6

Pattern analysis using eigen-decompositions

The previous chapter saw the development of some basic tools for working in a kernel-defined feature space resulting in some useful algorithms and techniques. The current chapter will extend the methods in order to understand the spread of the data in the feature space. This will be followed by examining the problem of identifying correlations between input vectors and target values. Finally, we discuss the task of identifying covariances between two different representations of the same object.

All of these important problems in kernel-based pattern analysis can be reduced to performing an eigen- or generalised eigen-analysis, that is the problem of finding solutions of the equation $\mathbf{Aw} = \lambda\mathbf{Bw}$ given symmetric matrices $\mathbf{A}$ and $\mathbf{B}$. These problems range from finding a set of k directions in the embedding space containing the maximum amount of variance in the data (principal components analysis (PCA)), through finding correlations between input and output representations (partial least squares (PLS)), to finding correlations between two different representations of the same data (canonical correlation analysis (CCA)). Also the Fisher discriminant analysis from Chapter 5 can be cast as a generalised eigenvalue problem.

The importance of this class of algorithms is that the generalised eigenvectors problem provides an efficient way of optimising an important family of cost functions; it can be studied with simple linear algebra and can be solved or approximated efficiently using a number of well-known techniques from computational algebra. Furthermore, we show that the problems can be solved in a kernel-defined feature space using a dual representation, that is, they only require information about inner products between datapoints.

6.1 Singular value decomposition

We have seen how we can sometimes learn something about the covariance matrix $\mathbf{C}$ by using the kernel matrix $\mathbf{K} = \mathbf{X}\mathbf{X}'$. For example in the previous chapter the variances were seen to be given by the covariance matrix, but could equally be evaluated using the kernel matrix. The close connection between these two matrices will become more apparent if we consider the eigen-decomposition of both matrices

$$\ell\mathbf{C} = \mathbf{X}'\mathbf{X} = \mathbf{U}\tilde{\mathbf{\Lambda}}_N\mathbf{U}' \text{ and } \mathbf{K} = \mathbf{X}\mathbf{X}' = \mathbf{V}\mathbf{\Lambda}_\ell\mathbf{V}',$$

where the columns $\mathbf{u}_i$ of the orthonormal matrix $\mathbf{U}$ are the eigenvectors of $\ell\mathbf{C}$, and the columns $\mathbf{v}_i$ of the orthonormal matrix $\mathbf{V}$ are the eigenvectors of $\mathbf{K}$. Now consider an eigenvector–eigenvalue pair $\mathbf{v}$, λ of $\mathbf{K}$. We have

$$\ell\mathbf{C}(\mathbf{X}'\mathbf{v}) = \mathbf{X}'\mathbf{X}\mathbf{X}'\mathbf{v} = \mathbf{X}'\mathbf{K}\mathbf{v} = \lambda\mathbf{X}'\mathbf{v},$$

implying that $\mathbf{X}'\mathbf{v}$, λ is an eigenvector–eigenvalue pair for $\ell\mathbf{C}$. Furthermore, the norm of $\mathbf{X}'\mathbf{v}$ is given by

$$\left\|\mathbf{X}'\mathbf{v}\right\|^2 = \mathbf{v}'\mathbf{X}\mathbf{X}'\mathbf{v} = \lambda,$$

so that the corresponding normalised eigenvector of $\ell\mathbf{C}$ is $\mathbf{u} = \lambda^{-1/2}\mathbf{X}'\mathbf{v}$. There is a symmetry here since we also have that

$$\lambda^{-1/2}\mathbf{X}\mathbf{u} = \lambda^{-1}\mathbf{X}\mathbf{X}'\mathbf{v} = \mathbf{v}.$$

We can summarise these relations as follows

$$\mathbf{u} = \lambda^{-1/2}\mathbf{X}'\mathbf{v} \quad \text{and} \quad \mathbf{v} = \lambda^{-1/2}\mathbf{X}\mathbf{u}.$$

