5

Elementary algorithms in feature space

In this chapter we show how to evaluate a number of properties of a data set in a kernel-defined feature space. The quantities we consider are of interest in their own right in data analysis, but they will also form building blocks towards the design of complex pattern analysis systems. Furthermore, the computational methods we develop will play an important role in subsequent chapters.

The quantities include the distance between two points, the centre of mass, the projections of data onto particular directions, the rank, the variance and covariance of projections of a set of data points, all measured in the feature space. We will go on to consider the distance between the centres of mass of two sets of data.

Through the development of these methods we will arrive at a number of algorithmic solutions for certain problems. We give Matlab code for normalising the data, centering the data in feature space, and standardising the different coordinates. Finally, we develop two pattern analysis algorithms, the first is a novelty-detection algorithm that comes with a theoretical guarantee on performance, while the second is a first kernelised version of the Fisher discriminant algorithm. This important pattern analysis algorithm is somewhat similar to the ridge regression algorithm already previewed in Chapter 2, but tackles classification and takes account of more subtle structure of the data.

5.1 Means and distances

Given a finite subset $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ of an input space X, a kernel $\kappa(\mathbf{x}, \mathbf{z})$ and a feature map $\boldsymbol{\phi}$ into a feature space F satisfying

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

let $\boldsymbol{\phi}(S) = \{\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_\ell)\}$ be the image of S under the map $\boldsymbol{\phi}$. Hence $\boldsymbol{\phi}(S)$ is a subset of the inner product space F. In this chapter we continue our investigation of the information that can be obtained about $\boldsymbol{\phi}(S)$ using only the inner product information contained in the kernel matrix $\mathbf{K}$ of kernel evaluations between all pairs of elements of S

$$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \quad i, j = 1, \ldots, \ell.$$

Working in a kernel-defined feature space means that we are not able to explicitly represent points. For example the image of an input point $\mathbf{x}$ is $\boldsymbol{\phi}(\mathbf{x})$, but we do not have access to the components of this vector, only to the evaluation of inner products between this point and the images of other points. Despite this handicap there is a surprising amount of useful information that can be gleaned about $\boldsymbol{\phi}(S)$.

Norm of feature vectors The simplest example already seen in Chapter 4 is the evaluation of the norm of $\boldsymbol{\phi}(\mathbf{x})$ that is given by

$$\|\boldsymbol{\phi}(\mathbf{x})\|_2 = \sqrt{\|\boldsymbol{\phi}(\mathbf{x})\|^2} = \sqrt{\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}) \rangle} = \sqrt{\kappa(\mathbf{x}, \mathbf{x})}.$$

Algorithm 5.1 [Normalisation] Using this observation we can now implement the normalisation transformation mentioned in Chapters 2 and 3 given by

$$\hat{\boldsymbol{\phi}}(\mathbf{x}) = \frac{\boldsymbol{\phi}(\mathbf{x})}{\|\boldsymbol{\phi}(\mathbf{x})\|}.$$

For two data points the transformed kernel $\hat{\kappa}$ is given by

$$\begin{aligned} \hat{\kappa}(\mathbf{x}, \mathbf{z}) &= \left\langle \hat{\boldsymbol{\phi}}(\mathbf{x}), \hat{\boldsymbol{\phi}}(\mathbf{z}) \right\rangle = \left\langle \frac{\boldsymbol{\phi}(\mathbf{x})}{\|\boldsymbol{\phi}(\mathbf{x})\|}, \frac{\boldsymbol{\phi}(\mathbf{z})}{\|\boldsymbol{\phi}(\mathbf{z})\|} \right\rangle = \frac{\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z}) \rangle}{\|\boldsymbol{\phi}(\mathbf{x})\| \|\boldsymbol{\phi}(\mathbf{z})\|} \quad (5.1) \\ &= \frac{\kappa(\mathbf{x}, \mathbf{z})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x}) \kappa(\mathbf{z}, \mathbf{z})}}. \end{aligned}$$

The corresponding transformation of the kernel matrix can be implemented by the operations given in Code Fragment 5.1. ■

```
% original kernel matrix stored in variable K
% output uses the same variable K
% D is a diagonal matrix storing the inverse of the norms
D = diag(1./sqrt(diag(K)));
K = D * K * D;
```

Code Fragment 5.1. Matlab code normalising a kernel matrix.

We can also evaluate the norms of linear combinations of images in the feature space. For example we have

$$\begin{aligned}\left\|\sum_{i=1}^{\ell}\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)\right\|^2 &= \left\langle\sum_{i=1}^{\ell}\alpha_i\boldsymbol{\phi}(\mathbf{x}_i),\sum_{j=1}^{\ell}\alpha_j\boldsymbol{\phi}(\mathbf{x}_j)\right\rangle\\ &= \sum_{i=1}^{\ell}\alpha_i\sum_{j=1}^{\ell}\alpha_j\left\langle\boldsymbol{\phi}(\mathbf{x}_i),\boldsymbol{\phi}(\mathbf{x}_j)\right\rangle\\ &= \sum_{i,j=1}^{\ell}\alpha_i\alpha_j\kappa(\mathbf{x}_i,\mathbf{x}_j).\end{aligned}$$

Distance between feature vectors A special case of the norm is the length of the line joining two images $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{z})$, which can be computed as

$$\begin{aligned}\|\boldsymbol{\phi}(\mathbf{x})-\boldsymbol{\phi}(\mathbf{z})\|^2 &= \left\langle\boldsymbol{\phi}(\mathbf{x})-\boldsymbol{\phi}(\mathbf{z}),\boldsymbol{\phi}(\mathbf{x})-\boldsymbol{\phi}(\mathbf{z})\right\rangle\\ &= \left\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x})\right\rangle-2\left\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{z})\right\rangle+\left\langle\boldsymbol{\phi}(\mathbf{z}),\boldsymbol{\phi}(\mathbf{z})\right\rangle\\ &= \kappa(\mathbf{x},\mathbf{x})-2\kappa(\mathbf{x},\mathbf{z})+\kappa(\mathbf{z},\mathbf{z}).\end{aligned}$$

Norm and distance from the centre of mass As a more complex and useful example consider the centre of mass of the set $\boldsymbol{\phi}(S)$. This is the vector

$$\boldsymbol{\phi}_S=\frac{1}{\ell}\sum_{i=1}^{\ell}\boldsymbol{\phi}(\mathbf{x}_i).$$

As with all points in the feature space we will not have an explicit vector representation of this point. However, in this case there may also not exist a point in X whose image under $\boldsymbol{\phi}$ is $\boldsymbol{\phi}_S$. In other words, we are now considering points that potentially lie outside $\boldsymbol{\phi}(X)$, that is the image of the input space X under the feature map $\boldsymbol{\phi}$.

Despite this apparent inaccessibility of the point $\boldsymbol{\phi}_S$, we can compute its

norm using only evaluations of the kernel on the inputs

$$\begin{aligned}\|\boldsymbol{\phi}_S\|_2^2 &= \langle \boldsymbol{\phi}_S, \boldsymbol{\phi}_S \rangle = \left\langle \frac{1}{\ell}\sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i), \frac{1}{\ell}\sum_{j=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_j) \right\rangle \\ &= \frac{1}{\ell^2}\sum_{i,j=1}^{\ell} \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle = \frac{1}{\ell^2}\sum_{i,j=1}^{\ell} \kappa(\mathbf{x}_i, \mathbf{x}_j).\end{aligned}$$

Hence, the square of the norm of the centre of mass is equal to the average of the entries in the kernel matrix. Incidentally this implies that this sum is greater than or equal to zero, with equality if the centre of mass is at the origin of the coordinate system. Similarly, we can compute the distance of the image of a point $\mathbf{x}$ from the centre of mass $\boldsymbol{\phi}_S$

$$\begin{aligned}\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_S\|^2 &= \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}) \rangle + \langle \boldsymbol{\phi}_S, \boldsymbol{\phi}_S \rangle - 2\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}_S \rangle \\ &= \kappa(\mathbf{x}, \mathbf{x}) + \frac{1}{\ell^2}\sum_{i,j=1}^{\ell} \kappa(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{\ell}\sum_{i=1}^{\ell} \kappa(\mathbf{x}, \mathbf{x}_i). \quad (5.2)\end{aligned}$$

Expected distance from the centre of mass Following the same approach it is also possible to express the expected squared distance of a point in a set from its mean

$$\begin{aligned}\frac{1}{\ell}\sum_{s=1}^{\ell} \|\boldsymbol{\phi}(\mathbf{x}_s) - \boldsymbol{\phi}_S\|^2 &= \frac{1}{\ell}\sum_{s=1}^{\ell} \kappa(\mathbf{x}_s, \mathbf{x}_s) + \frac{1}{\ell^2}\sum_{i,j=1}^{\ell} \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ &\quad - \frac{2}{\ell^2}\sum_{i,s=1}^{\ell} \kappa(\mathbf{x}_s, \mathbf{x}_i) \quad (5.3) \\ &= \frac{1}{\ell}\sum_{s=1}^{\ell} \kappa(\mathbf{x}_s, \mathbf{x}_s) - \frac{1}{\ell^2}\sum_{i,j=1}^{\ell} \kappa(\mathbf{x}_i, \mathbf{x}_j). \quad (5.4)\end{aligned}$$

Hence, the average squared distance of points to their centre of mass is the average of the diagonal entries of the kernel matrix minus the average of all the entries.

