\min_{\mathbf{w}} & \sum_{i=1}^{\ell} \xi_i^2 \\ \text{subject to} & y_i - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle = \xi_i, \\ & i = 1, \ldots, \ell, \text{ and } \|\mathbf{w}\| \leq B.\end{array} \tag{7.21}$$

■

Applying the Lagrange multiplier technique we obtain the Lagrangian

$$L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\beta}, \lambda) = \sum_{i=1}^{\ell} \xi_i^2 + \sum_{i=1}^{\ell} \beta_i \left[y_i - \langle \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w} \rangle - \xi_i \right] + \lambda \left(\|\mathbf{w}\|^2 - B^2 \right).$$

Again taking derivatives with respect to the primal variables gives

$$2\lambda \mathbf{w} = \sum_{i=1}^{\ell} \beta_i \boldsymbol{\phi}(\mathbf{x}_i) \text{ and } 2\xi_i = \beta_i, \; i = 1, \ldots, \ell.$$

Resubstituting into L we have

$$L(\boldsymbol{\beta}, \lambda) = -\frac{1}{4} \sum_{i=1}^{\ell} \beta_i^2 + \sum_{i=1}^{\ell} \beta_i y_i - \frac{1}{4\lambda} \sum_{i,j=1}^{\ell} \beta_i \beta_j \kappa(\mathbf{x}_i, \mathbf{x}_j) - \lambda B^2.$$

Letting $\alpha_i = \beta_i / (2\lambda)$ be the dual coefficients of the solution weight vector results in the optimisation

$$\min_{\boldsymbol{\alpha}} -\lambda \sum_{i=1}^{\ell} \alpha_i^2 + 2 \sum_{i=1}^{\ell} \alpha_i y_i - \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j).$$

Differentiating with respect to the parameters and setting the derivative equal to zero leads to the following algorithm.

Algorithm 7.41 [Kernel ridge regression] The ridge regression algorithm is implemented as follows:

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $\lambda > 0$
Process	$\boldsymbol{\alpha}^* = (\mathbf{K} + \boldsymbol{\lambda} \mathbf{I}_\ell)^{-1} \mathbf{y}$
2	$f(\mathbf{x}) = \sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x})$
3	$\mathbf{w} = \sum_{j=1}^{\ell} \alpha_j^* \phi(\mathbf{x}_j)$
Output	weight vector $\mathbf{w}$, dual $\boldsymbol{\alpha}^*$ and/or function f implementing ridge regression

■

The algorithm was already introduced in Chapter 2 (see (2.6)). Strictly speaking we should have optimised over λ, but clearly different values of λ correspond to different choices of B, hence varying λ is equivalent to varying B.

The example of ridge regression shows how once again the form of the bound on the stability of the pattern function leads to the optimisation problem that defines the solution of the learning task. Despite this well-founded motivation, dual ridge regression like dual partial least squares suffers from

the disadvantage that the solution vector $\boldsymbol{\alpha}^*$ is not sparse. Hence, to evaluate the learned function on a novel example we must evaluate the kernel with each of the training examples. For large training sets this will make the response time very slow.

The sparsity that arose in the case of novelty-detection and classification had its roots in the inequalities used to define the optimisation criterion. This follows because at the optimum those points for which the function output places them in a region where the loss function has zero derivative must have their Lagrange multipliers equal to zero. Clearly for the 2-norm loss this is never the case.

We therefore now examine how the square loss function of ridge regression can be altered with a view to introducing sparsity into the solutions obtained. This will then lead to the use of the optimisation techniques applied above for novelty-detection and classification but now used to solve regression problems, hence developing the support vector regression (SVR) algorithms.

7.3.3 ε-insensitive regression

In order to encourage sparseness, we need to define a loss function that involves inequalities in its evalution. This can be achieved by ignoring errors that are smaller than a certain threshold $\varepsilon > 0$. For this reason the band around the true output is sometimes referred to as a *tube*. This type of loss function is referred to as an ε-insensitive loss function. Using ε-insensitive loss functions leads to the support vector regression algorithms.

Figure 7.6 shows an example of a one-dimensional regression function with an ε-insensitive band. The variables ξ measure the cost of the errors on the training points. These are zero for all points inside the band. Notice that when $\varepsilon = 0$ we recover standard loss functions such as the squared loss used in the previous section as the following definition makes clear.

Definition 7.42 The *(linear) ε-insensitive loss function* $\mathcal{L}^{\varepsilon}(\mathbf{x}, y, g)$ is defined by

$$\mathcal{L}^{\varepsilon}(\mathbf{x}, y, g) = |y - g(\mathbf{x})|_{\varepsilon} = \max\left(0, |y - g(\mathbf{x})| - \varepsilon\right),$$

where g is a real-valued function on a domain X, $\mathbf{x} \in X$ and $y \in \mathbb{R}$. Similarly the *quadratic ε-insensitive loss* is given by

$$\mathcal{L}_2^{\varepsilon}(\mathbf{x}, y, g) = |y - g(\mathbf{x})|_{\varepsilon}^2.$$