We can deflate both $\ell\mathbf{C}$ and $\mathbf{K}$ of the corresponding eigenvalues by making the following deflation of $\mathbf{X}$:

$$\mathbf{X} \longmapsto \tilde{\mathbf{X}} = \mathbf{X} - \mathbf{v}\mathbf{v}'\mathbf{X} = \mathbf{X} - \lambda^{1/2}\mathbf{v}\mathbf{u}' = \mathbf{X} - \mathbf{X}\mathbf{u}\mathbf{u}'. \tag{6.1}$$

This follows from the equalities

$$\tilde{\mathbf{X}}\tilde{\mathbf{X}}' = \left(\mathbf{X} - \mathbf{v}\mathbf{v}'\mathbf{X}\right)\left(\mathbf{X} - \mathbf{v}\mathbf{v}'\mathbf{X}\right)' = \mathbf{X}\mathbf{X}' - \lambda\mathbf{v}\mathbf{v}',$$

and

$$\tilde{\mathbf{X}}'\tilde{\mathbf{X}} = \left(\mathbf{X} - \mathbf{v}\mathbf{v}'\mathbf{X}\right)'\left(\mathbf{X} - \mathbf{v}\mathbf{v}'\mathbf{X}\right) = \mathbf{X}'\mathbf{X} - \mathbf{X}'\mathbf{v}\mathbf{v}'\mathbf{X} = \mathbf{X}'\mathbf{X} - \lambda\mathbf{u}\mathbf{u}'.$$

Hence, the first $t = \text{rank}(\mathbf{X}\mathbf{X}') \leq \min(N, \ell)$ columns $\mathbf{U}_t$ of $\mathbf{U}$ can be chosen as

$$\mathbf{U}_t = \mathbf{X}'\mathbf{V}_t\mathbf{\Lambda}_t^{-1/2}, \tag{6.2}$$

where we assume the t non-zero eigenvalues of $\mathbf{K}$ and $\ell\mathbf{C}$ appear in descending order. But by the symmetry of $\ell\mathbf{C}$ and $\mathbf{K}$ these are the only non-zero eigenvalues of $\ell\mathbf{C}$, since we can transform any eigenvector–eigenvalue pair $\mathbf{u}$, λ of $\ell\mathbf{C}$ to an eigenvector–eigenvalue pair $\mathbf{Xu}$, λ of $\mathbf{K}$. It follows, as we have already seen, that

$$t = \text{rank}(\mathbf{XX}') = \text{rank}(\mathbf{X}'\mathbf{X}).$$

By extending $\mathbf{U}_t$ to $\mathbf{U}$ and $\mathbf{\Lambda}_t^{1/2}$ to an $N \times \ell$ matrix whose additional entries are all zero, we obtain the *singular value decomposition (SVD)* of the matrix $\mathbf{X}'$ defined as a decomposition

$$\mathbf{X}' = \mathbf{U}\mathbf{\Sigma}\mathbf{V}',$$

where $\mathbf{\Sigma}$ is an $N \times \ell$ matrix with all entries 0 except the leading diagonal which has entries $\sigma_i = \sqrt{\lambda_i}$ satisfying $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_t > 0$ for $t = \text{rank}(\mathbf{X}) \leq \min(N, \ell)$ with $\mathbf{U}$ and $\mathbf{V}$ square matrices satisfying

$$\mathbf{V}'\mathbf{V} = \mathbf{I} \text{ so that } \mathbf{V}' = \mathbf{V}^{-1} \text{ and similarly } \mathbf{U}' = \mathbf{U}^{-1},$$

also known as *orthogonal* matrices.

Consequences of singular value decomposition There are a number of interesting consequences. Notice how equation (6.2) implies a dual representation for the jth eigenvector $\mathbf{u}_j$ of $\ell\mathbf{C}$ with the coefficients given by the corresponding eigenvector $\mathbf{v}_j$ of $\mathbf{K}$ scaled by $\lambda_j^{-1/2}$, that is

$$\mathbf{u}_j = \lambda_j^{-1/2} \sum_{i=1}^{\ell} (\mathbf{v}_j)_i \, \boldsymbol{\phi}(\mathbf{x}_i) = \sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \boldsymbol{\phi}(\mathbf{x}_i), \; j = 1, \ldots, t,$$

where the dual variables $\boldsymbol{\alpha}^j$ for the jth vector $\mathbf{u}_j$ are given by

$$\boldsymbol{\alpha}^j = \lambda_j^{-1/2} \mathbf{v}_j. \tag{6.3}$$

and $\mathbf{v}_j$, λ_j are the jth eigenvector–eigenvalue pair of the kernel matrix.