Properties of the centre of mass If we translate the origin of the feature space, the norms of the training points alter, but the left-hand side of equation (5.4) does not change. If the centre of mass is at the origin, then, as we observed above, the entries in the matrix will sum to zero. Hence, moving the origin to the centre of mass minimises the first term on the right-hand side of equation (5.4), corresponding to the sum of the squared norms of the

points. This also implies the following proposition that will prove useful in Chapter 8.

Proposition 5.2 *The centre of mass $\boldsymbol{\phi}_S$ of a set of points $\boldsymbol{\phi}(S)$ solves the following optimisation problem*

$$\min_{\mu} \frac{1}{\ell} \sum_{s=1}^{\ell} \|\boldsymbol{\phi}(\mathbf{x}_s) - \mu\|^2 .$$

Proof Consider moving the origin to the point μ. The quantity to be optimised corresponds to the first term on the right-hand side of equation (5.4). Since the left-hand side does not depend on μ, the quantity will be minimised by minimising the second term on the right-hand side, something that is achieved by taking $\mu = \boldsymbol{\phi}_S$. The result follows. □

Centering data Since the first term on the right-hand side of equation (5.4) is the trace of the matrix divided by its size, moving the origin to the centre of mass also minimises the average eigenvalue. As announced in Chapter 3 we can perform this operation implicitly by transforming the kernel matrix. This follows from the fact that the new feature map is given by

$$\hat{\boldsymbol{\phi}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_S = \boldsymbol{\phi}(\mathbf{x}) - \frac{1}{\ell} \sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i).$$

Hence, the kernel for the transformed space is

$$\begin{aligned} \hat{\kappa}(\mathbf{x}, \mathbf{z}) &= \left\langle \hat{\boldsymbol{\phi}}(\mathbf{x}), \hat{\boldsymbol{\phi}}(\mathbf{z}) \right\rangle = \left\langle \boldsymbol{\phi}(\mathbf{x}) - \frac{1}{\ell} \sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{z}) - \frac{1}{\ell} \sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i) \right\rangle \\ &= \kappa(\mathbf{x}, \mathbf{z}) - \frac{1}{\ell} \sum_{i=1}^{\ell} \kappa(\mathbf{x}, \mathbf{x}_i) - \frac{1}{\ell} \sum_{i=1}^{\ell} \kappa(\mathbf{z}, \mathbf{x}_i) + \frac{1}{\ell^2} \sum_{i,j=1}^{\ell} \kappa(\mathbf{x}_i, \mathbf{x}_j). \end{aligned}$$

Expressed as an operation on the kernel matrix this can be written as

$$\hat{\mathbf{K}} = \mathbf{K} - \frac{1}{\ell} \mathbf{j}\mathbf{j}'\mathbf{K} - \frac{1}{\ell} \mathbf{K}\mathbf{j}\mathbf{j}' + \frac{1}{\ell^2} \left(\mathbf{j}'\mathbf{K}\mathbf{j}\right) \mathbf{j}\mathbf{j}',$$

where $\mathbf{j}$ is the all 1s vector. We have the following algorithm.

Algorithm 5.3 [Centering data] We can centre the data in the feature space with the short sequence of operations given in Code Fragment 5.2. ■

```
% original kernel matrix stored in variable K
% output uses the same variable K
% K is of dimension ell x ell
% D is a row vector storing the column averages of K
% E is the average of all the entries of K
ell = size(K,1);
D = sum(K) / ell;
E = sum(D) / ell;
J = ones(ell,1) * D;
K = K - J - J' + E * ones(ell, ell);
```

Code Fragment 5.2. Matlab code for centering a kernel matrix.

The stability of centering The example of centering raises the question of how reliably we can estimate the centre of mass from a training sample or in other words how close our sample centre will be to the true expectation

$$\mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})] = \int_X \boldsymbol{\phi}(\mathbf{x}) dP(\mathbf{x}).$$

Our analysis in Chapter 4 bounded the expected value of the quantity

$$g(S) = \|\boldsymbol{\phi}_S - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\|.$$

There it was shown that with probability at least $1-\delta$ over the choice of a random sample of ℓ points, we have

$$g(S) \leq \sqrt{\frac{2R^2}{\ell}} \left(\sqrt{2} + \sqrt{\ln \frac{1}{\delta}} \right), \tag{5.5}$$

assuring us that with high probability our sample does indeed give a good estimate of $\mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]$ in a way that does not depend on the dimension of the feature space, where the support of the distribution is contained in a ball of radius R around the origin.

5.1.1 A simple algorithm for novelty-detection

Centering suggests a simple novelty-detection algorithm. If we consider the training set as a sample of points providing an estimate of the distances $d_1, \ldots, d_\ell$ from the point $\mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]$, where

$$d_i = \|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\|,$$

we can bound the probability that a new random point $\mathbf{x}_{\ell+1}$ satisfies

$$d_{\ell+1} = \|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\| > \max_{1 \leq i \leq \ell} d_i,$$

with

$$P\left\{\|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\| > \max_{1\leq i\leq \ell} d_i\right\} = P\left\{\max_{1\leq i\leq \ell+1} d_i = d_{\ell+1} \neq \max_{1\leq i\leq \ell} d_i\right\} \leq \frac{1}{\ell+1},$$

by the symmetry of the i.i.d. assumption. Though we cannot compute the distance to the point $\mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]$, we can, by equation (5.2), compute

$$\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_S\| = \sqrt{\kappa(\mathbf{x},\mathbf{x}) + \frac{1}{\ell^2}\sum_{i,j=1}^{\ell}\kappa(\mathbf{x}_i,\mathbf{x}_j) - \frac{2}{\ell}\sum_{i=1}^{\ell}\kappa(\mathbf{x},\mathbf{x}_i)}. \tag{5.6}$$

Then we can with probability $1-\delta$ estimate $\|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\|$ using the triangle inequality and (5.5)

$$\begin{aligned} d_{\ell+1} &= \|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\| \\ quad &\geq \|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \boldsymbol{\phi}_S\| - \|\boldsymbol{\phi}_S - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\| \\ &\geq \|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \boldsymbol{\phi}_S\| - \sqrt{\frac{2R^2}{\ell}}\left(\sqrt{2} + \sqrt{\ln\frac{1}{\delta}}\right). \end{aligned}$$

Similarly, we have that for $i = 1,\ldots,\ell$

$$d_i = \|\boldsymbol{\phi}(\mathbf{x}_i) - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\| \leq \|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}_S\| + \|\boldsymbol{\phi}_S - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]\|.$$

We now use the inequalities to provide a bound on the probability that a test point lies outside a ball centred on the empirical centre of mass. Effectively we choose its radius to ensure that with high probability it contains the ball of radius $\max_{1\leq i\leq \ell} d_i$ with centre $\mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(\mathbf{x})]$. With probability $1-\delta$ we have that

$$\begin{aligned} P\left\{\|\boldsymbol{\phi}(\mathbf{x}_{\ell+1}) - \boldsymbol{\phi}_S\| > \max_{1\leq i\leq \ell}\|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}_S\| + 2\sqrt{\frac{2R^2}{\ell}}\left(\sqrt{2} + \sqrt{\ln\frac{1}{\delta}}\right)\right\} \\ \leq P\left\{\max_{1\leq i\leq \ell+1} d_i = d_{\ell+1} \neq \max_{1\leq i\leq \ell} d_i\right\} \leq \frac{1}{\ell+1}. \end{aligned} \tag{5.7}$$

Using $\mathcal{H}(x)$ to denote the Heaviside function we have in the notation of Chapter 1 a pattern analysis algorithm that returns the pattern function

$$f(\mathbf{x}) = \mathcal{H}\left(\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_S\| - \max_{1\leq i\leq \ell}\|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}_S\| - 2\sqrt{\frac{2R^2}{\ell}}\left(\sqrt{2} + \sqrt{\ln\frac{1}{\delta}}\right)\right),$$

since by inequality (5.7) with probability $1 - \delta$ the expectation is bounded by

$$\mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] \leq 1/(\ell + 1).$$

Hence, we can reject as anomalous data items satisfying $f(\mathbf{x}) = 1$, and reject authentic examples with probability at most $1/(\ell+1)$. This gives rise to the following novelty-detection algorithm.

Algorithm 5.4 [Simple novelty detection] An implementation of the simple novelty-detection algorithm is given in Code Fragment 5.3. ■

```
% K kernel matrix of training points
% inner products between ell training and t test points
%   stored in matrix Ktest of dimension (ell + 1) x t
%   last entry in each column is inner product with itself
% confidence parameter
delta = 0.01
% first compute distances of data to centre of mass
% D is a row vector storing the column averages of K
% E is the average of all the entries of K
ell = size(K,1);
D = sum(K) / ell;
E = sum(D) / ell;
traindist2 = diag(K) - 2 * D' + E * ones(ell, 1);
maxdist = sqrt(max(traindist2));
% compute the estimation error of empirical centre of mass
esterr = sqrt(2*max(diag(K))/ell)*(sqrt(2) + sqrt(log(1/delta)));
% compute resulting threshold
threshold = maxdist + 2 * esterr;
threshold = threshold * threshold;
% now compute distances of test data
t = size(Ktest,2);
Dtest = sum(Ktest(1:ell,:)) / ell;
testdist2 = Ktest(ell+1,:) - 2 * Dtest + E * ones(1, t);
% indices of novel test points are now
novelindices = find ( testdist2 > threshold )
```

Code Fragment 5.3. Matlab code for simple novelty detection algorithm.