■

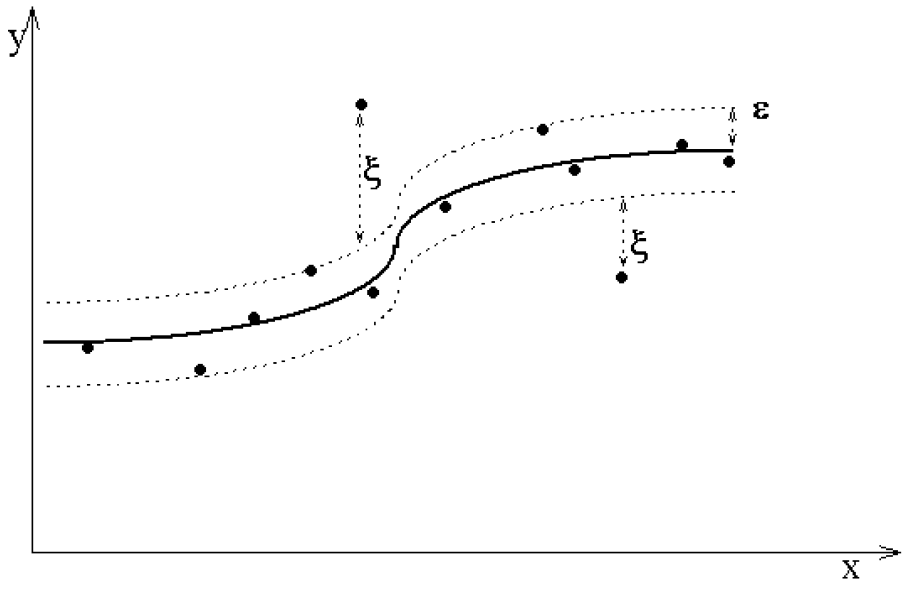

Fig. 7.6. Regression using ε-insensitive loss.

Continuing the development that we began with ridge regression it is most natural to consider taking the square of the ε-insensitive loss to give the so-called quadratic ε-insensitive loss.

Quadratic ε-insensitive loss We can optimise the sum of the quadratic ε-insensitive losses again subject to the constraint that the norm is bounded. This can be cast as an optimisation problem by introducing separate slack variables for the case where the output is too small and the output is too large. Rather than have a separate constraint for the norm of the weight vector we introduce the norm into the objective function together with a parameter C to measure the trade-off between the norm and losses. This leads to the following computation.

Computation 7.43 [Quadratic ε-insensitive SVR] The weight vector $\mathbf{w}$ and threshold b for the quadratic ε-insensitive support vector regression are chosen to optimise the following problem:

$$\left.\begin{array}{ll} \min_{\mathbf{w},b,\boldsymbol{\xi},\hat{\boldsymbol{\xi}}} & \|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}(\xi_i^2 + \hat{\xi}_i^2), \\ \text{subject to} & (\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - y_i \leq \varepsilon + \xi_i,\ i = 1,\ldots,\ell, \\ & y_i - (\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) \leq \varepsilon + \hat{\xi}_i,\ i = 1,\ldots,\ell. \end{array}\right\} \tag{7.22}$$

■

We have not constrained the slack variables to be positive since negative values will never arise at the optimal solution. We have further included an offset parameter b that is not penalised. The dual problem can be derived using the standard method and taking into account that $\xi_i\hat{\xi}_i = 0$ and therefore that the same relation $\alpha_i\hat{\alpha}_i = 0$ holds for the corresponding Lagrange multipliers

$$\begin{array}{ll}\max_{\hat{\boldsymbol{\alpha}},\boldsymbol{\alpha}} & \sum_{i=1}^{\ell} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon\sum_{i=1}^{\ell}(\hat{\alpha}_i + \alpha_i) \\ & \quad -\frac{1}{2}\sum_{i,j=1}^{\ell}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\left(\kappa(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{C}\delta_{ij}\right), \\ \text{subject to} & \sum_{i=1}^{\ell}(\hat{\alpha}_i - \alpha_i) = 0, \\ & \hat{\alpha}_i \geq 0,\ \alpha_i \geq 0,\ i = 1, \ldots, \ell.\end{array}$$

The corresponding KKT complementarity conditions are

$$\begin{aligned}&\alpha_i\left(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b - y_i - \varepsilon - \xi_i\right) = 0, && i = 1, \ldots, \ell, \\ &\hat{\alpha}_i\left(y_i - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle - b - \varepsilon - \hat{\xi}_i\right) = 0, && i = 1, \ldots, \ell, \\ &\xi_i\hat{\xi}_i = 0,\ \alpha_i\hat{\alpha}_i = 0, && i = 1, \ldots, \ell,\end{aligned}$$

Remark 7.44 [Alternative formulation] Note that by substituting $\boldsymbol{\beta} = \hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}$ and using the relation $\alpha_i\hat{\alpha}_i = 0$, it is possible to rewrite the dual problem in a way that more closely resembles the classification case

$$\begin{array}{ll}\max_{\boldsymbol{\beta}} & \sum_{i=1}^{\ell} y_i\beta_i - \varepsilon\sum_{i=1}^{\ell}|\beta_i| - \frac{1}{2}\sum_{i,j=1}^{\ell}\beta_i\beta_j\left(\kappa(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{C}\delta_{ij}\right), \\ \text{subject to} & \sum_{i=1}^{\ell}\beta_i = 0.\end{array}$$

Notice that if we set $\varepsilon = 0$ we recover ridge regression, but with an unpenalised offset that gives rise to the constraint

$$\sum_{i=1}^{\ell}\beta_i = 0.$$

We will in fact use $\boldsymbol{\alpha}$ in place of $\boldsymbol{\beta}$ when we use this form later. ■

Hence, we have the following result for a regression technique that will typically result in a sparse solution vector $\boldsymbol{\alpha}^*$.