It is important to remark that if we wish to compute the projection of a new data point $\boldsymbol{\phi}(\mathbf{x})$ onto the direction $\mathbf{u}_j$ in the feature space, this is given by

$$\begin{aligned} P_{\mathbf{u}_j}(\boldsymbol{\phi}(\mathbf{x})) &= \mathbf{u}_j' \boldsymbol{\phi}(\mathbf{x}) = \left\langle \sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \right\rangle = \sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}) \rangle \\ &= \sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \kappa(\mathbf{x}_i, \mathbf{x}), \end{aligned} \tag{6.4}$$

Hence we will be able to project new data onto the eigenvectors in the feature space by performing an eigen-decomposition of the kernel matrix. We will present the details of this algorithm in Section 6.2.1 after introducing primal principal components analysis in the next section.

Remark 6.1 [Centering not needed] Although the definition of the covariance matrix assumes the data to be centred, none of the derivations given in this section make use of this fact. Hence, we need not assume that the covariance matrix is computed for centred data to obtain dual representations of the projections. ■

Remark 6.2 [Notation conventions] We have used the notation $\mathbf{u}_j$ for the primal eigenvectors in contrast to our usual $\mathbf{w}_j$. This is to maintain consistency with the standard notation for the singular value decomposition of a matrix. Note that we have used the standard notation for the dual variables. ■

6.2 Principal components analysis

In the previous chapter we saw how the variance in any fixed direction in the feature space could be measured using only the kernel matrix. This made it possible to find the Fisher discriminant function in a kernel-defined feature space by appropriate manipulation of the kernel matrix. We now consider finding a direction that maximises the variance in the feature space.

Maximising variance If we assume that the data has been centred in the feature space using for example Code Fragment 5.2, then we can compute the variance of the projection onto a normalised direction $\mathbf{w}$ as

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

where we again use $\hat{\mathbb{E}}[f(\mathbf{x})]$ to denote the empirical mean of $f(\mathbf{x})$

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{\ell}\sum_{i=1}^{\ell} f(\mathbf{x}_i),$$

and $\mathbf{C} = \frac{1}{\ell}\mathbf{X}'\mathbf{X}$ is the covariance matrix of the data sample. Hence, finding the directions of maximal variance reduces to the following computation.

Computation 6.3 [Maximising variance] The direction that maximises the variance can be found by solving the following problem

$$\begin{array}{ll} \max_{\mathbf{w}} & \mathbf{w}'\mathbf{C}\mathbf{w}, \\ \text{subject to} & \|\mathbf{w}\|_2 = 1. \end{array} \tag{6.5}$$

■

Eigenvectors for maximising variance Consider the quotient

$$\rho(\mathbf{w}) = \frac{\mathbf{w}'C\mathbf{w}}{\mathbf{w}'\mathbf{w}}.$$

Since rescaling $\mathbf{w}$ has a quadratic effect on $\rho(\mathbf{w})$, the solution of (6.5) is the direction that maximises $\rho(\mathbf{w})$. Observe that this is the optimisation of the Raleigh quotient given in (3.2) , where it was observed that the solution is given by the eigenvector corresponding to the largest eigenvalue with the value of $\rho(\mathbf{w})$ given by the eigenvalue. We can search for the direction of second largest variance in the orthogonal subspace, by looking for the largest eigenvector in the matrix obtained by deflating the matrix $\mathbf{C}$ with respect to $\mathbf{w}$. This gives the eigenvector of $\mathbf{C}$ corresponding to the second-largest eigenvalue. Repeating this step shows that the mutually orthogonal directions of maximum variance in order of decreasing size are given by the eigenvectors of $\mathbf{C}$.