The pattern function is unusual in that it is not always a thresholded linear function in the kernel-defined feature space, though by equation (5.2) if the feature space is normalised the function can be represented in the standard form. The algorithm considers a sphere containing the data centred on the centre of mass of the data sample. Figure 5.1 illustrates the spheres for data generated according to a spherical two-dimensional Gaussian distribution.

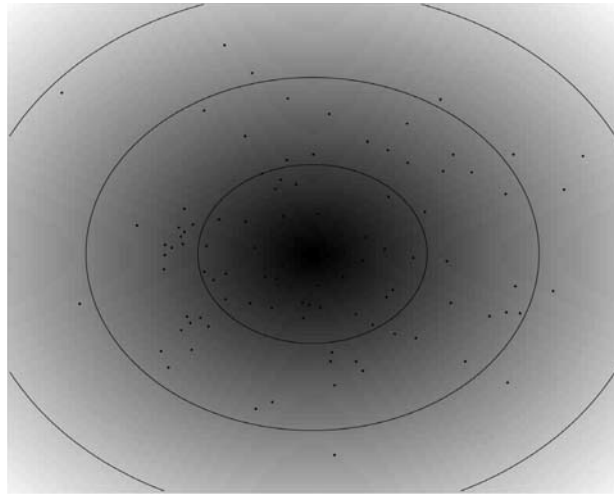

Fig. 5.1. Novelty detection spheres centred on the empirical centre of mass.

In Chapter 7 we will consider letting the centre of the hypersphere shift in order to reduce its radius. This approach results in a state-of-the-art method for novelty-detection.

Stability of novelty-detection The following proposition assesses the stability of the basic novelty-detection Algorithm 5.4.

Proposition 5.5 *Suppose that we wish to perform novelty-detection based on a training sample*

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

using the feature space implicitly defined by the kernel $\kappa(\mathbf{x}, \mathbf{z})$*; let* $f(\mathbf{x})$ *be given by*

$$f(\mathbf{x}) = \mathcal{H}\left(\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_S\| - \max_{1\leq i\leq \ell}\|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}_S\| - 2\sqrt{\frac{2R^2}{\ell}}\left(\sqrt{2} + \sqrt{\ln\frac{1}{\delta}}\right)\right)$$

where $\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_S\|$ *can be computed using equation (5.6). Then the function* $f(\mathbf{x})$ *is equivalent to identifying novel points that are further from the centre of mass in the feature space than any of the training points. Hence, with probability* $1 - \delta$ *over the random draw of the training set, any points*

drawn according to the same distribution will have $f(\mathbf{x}) = 1$ *with probability less than* $1/(\ell+1)$.

5.1.2 A simple algorithm for classification

If we consider now the case of binary classification, we can divide the training set S into two sets S_+ and S_- containing the positive and negative examples respectively. One could now use the above methodology to compute the distance $d_+(\mathbf{x}) = \left\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_{S_+}\right\|$ of a test point $\mathbf{x}$ from the centre of mass ϕ_{S_+} of S_+ and the distance $d_-(\mathbf{x}) = \left\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}_{S_-}\right\|$ from the centre of mass of the negative examples. A simple classification rule would be to assign $\mathbf{x}$ to the class corresponding to the smaller distance

$$h(\mathbf{x}) = \begin{cases} +1, & \text{if } d_-(\mathbf{x}) > d_+(\mathbf{x}); \\ -1, & \text{otherwise.} \end{cases}$$

We can express the function $h(\mathbf{x})$ in terms of the sign function

$$\begin{aligned} h(\mathbf{x}) &= \operatorname{sgn}\left(\left\|\boldsymbol{\phi}(\mathbf{x}) - \bar{\phi}_{S_-}\right\|^2 - \left\|\boldsymbol{\phi}(\mathbf{x}) - \bar{\phi}_{S_+}\right\|^2\right) \\ &= \operatorname{sgn}\left(-\kappa(\mathbf{x},\mathbf{x}) - \frac{1}{\ell_+^2}\sum_{i,j=1}^{\ell_+}\kappa(\mathbf{x}_i,\mathbf{x}_j) + \frac{2}{\ell_+}\sum_{i=1}^{\ell_+}\kappa(\mathbf{x},\mathbf{x}_i)\right. \\ &\qquad \left. +\kappa(\mathbf{x},\mathbf{x}) + \frac{1}{\ell_-^2}\sum_{i,j=\ell_++1}^{\ell_++\ell_-}\kappa(\mathbf{x}_i,\mathbf{x}_j) - \frac{2}{\ell_-}\sum_{i=\ell_++1}^{\ell_++\ell_-}\kappa(\mathbf{x},\mathbf{x}_i)\right) \\ &= \operatorname{sgn}\left(\frac{1}{\ell_+}\sum_{i=1}^{\ell_+}\kappa(\mathbf{x},\mathbf{x}_i) - \frac{1}{\ell_-}\sum_{i=\ell_++1}^{\ell}\kappa(\mathbf{x},\mathbf{x}_i) - b\right), \end{aligned}$$

where we have assumed that the positive examples are indexed from 1 to ℓ_+ and the negative examples from $\ell_+ + 1$ to $\ell_+ + \ell_- = \ell$ and where b is a constant being half of the difference between the average entry of the positive examples kernel matrix and the average entry of the negative examples kernel matrix. This gives the following algorithm.

Algorithm 5.6 [Parzen based classifier] The simple Parzen based classifier algorithm is as follows:

input	Data $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$.
process	$\alpha_i^+ = \ell_+^{-1}$ if $y_i = +1$, 0 otherwise.
2	$\alpha_i^- = \ell_-^{-1}$ if $y_i = -1$, 0 otherwise.
3	$b = 0.5\left(\boldsymbol{\alpha}^{+\prime}\mathbf{K}\boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^{-\prime}\mathbf{K}\boldsymbol{\alpha}^-\right)$
4	$\boldsymbol{\alpha} = \boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-$;
5	$h(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}) - b\right)$
output	Function h, dual variables $\boldsymbol{\alpha}$ and offset b.

■

If the origin of the feature space is equidistant from the two centres of mass, the offset b will be zero since the average entry of the kernel matrix is equal to the square of the norm of the centre of mass.

Note that $h(\mathbf{x})$ is a thresholded linear function in the feature space with weight vector given by

$$\mathbf{w} = \frac{1}{\ell_+}\sum_{i=1}^{\ell_+} \boldsymbol{\phi}(\mathbf{x}_i) - \frac{1}{\ell_-}\sum_{i=\ell_++1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i).$$

This function is the difference in likelihood of the Parzen window density estimator for positive and negative examples. The name derives from viewing the kernel $\kappa(\cdot,\cdot)$ as a Parzen window that can be used to estimate the input densities for the positive and negative empirical distributions. This is natural when for example considering the Gaussian kernel.

Remark 5.7 [On stability analysis] We will not present a stability bound for this classifier, though one could apply the novelty-detection argument for the case where a new example was outside the novelty-detection pattern function derived for its class. In this case we could assert with high confidence that it belonged to the other class. ■

Consideration of the distances to the centre of mass of a dataset has led to some simple algorithms for both novelty-detection and classification. They are, however, constrained by not being able to take into account information about the spread of the data. In Section 5.3 we will investigate how the variance of the data can also be estimated using only information contained in the kernel matrix. First, however, we turn our attention to projections.

5.2 Computing projections: Gram–Schmidt, QR and Cholesky

The basic classification function of the previous section had the form of a thresholded linear function

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

where the weight vector $\mathbf{w}$ had the form

$$\mathbf{w} = \frac{1}{\ell_+}\sum_{i=1}^{\ell_+}\boldsymbol{\phi}(\mathbf{x}_i) - \frac{1}{\ell_-}\sum_{i=\ell_++1}^{\ell}\boldsymbol{\phi}(\mathbf{x}_i).$$

Hence, the computation only requires knowledge of the inner product between two feature space vectors.

The projection $P_{\mathbf{w}}\left(\boldsymbol{\phi}(\mathbf{x})\right)$ of a vector $\boldsymbol{\phi}(\mathbf{x})$ onto the vector $\mathbf{w}$ is given as

$$P_{\mathbf{w}}\left(\boldsymbol{\phi}(\mathbf{x})\right) = \frac{\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x})\rangle}{\|\mathbf{w}\|^2}\mathbf{w}.$$

This example illustrates a general principle that also enables us to compute projections of vectors in the feature space. For example given a general vector

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

we can compute the norm of the projection $P_{\mathbf{w}}\left(\boldsymbol{\phi}(\mathbf{x})\right)$ of the image of a point $\mathbf{x}$ onto the vector $\mathbf{w}$ as

$$\left\|P_{\mathbf{w}}\left(\boldsymbol{\phi}(\mathbf{x})\right)\right\| = \frac{\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x})\rangle}{\|\mathbf{w}\|} = \frac{\sum_{i=1}^{\ell}\alpha_i\kappa\left(\mathbf{x}_i, \mathbf{x}\right)}{\sqrt{\sum_{i,j=1}^{\ell}\alpha_i\alpha_j\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)}}.$$

Using Pythagoras's theorem allows us to compute the distance of the point from its projection as

$$\begin{aligned}\left\|P_{\mathbf{w}}\left(\boldsymbol{\phi}(\mathbf{x})\right) - \boldsymbol{\phi}(\mathbf{x})\right\|^2 &= \|\boldsymbol{\phi}(\mathbf{x})\|^2 - \left\|P_{\mathbf{w}}\left(\boldsymbol{\phi}(\mathbf{x})\right)\right\|^2 \\ &= \kappa\left(\mathbf{x}, \mathbf{x}\right) - \frac{\left(\sum_{i=1}^{\ell}\alpha_i\kappa\left(\mathbf{x}_i, \mathbf{x}\right)\right)^2}{\sum_{i,j=1}^{\ell}\alpha_i\alpha_j\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)}.\end{aligned}$$

If we have a set of orthonormal vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$ with corresponding dual representations given by $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^k$, we can compute the orthogonal projection $P_V\left(\boldsymbol{\phi}(\mathbf{x})\right)$ of a point $\boldsymbol{\phi}(\mathbf{x})$ into the subspace V spanned by

$\mathbf{w}_1, \ldots, \mathbf{w}_k$ as

$$P_V(\boldsymbol{\phi}(\mathbf{x})) = \left(\sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \kappa(\mathbf{x}_i, \mathbf{x}) \right)_{j=1}^{k},$$

where we have used the vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$ as a basis for V.