Algorithm 7.45 [2-norm support vector regression] The 2-norm support vector regression algorithm is implemented in Code Fragment 7.9. ■

Though the move to the use of the ε-insensitive loss was motivated by the desire to introduce sparsity into the solution, remarkably it can also improve the generalisation error as measured by the expected value of the squared error as is bourne out in practical experiments.

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $C > 0$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
$\max_{\boldsymbol{\alpha}}$	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} y_i \alpha_i - \varepsilon \sum_{i=1}^{\ell} \lvert\alpha_i\rvert - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \left(\kappa(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{C}\delta_{ij}\right)$
subject to	$\sum_{i=1}^{\ell} \alpha_i = 0.$
4	$\mathbf{w} = \sum_{j=1}^{\ell} \alpha_j^* \phi(\mathbf{x}_j)$
5	$b^* = -\varepsilon - (\alpha_i^*/C) + y_i - \sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x}_i)$ for i with $\alpha_i^* > 0$.
6	$f(\mathbf{x}) = \sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x}) + b^*$,
Output	weight vector $\mathbf{w}$, dual $\boldsymbol{\alpha}^*$, b^* and/or function f implementing 2-norm support vector regression

Code Fragment 7.9. Pseudocode for 2-norm support vector regression.

The quadratic ε-insensitive loss follows naturally from the loss function used in ridge regression. There is, however, an alternative that parallels the use of the 1-norm of the slack variables in the support vector machine. This makes use of the linear ε-insensitive loss.

Linear ε-insensitive loss A straightforward rewriting of the optimisation problem (7.22) that minimises the linear loss is as follows:

Computation 7.46 [Linear ε-insensitive SVR] The weight vector $\mathbf{w}$ and threshold b for the linear ε-insensitive support vector regression are chosen to optimise the following problem:

$$\left.\begin{array}{ll} \min_{\mathbf{w},b,\boldsymbol{\xi},\hat{\boldsymbol{\xi}}} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \hat{\xi}_i), \\ \text{subject to} & (\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - y_i \leq \varepsilon + \xi_i,\ i = 1, \ldots, \ell, \\ & y_i - (\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) \leq \varepsilon + \hat{\xi}_i,\ i = 1, \ldots, \ell, \\ & \xi_i, \hat{\xi}_i \geq 0,\ i = 1, \ldots, \ell. \end{array}\right\} \tag{7.23}$$

■

The corresponding dual problem can be derived using the now standard techniques

$$\begin{array}{ll} \max & \sum_{i=1}^{\ell}(\hat{\alpha}_i - \alpha_i)y_i - \varepsilon\sum_{i=1}^{\ell}(\hat{\alpha}_i + \alpha_i) \\ & \qquad -\frac{1}{2}\sum_{i,j=1}^{\ell}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\kappa(\mathbf{x}_i, \mathbf{x}_j), \\ \text{subject to} & 0 \leq \alpha_i, \hat{\alpha}_i \leq C,\ i = 1, \ldots, \ell, \\ & \sum_{i=1}^{\ell}(\hat{\alpha}_i - \alpha_i) = 0,\ i = 1, \ldots, \ell. \end{array}$$

The KKT complementarity conditions are

$$\begin{aligned}
&\alpha_i \left(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b - y_i - \varepsilon - \xi_i\right) = 0, && i = 1, \ldots, \ell,\\
&\hat{\alpha}_i \left(y_i - \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle - b - \varepsilon - \hat{\xi}_i\right) = 0, && i = 1, \ldots, \ell,\\
&\xi_i \hat{\xi}_i = 0,\ \alpha_i \hat{\alpha}_i = 0, && i = 1, \ldots, \ell,\\
&(\alpha_i - C)\,\xi_i = 0,\ (\hat{\alpha}_i - C)\,\hat{\xi}_i = 0, && i = 1, \ldots, \ell.
\end{aligned}$$

Again as mentioned in Remark 7.44 substituting α_i for $\hat{\alpha}_i - \alpha_i$, and taking into account that $\alpha_i \hat{\alpha}_i = 0$, we obtain the following algorithm.

Algorithm 7.47 [1-norm support vector regression] The 1-norm support vector regression algorithm is implemented in Code Fragment 7.10. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $C > 0$
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
$\max_{\boldsymbol{\alpha}}$	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} y_i \alpha_i - \varepsilon \sum_{i=1}^{\ell} \lvert\alpha_i\rvert - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell} \alpha_i = 0$, $-C \leq \alpha_i \leq C$, $i = 1, \ldots, \ell$.
4	$\mathbf{w} = \sum_{j=1}^{\ell} \alpha_j^* \phi(\mathbf{x}_j)$
5	$b^* = -\varepsilon + y_i - \sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x}_i)$ for i with $0 < \alpha_i^* < C$.
6	$f(\mathbf{x}) = \sum_{j=1}^{\ell} \alpha_j^* \kappa(\mathbf{x}_j, \mathbf{x}) + b^*$,
Output	weight vector $\mathbf{w}$, dual $\boldsymbol{\alpha}^*$, b^* and/or function f implementing 1-norm support vector regression

Code Fragment 7.10. Pseudocode for 1-norm support vector regression.