Remark 6.4 [Explaining variance] We have seen that the size of the eigenvalue is equal to the variance in the chosen direction. Hence, if we project into a number of orthogonal directions the total variance is equal to the sum of the corresponding eigenvalues, making it possible to say what percentage of the overall variance has been captured, where the overall variance is given by the sum of all the eigenvalues, which equals the trace of the kernel matrix or the sum of the squared norms of the data. ■

Since rescaling a matrix does not alter the eigenvectors, but simply rescales the corresponding eigenvalues, we can equally search for the directions of maximum variance by analysing the matrix $\ell\mathbf{C} = \mathbf{X}'\mathbf{X}$. Hence, the first eigenvalue of the matrix $\ell\mathbf{C}$ equals the sum of the squares of the projections of the data into the first eigenvector in the feature space. A similar conclusion can be reached using the Courant–Fisher Theorem 3.6 applied to the first eigenvalue λ_1. By the above observations and equation (5.9) we have

$$\lambda_1(\ell\mathbf{C}) \;=\; \lambda_1(\mathbf{X}'\mathbf{X}) = \max_{\dim(T)=1} \min_{0\neq\mathbf{u}\in T} \frac{\mathbf{u}'\mathbf{X}'\mathbf{X}\mathbf{u}}{\mathbf{u}'\mathbf{u}}$$

$$
\begin{aligned}
&= \max_{0\neq\mathbf{u}} \frac{\mathbf{u}'\mathbf{X}'\mathbf{X}\mathbf{u}}{\mathbf{u}'\mathbf{u}} = \max_{0\neq\mathbf{u}} \frac{\|\mathbf{X}\mathbf{u}\|^2}{\|\mathbf{u}\|^2} = \max_{0\neq\mathbf{u}} \sum_{i=1}^{\ell} P_{\mathbf{u}}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)^2 \\
&= \sum_{i=1}^{\ell} \|\boldsymbol{\phi}(\mathbf{x}_i)\|^2 - \min_{0\neq\mathbf{u}} \sum_{i=1}^{\ell} \left\| P_{\mathbf{u}}^{\perp}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)\right\|^2,
\end{aligned}
$$

where $P_{\mathbf{u}}^{\perp}\left(\boldsymbol{\phi}(\mathbf{x})\right)$ is the projection of $\phi(\mathbf{x})$ into the space orthogonal to $\mathbf{u}$. The last equality follows from Pythagoras's theorem since the vectors are the sum of two orthogonal projections. Furthermore, the unit vector that realises the max and min is the first column $\mathbf{u}_1$ of the matrix $\mathbf{U}$ of the eigen-decomposition

$$\mathbf{X}'\mathbf{X} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}'$$

of $\mathbf{X}'\mathbf{X}$.

A similar application of the Courant–Fisher Theorem 3.6 to the ith eigenvalue of the matrix $\ell\mathbf{C}$ gives

$$
\begin{aligned}
\lambda_i(\ell\mathbf{C}) &= \lambda_i(\mathbf{X}'\mathbf{X}) = \max_{\dim(T)=i} \min_{0\neq\mathbf{u}\in T} \frac{\mathbf{u}'\mathbf{X}'\mathbf{X}\mathbf{u}}{\mathbf{u}'\mathbf{u}} \\
&= \max_{\dim(T)=i} \min_{0\neq\mathbf{u}\in T} \sum_{j=1}^{\ell} P_{\mathbf{u}}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)^2 = \sum_{j=1}^{\ell} P_{\mathbf{u}_i}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)^2,
\end{aligned}
$$

that is, the sum of the squares of the projections of the data in the direction of the ith eigenvector $\mathbf{u}_i$ in the feature space. If we consider projecting into the space U_k spanned by the first k eigenvectors, we have

$$\sum_{i=1}^{k} \lambda_i = \sum_{i=1}^{k}\sum_{j=1}^{\ell} P_{\mathbf{u}_i}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)^2 = \sum_{j=1}^{\ell}\sum_{i=1}^{k} P_{\mathbf{u}_i}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)^2 = \sum_{j=1}^{\ell} \|P_{U_k}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)\|^2,$$