Definition 5.8 A *projection* is a mapping P satisfying

$$P(\boldsymbol{\phi}(\mathbf{x})) = P^2(\boldsymbol{\phi}(\mathbf{x})) \text{ and } \langle P(\boldsymbol{\phi}(\mathbf{x})), \boldsymbol{\phi}(\mathbf{x}) - P(\boldsymbol{\phi}(\mathbf{x})) \rangle = 0,$$

with its dimension $\dim(P)$ given by the dimension of the image of P. The orthogonal projection to P is given by

$$P^{\perp}(\boldsymbol{\phi}(\mathbf{x})) = \boldsymbol{\phi}(\mathbf{x}) - P(\boldsymbol{\phi}(\mathbf{x}))$$

and projects the data onto the orthogonal complement of the image of P, so that $\dim(P) + \dim\left(P^{\perp}\right) = N$, the dimension of the feature space. ■

Remark 5.9 [Orthogonal projections] It is not hard to see that the orthogonal projection is indeed a projection, since

$$P^{\perp}\left(P^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right) = P^{\perp}(\boldsymbol{\phi}(\mathbf{x})) - P\left(P^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right) = P^{\perp}(\boldsymbol{\phi}(\mathbf{x})),$$

while

$$\begin{aligned} &\left\langle P^{\perp}(\boldsymbol{\phi}(\mathbf{x})), \boldsymbol{\phi}(\mathbf{x}) - P^{\perp}(\boldsymbol{\phi}(\mathbf{x})) \right\rangle \\ &= \left\langle P^{\perp}(\boldsymbol{\phi}(\mathbf{x})), \boldsymbol{\phi}(\mathbf{x}) - (\boldsymbol{\phi}(\mathbf{x}) - P(\boldsymbol{\phi}(\mathbf{x}))) \right\rangle \\ &= \langle (\boldsymbol{\phi}(\mathbf{x}) - P(\boldsymbol{\phi}(\mathbf{x}))), P(\boldsymbol{\phi}(\mathbf{x})) \rangle = 0. \end{aligned}$$

■

Projections and deflations The projection $P_{\mathbf{w}}(\boldsymbol{\phi}(\mathbf{x}))$ of $\boldsymbol{\phi}(\mathbf{x})$ onto $\mathbf{w}$ introduced above are onto a 1-dimensional subspace defined by the vector $\mathbf{w}$. If we assume that $\mathbf{w}$ is normalised, $P_{\mathbf{w}}(\boldsymbol{\phi}(\mathbf{x}))$ can also be expressed as

$$P_{\mathbf{w}}(\boldsymbol{\phi}(\mathbf{x})) = \mathbf{w}\mathbf{w}'\boldsymbol{\phi}(\mathbf{x}).$$

Hence, its orthogonal projection $P_{\mathbf{w}}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))$ can be expressed as

$$P_{\mathbf{w}}^{\perp}(\boldsymbol{\phi}(\mathbf{x})) = \left(\mathbf{I} - \mathbf{w}\mathbf{w}'\right)\boldsymbol{\phi}(\mathbf{x}).$$

If we have a data matrix $\mathbf{X}$ with rows $\boldsymbol{\phi}(\mathbf{x}_i)$, $i = 1, \ldots, \ell$, then deflating the matrix $\mathbf{X}'\mathbf{X}$ with respect to one of its eigenvectors $\mathbf{w}$ is equivalent to projecting the data using $P_{\mathbf{w}}^{\perp}$. This follows from the observation that projecting

the data creates the new data matrix

$$\tilde{\mathbf{X}} = \mathbf{X}\left(\mathbf{I} - \mathbf{w}\mathbf{w}'\right)' = \mathbf{X}\left(\mathbf{I} - \mathbf{w}\mathbf{w}'\right), \tag{5.8}$$

so that

$$\begin{aligned}\tilde{\mathbf{X}}'\tilde{\mathbf{X}} &= \left(\mathbf{I} - \mathbf{w}\mathbf{w}'\right)\mathbf{X}'\mathbf{X}\left(\mathbf{I} - \mathbf{w}\mathbf{w}'\right)\\ &= \mathbf{X}'\mathbf{X} - \mathbf{w}\mathbf{w}'\mathbf{X}'\mathbf{X} - \mathbf{X}'\mathbf{X}\mathbf{w}\mathbf{w}' + \mathbf{w}\mathbf{w}'\mathbf{X}'\mathbf{X}\mathbf{w}\mathbf{w}'\\ &= \mathbf{X}'\mathbf{X} - \lambda\mathbf{w}\mathbf{w}' - \boldsymbol{\lambda}\mathbf{w}\mathbf{w}' + \lambda\mathbf{w}\mathbf{w}'\mathbf{w}\mathbf{w}'\\ &= \mathbf{X}'\mathbf{X} - \lambda\mathbf{w}\mathbf{w}',\end{aligned}$$

where λ is the eigenvalue corresponding to $\mathbf{w}$.

The actual spread of the data may not be spherical as is implicitly assumed in the novelty detector derived in the previous section. We may indeed observe that the data lies in a subspace of the feature space of lower dimensionality.

We now consider how to find an orthonormal basis for such a subspace. More generally we seek a subspace that fits the data in the sense that the distances between data items and their projections into the subspace are small. Again we would like to compute the projections of points into subspaces of the feature space implicitly using only information provided by the kernel.

Gram–Schmidt orthonormalisation We begin by considering a well-known method of deriving an orthonormal basis known as the *Gram–Schmidt* procedure. Given a sequence of linearly independent vectors the method creates the basis by orthogonalising each vector to all of the earlier vectors. Hence, if we are given the vectors

$$\boldsymbol{\phi}\left(\mathbf{x}_1\right), \boldsymbol{\phi}\left(\mathbf{x}_2\right), \ldots, \boldsymbol{\phi}\left(\mathbf{x}_\ell\right),$$

the first basis vector is chosen to be

$$\mathbf{q}_1 = \frac{\boldsymbol{\phi}\left(\mathbf{x}_1\right)}{\|\boldsymbol{\phi}\left(\mathbf{x}_1\right)\|}.$$

The ith vector is then obtained by subtracting from $\boldsymbol{\phi}\left(\mathbf{x}_i\right)$ multiples of $\mathbf{q}_1, \ldots, \mathbf{q}_{i-1}$ in order to ensure it becomes orthogonal to each of them

$$\boldsymbol{\phi}\left(\mathbf{x}_i\right) \longrightarrow \boldsymbol{\phi}\left(\mathbf{x}_i\right) - \sum_{j=1}^{i-1} \left\langle \mathbf{q}_j, \boldsymbol{\phi}\left(\mathbf{x}_i\right)\right\rangle \mathbf{q}_j = \left(\mathbf{I} - \mathbf{Q}_{i-1}\mathbf{Q}_{i-1}'\right)\boldsymbol{\phi}\left(\mathbf{x}_i\right),$$

where $\mathbf{Q}_i$ is the matrix whose i columns are the first i vectors $\mathbf{q}_1, \ldots, \mathbf{q}_i$. The matrix $\left(\mathbf{I} - \mathbf{Q}_i\mathbf{Q}_i'\right)$ is a projection matrix onto the orthogonal complement

of the space spanned by the first i vectors $\mathbf{q}_1, \ldots, \mathbf{q}_i$. Finally, if we let

$$\nu_i = \left\| \left(\mathbf{I} - \mathbf{Q}_{i-1}\mathbf{Q}'_{i-1} \right) \boldsymbol{\phi}\left(\mathbf{x}_i\right) \right\|,$$

the next basis vector is obtained by normalising the projection

$$\mathbf{q}_i = \nu_i^{-1} \left(\mathbf{I} - \mathbf{Q}_{i-1}\mathbf{Q}'_{i-1} \right) \boldsymbol{\phi}\left(\mathbf{x}_i\right).$$

It follows that

$$\begin{aligned} \boldsymbol{\phi}\left(\mathbf{x}_i\right) &= \mathbf{Q}_{i-1}\mathbf{Q}'_{i-1}\boldsymbol{\phi}\left(\mathbf{x}_i\right) + \nu_i \mathbf{q}_i = \mathbf{Q}_i \begin{pmatrix} \mathbf{Q}'_{i-1}\boldsymbol{\phi}\left(\mathbf{x}_i\right) \\ \nu_i \end{pmatrix} \\ &= \mathbf{Q} \begin{pmatrix} \mathbf{Q}'_{i-1}\boldsymbol{\phi}\left(\mathbf{x}_i\right) \\ \nu_i \\ \mathbf{0}_{\ell-i} \end{pmatrix} = \mathbf{Q}\mathbf{r}_i, \end{aligned}$$

where $\mathbf{Q} = \mathbf{Q}_\ell$ is the matrix containing all the vectors $\mathbf{q}_i$ as columns. This implies that the matrix $\mathbf{X}$ containing the data vectors as rows can be decomposed as

$$\mathbf{X}' = \mathbf{Q}\mathbf{R},$$

where $\mathbf{R}$ is an upper triangular matrix with ith column

$$\mathbf{r}_i = \begin{pmatrix} \mathbf{Q}'_{i-1}\boldsymbol{\phi}\left(\mathbf{x}_i\right) \\ \nu_i \\ \mathbf{0}_{\ell-i} \end{pmatrix}.$$

We can also view $\mathbf{r}_i$ as the respresentation of $\mathbf{x}_i$ in the basis

$$\left\{\mathbf{q}_1, \ldots, \mathbf{q}_\ell\right\}.$$

QR-decomposition This is the well-known *QR-decomposition* of the matrix $\mathbf{X}'$ into the product of an orthonormal matrix $\mathbf{Q}$ and upper triangular matrix $\mathbf{R}$ with positive diagonal entries.