Remark 7.48 [Support vectors] If we consider the band of $\pm\varepsilon$ around the function output by the learning algorithm, the points that are not strictly inside the tube are support vectors. Those not touching the tube will have the absolute value of the corresponding α_i equal to C. ■

Stability analysis of ε-insensitive regression The linear ε-insensitive loss for support vector regression raises the question of what stability analysis is appropriate. When the output values are real there are a large range of possibilities for loss functions all of which reduce to the discrete loss in the case of classification. An example of such a loss function is the loss that counts an error if the function output deviates from the true output by more

than an error bound γ

$$\mathcal{H}^{\gamma}(\mathbf{x}, y, g) = \begin{cases} 0, & \text{if } |y - g(\mathbf{x})| \leq \gamma; \\ 1, & \text{otherwise.} \end{cases}$$

We can now apply a similar generalisation analysis to that developed for classification by introducing a loss function

$$\mathcal{A}(a) = \begin{cases} 0, & \text{if } a < \varepsilon, \\ (a - \varepsilon) / (\gamma - \varepsilon), & \text{if } \varepsilon \leq a \leq \gamma, \\ 1, & \text{otherwise.} \end{cases}$$

Observe that $\mathcal{H}^{\gamma}(\mathbf{x}, y, g) \leq \mathcal{A}(|y - g(\mathbf{x})|) \leq |y - g(\mathbf{x})|_{\varepsilon}$, so that we can apply Theorem 4.9 to $\mathcal{A}(|y - g(\mathbf{x})|)$ to give an upper bound on $\mathbb{E}_{\mathcal{D}}\left[\mathcal{H}^{\gamma}(\mathbf{x}, y, g)\right]$ while the empirical error can be upper bounded by

$$\sum_{i=1}^{\ell} |y_i - g(\mathbf{x}_i)|_{\varepsilon} = \sum_{i=1}^{\ell} (\xi_i + \hat{\xi}_i).$$

Putting the pieces together gives the following result.

Theorem 7.49 *Fix $B > 0$ and $\delta \in (0, 1)$. Let $\mathcal{F}_B$ be the class of linear functions with norm at most B, mapping from a feature space defined by the kernel κ over a space X. Let*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{\ell}, y_{\ell})\}$$

be drawn independently according to a probability distribution $\mathcal{D}$ on $X \times \mathbb{R}$. Then with probability at least $1 - \delta$ over the random draw of S, we have for all $g \in \mathcal{F}_B$

$$\begin{aligned} P_{\mathcal{D}}\left(|y - g(\mathbf{x})| > \gamma\right) &= \mathbb{E}_{\mathcal{D}}\left[\mathcal{H}^{\gamma}(\mathbf{x}, y, g)\right] \\ &\leq \frac{\left\|\boldsymbol{\xi} + \hat{\boldsymbol{\xi}}\right\|_1}{\ell\,(\gamma - \varepsilon)} + \frac{4B\sqrt{\mathrm{tr}(\mathbf{K})}}{\ell\,(\gamma - \varepsilon)} + 3\sqrt{\frac{\ln(2/\delta)}{2\ell}}, \end{aligned}$$

where $\mathbf{K}$ is the kernel matrix of the training set S.

The result shows that bounding a trade-off between the sum of the linear slack variables and the norm of the weight vector will indeed lead to an improved bound on the probability that the output error exceeds γ.

ν-support vector regression One of the attractive features of the 1-norm support vector machine was the ability to reformulate the problem so that the regularisation parameter specifies the fraction of support vectors in the so-called ν-support vector machine. The same approach can be adopted here

in what is known as ν-support vector regression. The reformulation involves the automatic adaptation of the size ε of the tube.

Computation 7.50 [ν-support vector regression] The weight vector $\mathbf{w}$ and threshold b for the ν-support vector regression are chosen to optimise the following problem:

$$\left.\begin{array}{ll} \min_{\mathbf{w},b,\varepsilon,\boldsymbol{\xi},\hat{\boldsymbol{\xi}}} & \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\nu\varepsilon + \frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \hat{\xi}_i)\right), \\ \text{subject to} & (\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) - y_i \leq \varepsilon + \xi_i, \\ & y_i - (\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + b) \leq \varepsilon + \hat{\xi}_i, \\ & \xi_i, \hat{\xi}_i \geq 0,\ i = 1, \ldots, \ell, \end{array}\right\} \tag{7.24}$$

■

Applying the now usual analysis leads to the following algorithm.

Algorithm 7.51 [ν-support vector regression] The ν-support vector regression algorithm is implemented in Code Fragment 7.11. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, $C > 0$, $0 < \nu < 1$.
Process	find $\boldsymbol{\alpha}^*$ as solution of the optimisation problem:
$\max_{\boldsymbol{\alpha}}$	$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} y_i\alpha_i - \varepsilon\sum_{i=1}^{\ell}\lvert\alpha_i\rvert - \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j\kappa(\mathbf{x}_i,\mathbf{x}_j)$
subject to	$\sum_{i=1}^{\ell}\alpha_i = 0$, $\sum_{i=1}^{\ell}\lvert\alpha_i\rvert \leq C\nu$, $-C/\ell \leq \alpha_i \leq C/\ell$, $i = 1, \ldots, \ell$.
4	$\mathbf{w} = \sum_{j=1}^{\ell}\alpha_j^*\phi(\mathbf{x}_j)$
5	$b^* = -\varepsilon + y_i - \sum_{j=1}^{\ell}\alpha_j^*\kappa(\mathbf{x}_j,\mathbf{x}_i)$ for i with $0 < \alpha_i^* < C/\ell$.
6	$f(\mathbf{x}) = \sum_{j=1}^{\ell}\alpha_j^*\kappa(\mathbf{x}_j,\mathbf{x}) + b^*$,
Output	weight vector $\mathbf{w}$, dual $\boldsymbol{\alpha}^*$, b^* and/or function f implementing ν-support vector regression

Code Fragment 7.11. Pseudocode for new SVR.