where we have used $P_{U_k}\left(\boldsymbol{\phi}(\mathbf{x})\right)$ to denote the orthogonal projection of $\phi(\mathbf{x})$ into the subspace U_k. Furthermore, notice that if we consider $k = N$ the projection becomes the identity and we have

$$\sum_{i=1}^{N} \lambda_i = \sum_{j=1}^{\ell} \|P_{U_N}\left(\boldsymbol{\phi}(\mathbf{x}_i)\right)\|^2 = \sum_{j=1}^{\ell} \|\boldsymbol{\phi}(\mathbf{x}_i)\|^2, \tag{6.6}$$

something that also follows from the fact that the expressions are the traces of two similar matrices $\ell\mathbf{C}$ and $\boldsymbol{\Lambda}$.

Definition 6.5 [Principal components analysis] Principal components analysis (PCA) takes an initial subset of the principal axes of the training data and projects the data (both training and test) into the space spanned by

this set of eigenvectors. We effectively preprocess a set of data by projecting it into the subspace spanned by the first k eigenvectors of the covariance matrix of the training set for some $k < \ell$. The new coordinates are known as the *principal coordinates* with the eigenvectors referred to as the *principal axes*. ■

Algorithm 6.6 [Primal principal components analysis] The primal principal components analysis algorithm performs the following computation:

input	Data $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\} \subset \mathbb{R}^n$, dimension k.
process	$\boldsymbol{\mu} = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathbf{x}_i$ $\mathbf{C} = \frac{1}{\ell}\sum_{i=1}^{\ell} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})'$ $[\mathbf{U}, \boldsymbol{\Lambda}] = \text{eig}(\ell\mathbf{C})$ $\tilde{\mathbf{x}}_i = \mathbf{U}_k'\mathbf{x}_\ell$, $i = 1, \ldots, \ell$.
output	Transformed data $\tilde{S} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_\ell\}$.

■

Remark 6.7 [Data lying in a subspace] Suppose that we have a data matrix in which one column is exactly constant for all examples. Clearly, this feature carries no information and will be set to zero by the centering operation. Hence, we can remove it by projecting onto the other dimensions without losing any information about the data. Data may in fact lie in a lower-dimensional subspace even if no individual feature is constant. This corresponds to the subspace not being aligned with any of the axes. The principal components analysis is nonetheless able to detect such a subspace. For example if the data has rank r then only the first r eigenvalues are non-zero and so the corresponding eigenvectors span the subspace containing the data. Therefore, projection into the first r principal axes exactly captures the training data. ■

Remark 6.8 [Denoising] More generally if the eigenvalues beyond the kth are small we can think of the data as being approximately k-dimensional, the features beyond the kth being approximately constant the data has little variance in these directions. In such cases it can make sense to project the data into the space spanned by the first k eigenvectors. It is possible that the variance in the dimensions we have removed is actually the result of noise, so that their removal can in some cases improve the representation of the data. Hence, performing principal components analysis can be regarded as an example of *denoising*. ■

Remark 6.9 [Applications to document analysis] We will also see in Chapter 10 how principal components analysis has a semantic focussing effect when applied in document analysis, with the eigenvectors representing concepts or themes inferred from the statistics of the data. The representation of an input in the principal coordinates can then be seen as an indication of how much it is related to these different themes. ■

Remark 6.10 [PCA for visualisation] In Chapter 8 we will also see how a low-dimensional PCA projection can be used as a visualisation tool. In the case of non-numeric datasets this is particularly powerful since the data itself does not have a natural geometric structure, but only a high-dimensional implicit representation implied by the choice of kernel. Hence, in this case kernel PCA can be seen as a way of inferring a low-dimensional explicit geometric feature space that best captures the structure of the data. ■

PCA explaining variance The eigenvectors of the covariance matrix ordered by decreasing eigenvalue correspond to directions of decreasing variance in the data, with the eigenvalue giving the amount of variance captured by its eigenvector. The larger the dimension k of the subspace U_k the greater percentage of the variance that is captured. These approximation properties are explored further in the alternative characterisation given below. We can view identification of a low-dimensional subspace capturing a high proportion of the variance as a pattern identified in the training data. This of course raises the question of whether the pattern is stable, that is, if the subspace we have identified will also capture the variance of new data arising from the same distribution. We will examine this statistical question once we have introduced a dual version of the algorithm.