We now consider the application of this technique in a kernel-defined feature space. Consider the matrix $\mathbf{X}$ whose rows are the projections of a dataset

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

into a feature space defined by a kernel κ with corresponding feature mapping $\boldsymbol{\phi}$. Applying the Gram–Schmidt method in the feature space would lead to the decomposition

$$\mathbf{X}' = \mathbf{Q}\mathbf{R},$$

defined above. This gives the following decomposition of the kernel matrix

$$\mathbf{K} = \mathbf{X}\mathbf{X}' = \mathbf{R}'\mathbf{Q}'\mathbf{Q}\mathbf{R} = \mathbf{R}'\mathbf{R}.$$

Definition 5.10 This is the *Cholesky decomposition* of a positive semi-definite matrix into the product of a lower triangular and upper triangular matrix that are transposes of each other.

Since the Cholesky decomposition is unique, performing a Cholesky decomposition of the kernel matrix is equivalent to performing Gram–Schmidt orthonormalisation in the feature space and hence we can view Cholesky decomposition as the dual implementation of the Gram–Schmidt orthonormalisation. ■

Cholesky implementation The computation of the (j, i)th entry in the matrix $\mathbf{R}$ corresponds to evaluating the inner product between the ith vector $\boldsymbol{\phi}(\mathbf{x}_i)$ with the jth basis vector $\mathbf{q}_j$, for $i > j$. Since we can decompose $\boldsymbol{\phi}(\mathbf{x}_i)$ into a component lying in the subspace spanned by the basis vectors up to the jth for which we have already computed the inner products and the perpendicular complement, this inner product is given by

$$\nu_j \langle \mathbf{q}_j, \boldsymbol{\phi}(\mathbf{x}_i) \rangle = \langle \boldsymbol{\phi}(\mathbf{x}_j), \boldsymbol{\phi}(\mathbf{x}_i) \rangle - \sum_{t=1}^{j-1} \langle \mathbf{q}_t, \boldsymbol{\phi}(\mathbf{x}_j) \rangle \langle \mathbf{q}_t, \boldsymbol{\phi}(\mathbf{x}_i) \rangle,$$

which corresponds to the Cholesky computation performed for $j = 1, \ldots, \ell$

$$\mathbf{R}_{ji} = \nu_j^{-1} \left(\mathbf{K}_{ji} - \sum_{t=1}^{j-1} \mathbf{R}_{tj}\mathbf{R}_{ti} \right), \; i = j+1, \ldots, \ell,$$

where ν_j is obtained by keeping track of the residual norm squared d_i of the vectors in the orthogonal complement. This is done by initialising with the diagonal of the kernel matrix

$$d_i = \mathbf{K}_{ii}$$

and updating with

$$d_i \leftarrow d_i - \mathbf{R}_{ji}^2$$

as the ith entry is computed. The value of ν_j is then the residual norm of the next vector; that is

$$\nu_j = \sqrt{d_j}.$$

Note that the new representation of the data as the columns of the matrix $\mathbf{R}$ gives rise to exactly the same kernel matrix. Hence, we have found a new projection function

$$\hat{\phi} : \mathbf{x}_i \longmapsto \mathbf{r}_i$$

which gives rise to the same kernel matrix on the set S; that is

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \hat{\kappa}(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \hat{\phi}(\mathbf{x}_i), \hat{\phi}(\mathbf{x}_j) \right\rangle, \text{ for all } i, j = 1, \ldots, \ell.$$

This new projection maps data into the coordinate system determined by the orthonormal basis $\mathbf{q}_1, \ldots, \mathbf{q}_\ell$. Hence, to compute $\hat{\phi}$ and thus $\hat{\kappa}$ for new examples, we must evaluate the projections onto these basis vectors in the feature space. This can be done by effectively computing an additional column denoted by $\mathbf{r}$ of an extension of the matrix $\mathbf{R}$ from an additional column of $\mathbf{K}$ denoted by $\mathbf{k}$

$$\mathbf{r}_j = \nu_j^{-1} \left(\mathbf{k}_j - \sum_{t=1}^{j-1} \mathbf{R}_{tj} \mathbf{r}_t \right), \; j = 1, \ldots, \ell.$$

We started this section by asking how we might find a basis for the data when it lies in a subspace, or close to a subspace, of the feature space. If the data are not linearly independent the corresponding residual norm d_j will be equal to 0 when we come to process an example that lies in the subspace spanned by the earlier examples. This will occur if and only if the data lies in a subspace of dimension $j-1$, which is equivalent to saying that the rank of the matrix $\mathbf{X}$ is $j-1$. But this is equivalent to deriving

$$\mathbf{K} = \mathbf{R}'\mathbf{R}$$

with $\mathbf{R}$ a $(j-1) \times \ell$ matrix, or in other words to $\mathbf{K}$ having rank $j-1$. We have shown the following result.

Proposition 5.11 *The rank of the dataset S is equal to that of the kernel matrix $\mathbf{K}$ and by symmetry that of the matrix $\mathbf{X}'\mathbf{X}$.*

We can therefore compute the rank of the data in the feature space by computing the rank of the kernel matrix that only involves the inner products between the training points. Of course in high-dimensional feature spaces we may expect the rank to be equal to the number of data points. If we use the Gaussian kernel this will always be the case if the points are distinct.

Clearly the size of d_j indicates how independent the next example is from

the examples processed so far. If we wish to capture the most important dimensions of the data points it is therefore natural to vary the order that the examples are processed in the Cholesky decomposition by always choosing the point with largest residual norm, while those with small residuals are eventually ignored altogether. This leads to a reordering of the order in which the examples are processed. The reordering is computed by the statement

$$[\mathtt{a}, \mathtt{I}(\mathtt{j}+1)] \;=\; \mathtt{max}(\mathtt{d});$$

in the Matlab code below with the array `I` storing the permutation.

This approach corresponds to pivoting in Cholesky decomposition, while failing to include all the examples is referred to as an *incomplete Cholesky decomposition*. The corresponding approach in the feature space is known as *partial Gram–Schmidt orthonormalisation*.

Algorithm 5.12 [Cholesky decomposition or dual Gram–Schmidt] Matlab code for the incomplete Cholesky decomposition, equivalent to the dual partial Gram–Schmidt orthonormalisation is given in Code Fragment 5.4. ■

Notice that the index array I stores the indices of the vectors in the order in which they are chosen, while the parameter η allows for the possibility that the data is only approximately contained in a subspace. The residual norms will all be smaller than this value, while the dimension of the feature space obtained is given by T. If η is set small enough then T will be equal to the rank of the data in the feature space. Hence, we can determine the rank of the data in the feature space using Code Fragment 5.4.

The partial Gram–Schmidt procedure can be viewed as a method of reducing the size of the residuals by a greedy strategy of picking the largest at each iteration. This naturally raises the question of whether smaller residuals could result if the subspace was chosen globally to minimise the residuals. The solution to this problem will be given by choosing the eigensubspace that will be shown to minimise the sum-squared residuals. The next section begins to examine this approach to assessing the spread of the data in the feature space, though final answers to these questions will be given in Chapter 6.