As with the ν-support vector machine the parameter ν controls the fraction of errors in the sense that there are at most $\nu\ell$ training points that fall outside the tube, while at least $\nu\ell$ of the training points are support vectors and so lie either outside the tube or on its surface.

7.4 On-line classification and regression

The algorithms we have described in this section have all taken a training set S as input and processed all of the training examples at once. Such an algorithm is known as a *batch* algorithm.

In many practical tasks training data must be processed one at a time as it is received, so that learning is started as soon as the first example is received. The learning follows the following protocol. As each example is received the learner makes a prediction of the correct output. The true output is then made available and the degree of mismatch or loss made in the prediction is recorded. Finally, the learner can update his current pattern function in response to the feedback received on the current example. If updates are only made when non-zero loss is experience, the algorithm is said to be *conservative*.

Learning that follows this protocol is known as *on-line learning*. The aim of the learner is to adapt his pattern function as rapidly as possible. This is reflected in the measures of performance adopted to analyse on-line learning. Algorithms are judged according to their ability to control the accumulated loss that they will suffer in processing a sequence of examples. This measure takes into account the rate at which learning takes place.

We first consider a simple on-line algorithm for learning linear functions in an on-line fashion.

The perceptron algorithm The algorithm learns a thresholded linear function

$$h(\mathbf{x}) = \operatorname{sgn}\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle$$

in a kernel-defined feature space in an on-line fashion making an update whenever a misclassified example is processed. If the weight vector after t updates is denoted by $\mathbf{w}_t$ then the update rule for the $(t+1)$st update when an example $(\mathbf{x}_i, y_i)$ is misclassified is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \boldsymbol{\phi}(\mathbf{x}_i).$$

Hence, the corresponding dual update rule is simply

$$\alpha_i = \alpha_i + 1,$$

if we assume that the weight vector is expressed as

$$\mathbf{w}_t = \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{\phi}(\mathbf{x}_i).$$

This is summarised in the following algorithm.

Algorithm 7.52 [Kernel perceptron] The dual perceptron algorithm is implemented in Code Fragment 7.12. ■

Input	training sequence $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell), \ldots$
Process	$\boldsymbol{\alpha} = 0$, $i = 0$, loss $= 0$
2	repeat
3	$i = i + 1$
4	if $\operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j y_j \kappa(\mathbf{x}_j, \mathbf{x}_i)\right) \neq y_i$
5	$\alpha_i = \alpha_i + 1$
6	loss $=$ loss $+1$
7	until finished
8	$f(\mathbf{x}) = \sum_{j=1}^{\ell} \alpha_j y_j \kappa(\mathbf{x}_j, \mathbf{x})$
Output	dual variables $\boldsymbol{\alpha}$, loss and function f

Code Fragment 7.12. Pseudocode for the kernel perceptron algorithm.

We can apply the perceptron algorithm as a batch algorithm to a full training set by running through the set repeating the updates until all of the examples are correctly classified.

Assessing the performance of the perceptron algorithm The algorithm does not appear to be aiming for few updates, but for the batch case the well-known perceptron convergence theorem provides a bound on their number in terms of the margin of the corresponding hard margin support vector machine as stated in the theorem due to Novikoff.

Theorem 7.53 (Novikoff) *If the training points*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

are contained in a ball of radius R about the origin, the hard margin support vector machine weight vector $\mathbf{w}^*$ with no bias has geometric margin γ and we begin with the weight vector

$$\mathbf{w}_0 = \mathbf{0} = \sum_{i=1}^{\ell} 0\boldsymbol{\phi}(\mathbf{x}_i);$$

then the number of updates of the perceptron algorithm is bounded by

$$\frac{R^2}{\gamma^2}.$$

Proof The result follows from two sequences of inequalities. The first shows that as the updates are made the norm of the resulting weight vector cannot grow too fast, since if i is the index of the example used for the tth update, we have

$$\begin{aligned}\|\mathbf{w}_{t+1}\|^2 &= \langle \mathbf{w}_t + y_i \boldsymbol{\phi}(\mathbf{x}_i), \mathbf{w}_t + y_i \boldsymbol{\phi}(\mathbf{x}_i)\rangle \\ &= \|\mathbf{w}_t\|^2 + 2y_i \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}_i)\rangle + \|\boldsymbol{\phi}(\mathbf{x}_i)\|^2 \\ &\leq \|\mathbf{w}_t\|^2 + R^2 \leq (t+1)R^2,\end{aligned}$$

since the fact we made the update implies the middle term cannot be positive. The other sequence of inequalities shows that the inner product between the sequence of weight vectors and the vector $\mathbf{w}^*$ (assumed without loss of generality to have norm 1) increases by a fixed amount each update

$$\langle \mathbf{w}^*, \mathbf{w}_{t+1}\rangle = \langle \mathbf{w}^*, \mathbf{w}_t\rangle + y_i \langle \mathbf{w}^*, \boldsymbol{\phi}(\mathbf{x}_i)\rangle \geq \langle \mathbf{w}^*, \mathbf{w}_t\rangle + \gamma \geq (t+1)\gamma.$$

The two inequalities eventually become incompatible as they imply that

$$t^2\gamma^2 \leq \langle \mathbf{w}^*, \mathbf{w}_t\rangle^2 \leq \|\mathbf{w}_t\|^2 \leq tR^2.$$

Clearly, we must have

$$t \leq \frac{R^2}{\gamma^2},$$

as required. □

The bound on the number of updates indicates that each time we make a mistake we are effectively offsetting the cost of that mistake with some progress towards learning a function that correctly classifies the training set. It is curious that the bound on the number of updates is reminiscient of the bound on the generalisation of the hard margin support vector machine.