5.3 Measuring the spread of the data

The mean estimates where the data is centred, while the variance measures the extent to which the data is spread. We can compare two zero-mean uni-

```
% original kernel matrix stored in variable K
% of size ell x ell.
% new features stored in matrix R of size T x ell
% eta gives threshold residual cutoff
j = 0;
R = zeros(ell,ell);
d = diag(K);
[a,I(j+1)] = max(d);
while a > eta
  j = j + 1;
  nu(j) = sqrt(a);
  for i = 1:ell
    R(j,i) = (K(I(j),i) - R(:,i)'*R(:,I(j)))/nu(j);
    d(i) = d(i) - R(j,i)^2;
  end
  [a,I(j+1)] = max(d);
end
T = j;
R = R(1:T,:);
% for new example with vector of inner products
% k of size ell x 1 to compute new features r
r = zeros(T, 1);
for j=1:T
  r(j) = (k(I(j)) - r'*R(:,I(j)))/nu(j);
end
```

Code Fragment 5.4. Matlab code for performing incomplete Cholesky decomposition or dual partial Gram–Schmidt orthogonalisation.

variate random variables using a measure known as the *covariance* defined to be the expectation of their product

$$\operatorname{cov}(x, y) = \mathbb{E}_{xy}[xy].$$

Frequently, raw feature components from different sensors are difficult to compare because the units of measurement are different. It is possible to compensate for this by standardising the features into unitless quantities. The standardisation $\hat{x}$ of a feature x is

$$\hat{x} = \frac{x - \mu_x}{\sigma_x},$$

where μ_x and σ_x are the mean and standard deviation of the random variable x. The measure $\hat{x}$ is known as the standard score. The covariance

$$\mathbb{E}_{\hat{x}\hat{y}}[\hat{x}\hat{y}]$$

of two such scores gives a measure of *correlation*

$$\rho_{xy} = \operatorname{corr}(x, y) = \mathbb{E}_{xy}\left[\frac{(x - \mu_x)(y - \mu_y)}{\sigma_x \sigma_y}\right]$$

between two random variables. A standardised score $\hat{x}$ has the property that $\mu_{\hat{x}} = 0$, $\sigma_{\hat{x}} = 1$. Hence, the correlation can be seen as the cosine of the angle between the standardised scores. The value ρ_{xy} is also known as the Pearson correlation coefficient. Note that for two random vectors x and y the following three conditions are equivalent:

$$\begin{aligned} \rho_{xy} &= 1; \\ \hat{x} &= \hat{y}; \\ y &= b + wx \text{ for some } b \text{ and for some } w > 0. \end{aligned}$$

Similarly $\rho_{xy} = -1$ if and only if $\hat{x} = -\hat{y}$ and the same holds with a negative w. This means that by comparing their standardised scores we can measure for linear correlations between two (univariate) random variables. In general we have

$$\rho_{xy} = \begin{cases} 0; & \text{if the two variables are linearly uncorrelated,} \\ \pm 1; & \text{if there is an exact linear relation between them.} \end{cases}$$

More generally

$$\left|\rho_{xy}\right| \approx 1 \text{ if and only if } y \approx b + wx,$$

and we talk about positive and negative linear correlations depending on the sign of ρ_{xy}. Hence, we can view $\left|\rho_{xy}\right|$ as an indicator for the presence of a pattern function of the form $g(x, y) = y - b - wx$.

The above observations suggest the following preprocessing might be helpful if we are seeking linear models.

Algorithm 5.13 [Standardising data] When building a linear model it is natural to standardise the features in order to make linear relations more apparent. Code Fragment 5.5 gives Matlab code to standardise input features by estimating the mean and standard deviation over a training set. ■

Variance of projections The above standardisation treats each coordinate independently. We will now consider measures that can take into account the interaction between different features. As discussed above if we are working with a kernel-induced feature space, we cannot access the coordinates of the points $\phi(S)$. Despite this we can learn about the spread in the

```
% original data stored in ell x N matrix X
% output uses the same variable X
% M is a row vector storing the column averages
% SD stores the column standard deviations
ell = size(X,1);
M = sum(X) / ell;
M2 = sum(X.^2)/ell;
SD = sqrt(M2 - M.^2);
X = (X - ones(ell,1)*M)./(ones(ell,1)*SD);
```

Code Fragment 5.5. Matlab code for standardising data.

feature space. Consider the $\ell \times N$ matrix $\mathbf{X}$ whose rows are the projections of the training points into the N-dimensional feature space

$$\mathbf{X} = \left[\begin{array}{cccc} \boldsymbol{\phi}(\mathbf{x}_1) & \boldsymbol{\phi}(\mathbf{x}_2) & \ldots & \boldsymbol{\phi}(\mathbf{x}_\ell) \end{array} \right]'.$$

Note that the feature vectors themselves are column vectors. If we assume that the data has zero mean or has already been centred then the covariance matrix $\mathbf{C}$ has entries

$$\mathbf{C}_{st} = \frac{1}{\ell} \sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i)_s \boldsymbol{\phi}(\mathbf{x}_i)_t, \; s,t = 1, \ldots, N.$$

Observe that

$$\ell \mathbf{C}_{st} = \sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i)_s \boldsymbol{\phi}(\mathbf{x}_i)_t = \left(\sum_{i=1}^{\ell} \boldsymbol{\phi}(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_i)' \right)_{st} = \left(\mathbf{X}'\mathbf{X} \right)_{st}.$$

If we consider a unit vector $\mathbf{v} \in \mathbb{R}^N$ then the expected value of the norm of the projection $\| P_{\mathbf{v}} \left(\boldsymbol{\phi}(\mathbf{x}) \right) \| = \mathbf{v}' \boldsymbol{\phi}(\mathbf{x}) / \left(\mathbf{v}'\mathbf{v} \right) = \mathbf{v}' \boldsymbol{\phi}(\mathbf{x})$ of the training points onto the space spanned by $\mathbf{v}$ is

$$\mu_{\mathbf{v}} = \hat{\mathbb{E}} \left[\| P_{\mathbf{v}} \left(\boldsymbol{\phi}(\mathbf{x}) \right) \| \right] = \hat{\mathbb{E}} \left[\mathbf{v}' \boldsymbol{\phi}(\mathbf{x}) \right] = \mathbf{v}' \hat{\mathbb{E}} \left[\boldsymbol{\phi}(\mathbf{x}) \right] = 0,$$

where we have again used the fact that the data is centred. Hence, if we wish to compute the variance of the norms of the projections onto $\mathbf{v}$ we have

$$\sigma_{\mathbf{v}}^2 = \hat{\mathbb{E}} \left[\left(\| P_{\mathbf{v}} \left(\boldsymbol{\phi}(\mathbf{x}) \right) \| - \mu_{\mathbf{v}} \right)^2 \right] = \hat{\mathbb{E}} \left[\| P_{\mathbf{v}} \left(\boldsymbol{\phi}(\mathbf{x}) \right) \|^2 \right] = \frac{1}{\ell} \sum_{i=1}^{\ell} \| P_{\mathbf{v}} \left(\boldsymbol{\phi}(\mathbf{x}_i) \right) \|^2$$

but we have

$$\begin{aligned} \frac{1}{\ell} \sum_{i=1}^{\ell} \| P_{\mathbf{v}} \left(\boldsymbol{\phi}(\mathbf{x}) \right) \|^2 &= \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{v}' \boldsymbol{\phi}(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_i)' \mathbf{v} = \hat{\mathbb{E}} \left[\mathbf{v}' \boldsymbol{\phi}(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_i)' \mathbf{v} \right] \quad (5.9) \\ &= \frac{1}{\ell} \mathbf{v}' \mathbf{X}' \mathbf{X} \mathbf{v}. \end{aligned}$$

So the covariance matrix contains the information needed to compute the variance of the data along any projection direction. If the data has not been centred we must subtract the square of the mean projection since the variance is given by

$$\begin{aligned}\sigma_{\mathbf{v}}^2 &= \hat{\mathbb{E}}\left[\left(\|P_{\mathbf{v}}\left(\boldsymbol{\phi}(\mathbf{x})\right)\| - \mu_{\mathbf{v}}\right)^2\right] = \hat{\mathbb{E}}\left[\|P_{\mathbf{v}}\left(\boldsymbol{\phi}(\mathbf{x})\right)\|^2\right] - \mu_{\mathbf{v}}^2 \\ &= \frac{1}{\ell}\mathbf{v}'\mathbf{X}'\mathbf{X}\mathbf{v} - \left(\frac{1}{\ell}\mathbf{v}'\mathbf{X}'\mathbf{j}\right)^2,\end{aligned}$$

where $\mathbf{j}$ is the all 1s vector.

Variance of projections in a feature space It is natural to ask if we can compute the variance of the projections onto a fixed direction $\mathbf{v}$ in the feature space using only inner product information. Clearly, we must choose the direction $\mathbf{v}$ so that we can express it as a linear combination of the projections of the training points

$$\mathbf{v} = \sum_{i=1}^{\ell} \alpha_i \boldsymbol{\phi}(\mathbf{x}_i) = \mathbf{X}'\boldsymbol{\alpha}.$$

For this $\mathbf{v}$ we can now compute the variance as

$$\begin{aligned}\sigma_{\mathbf{v}}^2 &= \frac{1}{\ell}\mathbf{v}'\mathbf{X}'\mathbf{X}\mathbf{v} - \left(\frac{1}{\ell}\mathbf{v}'\mathbf{X}'\mathbf{j}\right)^2 = \frac{1}{\ell}\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\mathbf{X}\mathbf{X}'\boldsymbol{\alpha} - \left(\frac{1}{\ell}\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\mathbf{j}\right)^2 \\ &= \frac{1}{\ell}\boldsymbol{\alpha}'\left(\mathbf{X}\mathbf{X}'\right)^2\boldsymbol{\alpha} - \frac{1}{\ell^2}\left(\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\mathbf{j}\right)^2 \\ &= \frac{1}{\ell}\boldsymbol{\alpha}'\mathbf{K}^2\boldsymbol{\alpha} - \frac{1}{\ell^2}\left(\boldsymbol{\alpha}'\mathbf{K}\mathbf{j}\right)^2,\end{aligned}$$

again computable from the kernel matrix.

Being able to compute the variance of projections in the feature space suggests implementing a classical method for choosing a linear classifier known as the Fisher discriminant. Using the techniques we have developed we will be able to implement this algorithm in the space defined by the kernel.

5.4 Fisher discriminant analysis I

The Fisher discriminant is a classification function

$$f(x) = \operatorname{sgn}\left(\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x})\rangle + b\right),$$

where the weight vector $\mathbf{w}$ is chosen to maximise the quotient

$$J(\mathbf{w}) = \frac{\left(\mu_{\mathbf{w}}^{+} - \mu_{\mathbf{w}}^{-}\right)^2}{\left(\sigma_{\mathbf{w}}^{+}\right)^2 + \left(\sigma_{\mathbf{w}}^{-}\right)^2}, \tag{5.10}$$

where $\mu_{\mathbf{w}}^{+}$ is the mean of the projection of the positive examples onto the direction $\mathbf{w}$, $\mu_{\mathbf{w}}^{-}$ the mean for the negative examples, and $\sigma_{\mathbf{w}}^{+}$, $\sigma_{\mathbf{w}}^{-}$ the corresponding standard deviations. Figure 5.2 illustrates the projection onto a particular direction $\mathbf{w}$ that gives good separation of the means with small variances of the positive and negative examples. The Fisher discriminant maximises the ratio between these quantities. The motivation for this choice

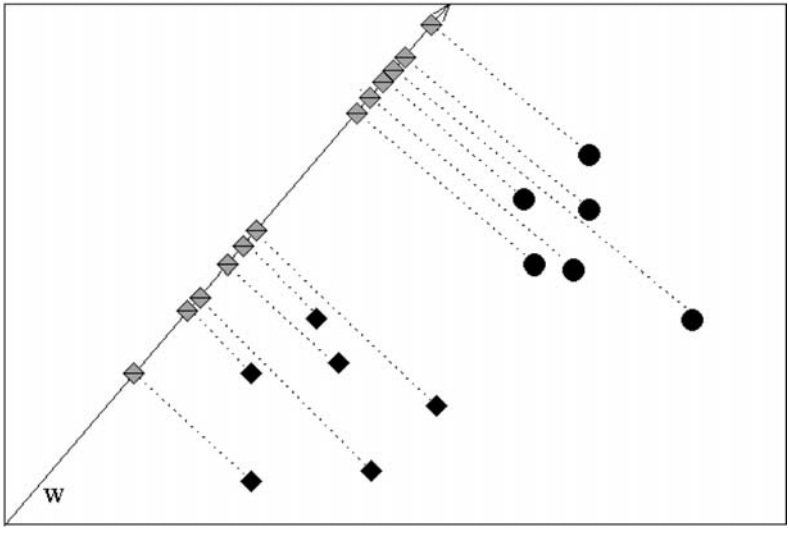

Fig. 5.2. The projection of points on to a direction $\mathbf{w}$ with positive and negative examples grouped separately.

is that the direction chosen maximises the separation of the means scaled according to the variances in that direction. Since we are dealing with kernel-defined feature spaces, it makes sense to introduce a regularisation on the norm of the weight vector $\mathbf{w}$ as motivated by Theorem 4.12. Hence, we consider the following optimisation.

Computation 5.14 [Regularised Fisher discriminant] The regularised Fisher discriminant chooses $\mathbf{w}$ to solve the following optimisation problem

$$\max_{\mathbf{w}} J(\mathbf{w}) = \frac{\left(\mu_{\mathbf{w}}^{+} - \mu_{\mathbf{w}}^{-}\right)^2}{\left(\sigma_{\mathbf{w}}^{+}\right)^2 + \left(\sigma_{\mathbf{w}}^{-}\right)^2 + \lambda \|\mathbf{w}\|^2} \tag{5.11}$$

■

First observe that the quotient is invariant under rescalings of the vector $\mathbf{w}$ so that we can constrain the denominator to have a fixed value C. Using a Lagrange multiplier ν we obtain the solution vector as

$$\begin{aligned}\mathbf{w}^\star &= \operatorname*{argmax}_{\mathbf{w}} \left(\left(\hat{\mathbb{E}} \left[y \mathbf{w}' \boldsymbol{\phi}(\mathbf{x}) \right] \right)^2 - \nu \left(\frac{1}{\ell^+} \mathbf{w}' \mathbf{X}' \mathbf{I}_+ \mathbf{I}_+ \mathbf{X} \mathbf{w} - \left(\frac{\mathbf{1}}{\ell^+} \mathbf{w}' \mathbf{X}' \mathbf{j}_+ \right)^{\mathbf{2}} \right.\right. \\ &\left.\left. + \frac{1}{\ell^-} \mathbf{w}' \mathbf{X}' \mathbf{I}_- \mathbf{I}_- \mathbf{X} \mathbf{w} - \left(\frac{\mathbf{1}}{\ell^-} \mathbf{w}' \mathbf{X}' \mathbf{j}_- \right)^{\mathbf{2}} + \lambda \mathbf{w}' \mathbf{w} - C \right) \right),\end{aligned}$$

where we have used a simplification of the numerator and the results of the previous section for the denominator. It is now a relatively straightforward derivation to obtain

$$\begin{aligned}\mathbf{w}^\star &= \operatorname*{argmax}_{\mathbf{w}} \left(\left(\frac{1}{\ell} \mathbf{y}' \mathbf{X} \mathbf{w} \right)^2 - \nu \left(\frac{1}{\ell^+} \mathbf{w}' \mathbf{X}' \mathbf{I}_+ \mathbf{X} \mathbf{w} - \left(\frac{\mathbf{1}}{\ell^+} \mathbf{w}' \mathbf{X}' \mathbf{j}_+ \right)^{\mathbf{2}} \right.\right. \\ &\left.\left. + \frac{1}{\ell^-} \mathbf{w}' \mathbf{X}' \mathbf{I}_- \mathbf{X} \mathbf{w} - \left(\frac{\mathbf{1}}{\ell^-} \mathbf{w}' \mathbf{X}' \mathbf{j}_- \right)^{\mathbf{2}} + \lambda \mathbf{w}' \mathbf{w} - C \right) \right) \\ &= \operatorname*{argmax}_{\mathbf{w}} \left(\left(\frac{1}{\ell} \mathbf{y}' \mathbf{X} \mathbf{w} \right)^2 - \nu \left(\lambda \mathbf{w}' \mathbf{w} - C \right.\right. \\ &\left.\left. + \frac{\ell}{2\ell^+\ell^-} \mathbf{w}' \mathbf{X}' \left(\frac{2\ell^-}{\ell} \mathbf{I}_+ + \frac{\mathbf{2}\ell^-}{\ell\ell^+} \mathbf{j}_+ \mathbf{j}_+' - \frac{2\ell^+}{\ell} \mathbf{I}_- + \frac{2\ell^+}{\ell\ell^-} \mathbf{j}_- \mathbf{j}_-' \right) \mathbf{X} \mathbf{w} \right) \right),\end{aligned}$$

where we have used $\mathbf{y}$ to denote the vector of $\{-1, +1\}$ labels, $\mathbf{I}_+$ (resp. $\mathbf{I}_-$) to indicate the identity matrix with 1s only in the columns corresponding to positive (resp. negative) examples and $\mathbf{j}_+$ (resp. $\mathbf{j}_-$) to denote the vector with 1s in the entries corresponding to positive (resp. negative) examples and otherwise 0s. Letting

$$\mathbf{B} = \mathbf{D} - \mathbf{C}^+ - \mathbf{C}^- \tag{5.12}$$

where $\mathbf{D}$ is a diagonal matrix with entries

$$\mathbf{D}_{ii} = \begin{cases} 2\ell^-/\ell & \text{if } y_i = +1 \\ 2\ell^+/\ell & \text{if } y_i = -1, \end{cases} \tag{5.13}$$

and $\mathbf{C}^+$ and $\mathbf{C}^-$ are given by

$$\mathbf{C}^+_{ij} = \begin{cases} 2\ell^- / (\ell\ell^+) & \text{if } y_i = +1 = y_j \\ 0 & \text{otherwise} \end{cases} \tag{5.14}$$

and

$$\mathbf{C}^-_{ij} = \begin{cases} 2\ell^+ / (\ell\ell^-) & \text{if } y_i = -1 = y_j \\ 0 & \text{otherwise,} \end{cases} \tag{5.15}$$

we can write

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmax}} \left(\left(\frac{1}{\ell} \mathbf{y}'\mathbf{X}\mathbf{w} \right)^2 - \nu \left(\lambda \mathbf{w}'\mathbf{w} - C + \frac{\ell}{2\ell^+\ell^-} \mathbf{w}'\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{w} \right) \right). \tag{5.16}$$

Varying C will only cause the solution to be rescaled since any rescaling of the weight vector will not affect the ratio of the numerator to the quantity constrained. If we now consider the optimisation

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmax}} \left(\mathbf{y}'\mathbf{X}\mathbf{w} - \nu' \left(\lambda \mathbf{w}'\mathbf{w} - C + \frac{\ell}{2\ell^+\ell^-} \mathbf{w}'\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{w} \right) \right), \tag{5.17}$$

it is clear that the solutions of problems (5.16) and (5.17) will be identical up to reversing the direction of $\mathbf{w}^\star$, since once the denominator is constrained to have value C the weight vector $\mathbf{w}$ that maximises (5.17) will maximise (5.16). This holds since the maxima of $\mathbf{y}'\mathbf{X}\mathbf{w}$ and $(\mathbf{y}'\mathbf{X}\mathbf{w})^2$ coincide with a possible change of sign of $\mathbf{w}$. Hence, with an appropriate re-definition of ν, λ and C

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmax}} \left(\mathbf{y}'\mathbf{X}\mathbf{w} - \frac{\nu}{2} \mathbf{w}'\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{w} + C - \frac{\lambda\nu}{2} \mathbf{w}'\mathbf{w} \right).$$