Despite the number of updates not being a bound on the generalisation performance of the resulting classifier, we now show that it does imply such a bound. Indeed the type of analysis we now present will also imply a bound on the generalisation of the hard margin support vector machine in terms of the number of support vectors.

Recall that for the various support vector machines for classification and the ε-insensitive support vector machine for regression only a subset of the Lagrange multipliers is non-zero. This property of the solutions is referred to as *sparseness*. Furthermore, the support vectors contain all the information necessary to reconstruct the hyperplane or regression function. Hence, for classification even if all of the other points were removed the same maximal separating hyperplane would be found from the remaining subset of

the support vectors. This shows that the maximal margin hyperplane is a compression scheme in the sense that from the subset of support vectors we can reconstruct the maximal margin hyperplane that correctly classifies the whole training set.

For the perceptron algorithm the bound is in terms of the number of updates made to the hypothesis during learning, that is the number bounded by Novikoff's theorem. This is because the same hypothesis would be generated by performing the same sequence of updates while ignoring the examples on which updates were not made. These examples can then be considered as test examples since they were not involved in the generation of the hypothesis. There are ℓ^k ways in which a sequence of k updates can be created from a training set of size ℓ, so we have a bound on the number of hypotheses considered. Putting this together using a union bound on probability gives the following proposition.

Theorem 7.54 *Fix $\delta > 0$. If the perceptron algorithm makes $1 \leq k \leq \ell/2$ updates before converging to a hypothesis $f(\mathbf{x})$ that correctly ranks a training set*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

drawn independently at random according to a distribution $\mathcal{D}$, then with probability at least $1 - \delta$ over the draw of the set S, the generalisation error of $f(x)$ is bounded by

$$P_{\mathcal{D}}(f(\mathbf{x}) \neq y) \leq \frac{1}{\ell - k}\left(k \ln \ell + \ln \frac{\ell}{2\delta}\right). \tag{7.25}$$

Proof We first fix $1 \leq k \leq \ell/2$ and consider the possible hypotheses that can be generated by sequences of k examples. The proof works by bounding the probability that we choose a hypothesis for which the generalisation error is worse than the bound. We use bold $\mathbf{i}$ to denote the sequence of indices on which the updates are made and $\mathbf{i}_0$ to denote some a priori fixed sequence of indices. With $f_{\mathbf{i}}$ we denote the function obtained by updating on the sequence $\mathbf{i}$

$$\begin{aligned}
& P\left\{S : \exists \mathbf{i} \text{ s.t. } P_{\mathcal{D}}(f_{\mathbf{i}}(\mathbf{x}) \neq y) > \frac{1}{\ell - k}\left(k \ln \ell + \ln \frac{\ell}{2\delta}\right)\right\} \\
\leq\ & \ell^k P\left\{S : P_{\mathcal{D}}(f_{\mathbf{i}_0}(\mathbf{x}) \neq y) > \frac{1}{\ell - k}\left(k \ln \ell + \ln \frac{\ell}{2\delta}\right)\right\} \\
\leq\ & \ell^k \left(1 - \frac{1}{\ell - k}\left(k \ln \ell + \ln \frac{\ell}{2\delta}\right)\right)^{\ell - k}
\end{aligned}$$

$$\begin{aligned} &\leq \ell^k \exp\left(-\frac{\ell-k}{\ell-k}\left(k\ln\ell+\ln\frac{\ell}{2\delta}\right)\right) \\ &\leq \frac{2\delta}{\ell}. \end{aligned}$$

Hence, the total probability of the bound failing over the different choices of k is at most δ as required. □

Combining this with the bound on the number of updates provided by Novikoff's theorem gives the following corollary.

Corollary 7.55 *Fix $\delta > 0$. Suppose the hard margin support vector machine has margin γ on the training set*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

drawn independently at random according to a distribution $\mathcal{D}$ and contained in a ball of radius R about the origin. Then with probability at least $1-\delta$ over the draw of the set S, the generalisation error of the function $f(\mathbf{x})$ obtained by running the perceptron algorithm on S in batch mode is bounded by

$$P_{\mathcal{D}}\left(f(\mathbf{x}) \neq y\right) \leq \frac{2}{\ell}\left(\frac{R^2}{\gamma^2}\ln\ell + \ln\frac{\ell}{2\delta}\right),$$

provided

$$\frac{R^2}{\gamma^2} \leq \frac{\ell}{2}.$$

There is a similar bound on the generalisation of the hard margin support vector machine in terms of the number of support vectors. The proof technique mimics that of Theorem 7.54, the only difference being that the order of the support vectors does not affect the function obtained. This gives the following bound on the generalisation quoted without proof.