Taking derivatives with respect to $\mathbf{w}$ we obtain

$$\begin{aligned} \mathbf{0} &= \mathbf{X}'\mathbf{y} - \nu\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{w} - \lambda\nu\mathbf{w}, \\ \text{so that } \lambda\nu\mathbf{w} &= \mathbf{X}' \left(\mathbf{y} - \nu\mathbf{B}\mathbf{X}\mathbf{w} \right), \end{aligned}$$

Dual expression This implies that we can express $\mathbf{w}$ in the dual form as a linear combination of the training examples $\mathbf{w} = \mathbf{X}'\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is given by

$$\boldsymbol{\alpha} = \frac{1}{\lambda\nu} \left(\mathbf{y} - \nu\mathbf{B}\mathbf{X}\mathbf{w} \right). \tag{5.18}$$

Substituting for $\mathbf{w}$ in equation (5.18) we obtain

$$\lambda\nu\boldsymbol{\alpha} = \mathbf{y} - \nu\mathbf{B}\mathbf{X}\mathbf{X}'\boldsymbol{\alpha} = \mathbf{y} - \nu\mathbf{B}\mathbf{K}\boldsymbol{\alpha}.$$

giving

$$\left(\nu\mathbf{B}\mathbf{K} + \lambda\nu\mathbf{I} \right) \boldsymbol{\alpha} = \mathbf{y}.$$

Since the classification function is invariant to rescalings of the weight vector, we can rescale $\boldsymbol{\alpha}$ by ν to obtain

$$\left(\mathbf{B}\mathbf{K} + \lambda\mathbf{I} \right) \boldsymbol{\alpha} = \mathbf{y}.$$

Notice the similarity with the ridge regression solution considered in Chapter 2, but here the real-valued outputs are replaced by the binary labels and

the additional matrix $\mathbf{B}$ is included, though for balanced datasets this will be close to $\mathbf{I}$. In general the solution is given by

$$\boldsymbol{\alpha} = (\mathbf{BK} + \lambda \mathbf{I})^{-1} \mathbf{y},$$

so that the corresponding classification function is

$$h(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{\ell} \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i) - b\right) = \text{sgn}\left(\mathbf{k}' (\mathbf{BK} + \lambda \mathbf{I})^{-1} \mathbf{y} - b\right), \quad (5.19)$$

where $\mathbf{k}$ is the vector with entries $\kappa(\mathbf{x}, \mathbf{x}_i)$, $i = 1, \ldots, \ell$ and b is an appropriate offset. The value of b is chosen so that $\mathbf{w}'\boldsymbol{\mu}^+ - b = b - \mathbf{w}'\boldsymbol{\mu}^-$, that is so that the decision boundary bisects the line joining the two centres of mass. Taking the weight vector $\mathbf{w} = \mathbf{X}'\boldsymbol{\alpha}$, we have

$$b = 0.5\boldsymbol{\alpha}'\mathbf{X}\left(\frac{1}{\ell^+}\mathbf{X}'\mathbf{j}_+ + \frac{1}{\ell^-}\mathbf{X}'\mathbf{j}_-\right) = 0.5\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\mathbf{t} = 0.5\boldsymbol{\alpha}'\mathbf{K}\mathbf{t}, \quad (5.20)$$

where $\mathbf{t}$ is the vector with entries

$$t_i = \begin{cases} 1/\ell^+ & \text{if } y_i = +1 \\ 1/\ell^- & \text{if } y_i = -1. \end{cases} \quad (5.21)$$

We summarise in the following computation.

Computation 5.15 [Regularised kernel Fisher discriminant] The regularised kernel Fisher discriminant chooses the dual variables $\boldsymbol{\alpha}$ as follows

$$\boldsymbol{\alpha} = (\mathbf{BK} + \lambda \mathbf{I})^{-1} \mathbf{y},$$

where $\mathbf{K}$ is the kernel matrix, $\mathbf{B}$ is given by (5.12)-(5.15), and the resulting classification function is given by (5.19) and the threshold b by (5.20) and (5.21). ■

Finally, we give a more explicit description of the dual algorithm.

Algorithm 5.16 [Dual Fisher discriminant] Matlab code for the dual Fisher discriminant algorithm is given in Code Fragment 5.6. ■

Proposition 5.17 *Consider the classification training set*

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

with a feature space implicitly defined by the kernel $\kappa(\mathbf{x}, \mathbf{z})$. *Let*

$$f(\mathbf{x}) = \mathbf{y}'(\mathbf{BK} + \lambda \mathbf{I})^{-1}\mathbf{k} - b,$$

```
% K is the kernel matrix of ell training points
% lambda the regularisation parameter
% y the labels
% The inner products between the training and t test points
% are stored in the matrix Ktest of dimension ell x t
% the true test labels are stored in ytruetest
ell = size(K,1);
ellplus = (sum(y) + ell)/2;
yplus = 0.5*(y + 1);
ellminus = ell - ellplus;
yminus = yplus - y;
t = size(Ktest,2);
rescale = ones(ell,1)+y*((ellminus-ellplus)/ell);
plusfactor = 2*ellminus/(ell*ellplus);
minusfactor = 2*ellplus/(ell*ellminus);
B = diag(rescale) - (plusfactor * yplus) * yplus'
    - (minusfactor * yminus) * yminus';
alpha = (B*K + lambda*eye(ell,ell))\y;
b = 0.25*(alpha'*K*rescale)/(ellplus*ellminus);
ytest = sign(Ktest'*alpha - b);
error = sum(abs(ytruetest - ytest))/(2*t)
```

Code Fragment 5.6. Kernel Fisher discriminant algorithm

where $\mathbf{K}$ *is the* $\ell \times \ell$ *matrix with entries* $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}$ *is the vector with entries* $\mathbf{k}_i = \kappa(\mathbf{x}_i, \mathbf{x})$, $\mathbf{B}$ *is defined by equations (5.12)–(5.15) and* b *is defined by equations (5.20)–(5.21). Then the function* $f(\mathbf{x})$ *is equivalent to the hyperplane in the feature space implicitly defined by the kernel* $\kappa(\mathbf{x}, \mathbf{z})$ *that solves the Fisher discriminant problem (5.10) regularised by the parameter* λ.

Remark 5.18 [Statistical properties] In this example of the kernel Fisher discriminant we did not obtain an explicit performance guarantee. If we observe that the function obtained has a non-zero margin γ we could apply Theorem 4.17 but this in itself does not motivate the particular choice of optimisation criterion. Theorem 4.12 as indicated above can motivate the regularisation of the norm of the weight vector, but a direct optimisation of the bound will lead to the more advanced algorithms considered in Chapter 7. ■

5.5 Summary

- Many properties of the data in the embedding space can be calculated using only information obtained through kernel evaluations. These include

distances between points, distances of points from the centre of mass, dimensionality of the subspace spanned by the data, and so on.

- Many transformations of the data in the embedding space can be realised through operations on the kernel matrix. For example, translating a dataset so that its centre of mass coincides with the origin corresponds to a set of operations on the kernel matrix; normalisation of the data produces a mapping to vectors of norm 1, and so on.
- Certain transformations of the kernel matrix correspond to performing projections in the kernel-defined feature space. Deflation corresponds to one such projection onto the orthogonal complement of a 1-dimensional subspace. Using these insights it is shown that incomplete Cholesky decomposition of the kernel matrix is a dual implementation of partial Gram–Schmidt orthonormalisation in the feature space.
- Three simple pattern analysis algorithms, one for novelty-detection and the other two for classification, have been described using the basic geometric relations derived in this chapter.
- The Fisher discriminant can be viewed as optimising a measure of the separation of the projections of the data onto a 1-dimensional subspace.

5.6 Further reading and advanced topics

In this chapter we have shown how to evaluate a number of properties of a set of points in a kernel defined feature space, typically the image of a generic dataset through the embedding map $\boldsymbol{\phi}$. This discussion is important both as a demonstration of techniques and methods that will be used in the following three chapters, and because the properties discussed can be directly used to analyse data, albeit in simple ways. In this sense, they are some of the first pattern analysis algorithms we have presented.

It is perhaps surprising how much information about a dataset can be obtained simply from its kernel matrix. The idea of using Mercer kernels as inner products in an embedding space in order to implement a learning algorithm dates back to Aizermann, Braverman and Rozonoer [1], who considered a dual implementation of the perceptron algorithm. However, its introduction to mainstream machine learning literature had to wait until 1992 with the first paper on support vector machines [16]. For some time after that paper, kernels were only used in combination with the maximal margin algorithm, while the idea that other types of algorithms could be implemented in this way began to emerge. The possibility of using kernels in any algorithm that can be formulated in terms of inner products was first mentioned in the context of kernel PCA (discussed in Chapter 6) [121], [20].

The centre of mass, the distance, the expected squared distance from the centre are all straight-forward applications of the kernel concept, and appear to have been introduced independently by several authors since the early days of research in this field. The connection between Parzen windows and the centres of mass of the two classes was pointed out by Schölkopf and is discussed in the book [120]. Also the normalisation procedure is well-known, while the centering procedure was first published in the paper [121]. Kernel Gram–Schmidt was introduced in [31] and can also be seen as an approximation of kernel PCA. The equivalent method of incomplete Cholesky decomposition was presented by [7]. See [49] for a discussion of QR decomposition.

Note that in Chapter 6 many of these ideas will be re-examined, including the kernel Fisher discriminant and kernel PCA, so more references can be found in Section 6.9.

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