Theorem 7.56 *Fix $\delta > 0$. Suppose the hard margin support vector machine has margin γ on the training set*

$$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

drawn independently at random according to a distribution $\mathcal{D}$. Then with probability at least $1-\delta$ over the draw of the set S, its generalisation error is bounded by

$$\frac{1}{\ell-d}\left(d\log\frac{e\ell}{d} + \log\frac{\ell}{\delta}\right),$$

where $d = \#\,\mathrm{sv}$ *is the number of support vectors.*

The theorem shows that the smaller the number of support vectors the better the generalisation that can be expected. If we were to use the bound to guide the learning algorithm a very different approach would result. Indeed we can view the perceptron algorithm as a greedy approach to optimising this bound, in the sense that it only makes updates and hence creates non-zero α_i when this is forced by a misclassification.

Curiously the generalisation bound for the perceptron algorithm is at least as good as the margin bound obtained for the hard margin support vector machine! In practice the support vector machine typically gives better generalisation, indicating that the apparent contradiction arises as a result of the tighter proof technique that can be used in this case.

Remark 7.57 [Expected generalisation error] A slightly tighter bound on the *expected* generalisation error of the support vector machine in terms of the same quantities can be obtained by a leave-one-out argument. Since, when a non-support vector is omitted, it is correctly classified by the remaining subset of the training data the leave-one-out estimate of the generalisation error is

$$\frac{\#\,\mathrm{sv}}{\ell}.$$

A cyclic permutation of the training set shows that the expected error of a test point is bounded by this quantity. The use of an expected generalisation bound gives no guarantee about its variance and hence its reliability. Indeed leave-one-out bounds are known to suffer from this problem. Theorem 7.56 can be seen as showing that in the case of maximal margin classifiers a slightly weaker bound does hold with high probability and hence that in this case the variance cannot be too high. ■

Remark 7.58 [Effects of the margin] Note that in SVMs the margin has two effects. Its maximisation ensures a better bound on the generalisation, but at the same time it is the margin that is the origin of the sparseness of the solution vector, as the inequality constraints generate the KKT complementarity conditions. As indicated above the maximal margin classifier does not attempt to control the number of support vectors and yet in practice there are frequently few non-zero α_i. This sparseness of the solution can be exploited by implementation techniques for dealing with large datasets. ■

Kernel adatron There is an on-line update rule that models the hard margin support vector machine with fixed threshold 0. It is a simple adaptation of the perceptron algorithm.

Algorithm 7.59 [Kernel adatron] The kernel adatron algorithm is implemented in Code Fragment 7.13. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$
Process	$\boldsymbol{\alpha} = 0$, $i = 0$, loss $= 0$
2	repeat
3	for $i = 1 : \ell$
4	$\alpha_i \leftarrow \alpha_i + \left(1 - y_i \sum_{j=1}^{\ell} \alpha_j y_j \kappa\left(\mathbf{x}_j, \mathbf{x}_i\right)\right)$
5	if $\alpha_i < 0$ then $\alpha_i \leftarrow 0$.
6	end
7	until $\boldsymbol{\alpha}$ unchanged
8	$f\left(\mathbf{x}\right) = \operatorname{sgn}\left(\sum_{j=1}^{\ell} \alpha_j y_j \kappa(\mathbf{x}_j, \mathbf{x})\right)$
Output	dual variables $\boldsymbol{\alpha}$, loss and function f

Code Fragment 7.13. Pseudocode for the kernel adatron algorithm.

For each α_i this can be for one of two reasons. If the first update did not change α_i then

$$1 - y_i \sum_{j=1}^{\ell} \alpha_j y_j \kappa\left(\mathbf{x}_j, \mathbf{x}_i\right) = 0$$

and so $(\mathbf{x}_i, y_i)$ has functional margin 1. If, on the other hand, the value of α_i remains 0 as a result of the second update, we have

$$1 - y_i \sum_{j=1}^{\ell} \alpha_j y_j \kappa\left(\mathbf{x}_j, \mathbf{x}_i\right) < 0$$

implying $(\mathbf{x}_i, y_i)$ has functional margin greater than 1. It follows that at convergence the solution satisfies the KKT complementarity conditions for the alternative hard margin support vector machine of Algorithm 7.26 once the condition

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0$$

arising from a variable threshold has been removed. The algorithm can be adapted to handle a version of the 1-norm soft margin support vector

machine by introducing an upper bound on the value of α_i, while a version of the 2-norm support vector machine can be implemented by adding a constant to the diagonal of the kernel matrix.

Remark 7.60 [SMO algorithm] If we want to allow a variable threshold the updates must be made on a pair of examples, an approach that results in the SMO algorithm. The rate of convergence of both of these algorithms is strongly affected by the order in which the examples are chosen for updating. Heuristic measures such as the degree of violation of the KKT conditions can be used to ensure very effective convergence rates in practice. ■

On-line regression On-line learning algorithms are not restricted to classification problems. Indeed in the next chapter we will describe such an algorithm for ranking that will be useful in the context of collaborative filtering. The update rule for the kernel adatron algorithm also suggests a general methodology for creating on-line versions of the optimisations we have described. The objective function for the alternative hard margin SVM is

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j).$$

If we consider the gradient of this quantity with respect to an individual α_i we obtain

$$\frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i} = 1 - y_i \sum_{j=1}^{\ell} \alpha_j y_j \kappa(\mathbf{x}_j, \mathbf{x}_i)$$

making the first update of the kernel adatron algorithm equivalent to

$$\alpha_i \leftarrow \alpha_i + \frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_i}$$

making it a simple gradient ascent algorithm augmented with corrections to ensure that the additional constraints are satisfied. If, for example, we apply this same approach to the linear ε-insensitive loss version of the support vector regression algorithm with fixed offset 0, we obtain the algorithm.

Algorithm 7.61 [On-line support vector regression] On-line support vector regression is implemented in Code Fragment 7.14. ■

Input	training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$
Process	$\boldsymbol{\alpha} = 0$, $i = 0$, loss $= 0$
2	repeat
3	for $i = 1 : \ell$
4	$\hat{\alpha}_i \leftarrow \alpha_i$;
5	$\alpha_i \leftarrow \alpha_i + y_i - \varepsilon \operatorname{sgn}(\alpha_i) - \sum_{j=1}^{\ell} \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_i)$;
6	if $\hat{\alpha}_i \alpha_i < 0$ then $\alpha_i \leftarrow 0$;
7	end
8	until $\boldsymbol{\alpha}$ unchanged
9	$f(\mathbf{x}) = \sum_{j=1}^{\ell} \alpha_j \kappa(\mathbf{x}_j, \mathbf{x})$
Output	dual variables $\boldsymbol{\alpha}$, loss and function f

where for $\alpha_i = 0$, $\operatorname{sgn}(\alpha_i)$ is interpreted to be the number in $[-1, +1]$ that gives the update in line 5 the smallest absolute value.

Code Fragment 7.14. Pseudocode for the on-line support vector regression.

7.5 Summary

- The smallest hypersphere enclosing all points in the embedding space can be found by solving a convex quadratic program. This suggests a simple novelty-detection algorithm.
- The stability analysis suggests a better novelty detector may result from a smaller hypersphere containing a fixed fraction of points that minimises the sum of the distances to the external points. This can again be computed by a convex quadratic program. Its characteristic function can be written in terms of a kernel expansion, where only certain points have non-zero coefficients. They are called support vectors and because of the many zero coefficients the expansion is called 'sparse'.
- If there is a maximal margin hyperplane separating two sets of points in the embedding space, it can be found by solving a convex quadratic program. This gives the hard margin support vector machine classification algorithm.
- The stability analysis again suggests improved generalisation will frequently result from allowing a certain (prefixed) fraction of points to be 'margin' errors while minimising the sizes of those errors. This can again be found by solving a convex quadratic program and gives the well-known soft margin support vector machines. Also in this case, the kernel expansion of the classification function can be sparse, as a result of the Karush–Kuhn–Tucker conditions. The pre-image of this hyperplane in the input space can be very complex, depending on the choice of ker-

nel. Hence, these algorithms are able to optimise over highly nonlinear function classes through an application of the kernel trick.

- A nonlinear regression function that realises a trade-off between loss and smoothness can be found by solving a convex quadratic program. This corresponds to a regularised linear function in the embedding space. Fixing the regularization term to be the 2-norm of the linear function in the embedding space, different choices of loss can be made. The quadratic loss yields the nonlinear version of ridge regression introduced in Chapter 2. Both linear and quadratic ε-insensitive losses yield support vector machines for regression. Unlike ridge regression these again result in sparse kernel expansions of the solutions.
- The absence of local minima from the above algorithms marks a major departure from traditional systems such as neural networks, and jointly with sparseness properties makes it possible to create very efficient implementations.

7.6 Further reading and advanced topics

The systematic use of optimisation in pattern recognition dates back at least to Mangasarian's pioneering efforts [93] and possibly earlier. Many different authors have independently proposed algorithms for data classification or other tasks based on the theory of Lagrange multipliers, and this approach is now part of the standard toolbox in pattern analysis.

The problem of calculating the smallest sphere containing a set of data was first posed in the hard-margin case by [115], [20] for the purpose of calculating generalisation bounds that depend on the radius of the enclosing sphere. It was subsequently addressed by Tax and Duin [134] in a soft margin setting for the purpose of modeling the input distribution and hence detecting novelties. This approach to novelty detection was cast in a ν-SVM setting by Schölkopf *et al.* [117].

The problem of separating two sets of data with a maximal margin hyperplane has been independently addressed by a number of authors over a long period of time. Large margin hyperplanes in the input space were, for example, discussed by Duda and Hart [39], Cover [28], Smith [127], Vapnik *et al.* [144], [141], and several statistical mechanics papers (for example [3]). It is, however, the combination of this optimisation problem with kernels that produced support vector machines, as we discuss briefly below. See Chapter 6 of [32] for a more detailed reconstruction of the history of SVMs.

The key features of SVMs are the use of kernels, the absence of local minima, the sparseness of the solution and the capacity control obtained by

optimising the margin. Although many of these components were already used in different ways within machine learning, it is their combination that was first realised in the paper [16]. The use of slack variables for noise tolerance, tracing back to [13] and further to [127], was introduced to the SVM algorithm in 1995 in the paper of Cortes and Vapnik [27]. The ν-support vector algorithm for classification and regression is described in [120].

Extensive work has been done over more than a decade by a fast growing community of theoreticians and practitioners, and it would be difficult to document all the variations on this theme. In a way, this entire book is an attempt to systematise this body of literature.

Among many connections, it is worth emphasising the connection between SVM regression, ridge regression and regularisation networks. The concept of regularisation was introduced by Tikhonov [136], and was introduced into machine learning in the form of regularisation networks by Girosi *et al.* [48]. The relation between regularisation networks and support vector machines has been explored by a number of authors [47], [154], [129], [43].

Finally for a background on convex optimisation and Kuhn–Tucker theory see for example [92], and for a brief introduction see Chapter 5 of [32].

For constantly updated pointers to online literature and free software see the book's companion website: www.kernel-methods.net.