Remark 6.11 [Centering not needed] The above derivation does not make use of the fact that the data is centred. It therefore follows that if we define

$$\mathbf{C} = \frac{1}{\ell}\mathbf{X}'\mathbf{X}$$

with $\mathbf{X}$ not centred, the same derivation holds as does the proposition given below. Centering the data has the advantage of reducing the overall sum of the eigenvalues, hence removing irrelevant variance arising from a shift of the centre of mass, but we can use principal components analysis on uncentred data. ■

Alternative characterisations An alternative characterisation of the principal components (or principal axes) of a dataset will be important for the

analysis of kernel PCA in later sections. We first introduce some additional notation. We have used $P_U(\boldsymbol{\phi}(\mathbf{x}))$ to denote the orthogonal projection of an embedded point $\boldsymbol{\phi}(\mathbf{x})$ into the subspace U. We have seen above that we are also interested in the error resulting from using the projection rather than the actual vector $\boldsymbol{\phi}(\mathbf{x})$. This difference

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

is the projection into the orthogonal subspace and will be referred to as the *residual*. We can compute its norm from the norms of $\boldsymbol{\phi}(\mathbf{x})$ and $P_U(\boldsymbol{\phi}(\mathbf{x}))$ using Pythagoras's theorem. We will typically assess the quality of a projection by the average of the squared norms of the residuals of the training data

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\left\|P_U^{\perp}(\boldsymbol{\phi}(\mathbf{x}_i))\right\|^2 = \frac{1}{\ell}\|\boldsymbol{\xi}\|^2, \text{ where } \xi_i = \left\|P_U^{\perp}(\boldsymbol{\phi}(\mathbf{x}_i))\right\|.$$

The next proposition shows that using the space spanned by the first k principal components of the covariance matrix minimises this quantity.

Proposition 6.12 *Given a training set S with covariance matrix $\mathbf{C}$, the orthogonal projection $P_{U_k}(\boldsymbol{\phi}(\mathbf{x}))$ into the subspace U_k spanned by the first k eigenvectors of $\mathbf{C}$ is the k-dimensional orthogonal projection minimising the average squared distance between each training point and its image, in other words U_k solves the optimisation problem*

$$\begin{array}{ll} \min_U & J^{\perp}(U) = \sum_{i=1}^{\ell}\left\|P_U^{\perp}(\boldsymbol{\phi}(\mathbf{x}_i))\right\|_2^2 \\ \textit{subject to} & \dim U = k. \end{array} \tag{6.7}$$

Furthermore, the value of $J^{\perp}(U)$ at the optimum is given by

$$J^{\perp}(U) = \sum_{i=k+1}^{N}\lambda_i, \tag{6.8}$$

where $\lambda_1, \ldots, \lambda_N$ are the eigenvalues of the matrix $\ell\mathbf{C}$ in decreasing order.

Proof A demonstration of this fact will also illuminate various features of the principal coordinates. Since, $P_U(\phi(\mathbf{x}_i))$ is an orthogonal projection it follows from Pythagoras's theorem that

$$J^{\perp}(U) \;=\; \sum_{i=1}^{\ell}\left\|P_U^{\perp}(\boldsymbol{\phi}(\mathbf{x}_i))\right\|_2^2 = \sum_{i=1}^{\ell}\|\boldsymbol{\phi}(\mathbf{x}_i) - P_U(\boldsymbol{\phi}(\mathbf{x}_i))\|_2^2$$

$$= \sum_{i=1}^{\ell} \|\boldsymbol{\phi}(\mathbf{x}_i)\|^2 - \sum_{i=1}^{\ell} \|P_U(\boldsymbol{\phi}(\mathbf{x}_i))\|_2^2 . \tag{6.9}$$

Hence, the optimisation (6.7) has the same solution as the optimisation problem

$$\begin{array}{ll} \max_U & J(U) = \sum_{i=1}^{\ell} \|P_U(\boldsymbol{\phi}(\mathbf{x}_i))\|_2^2 \\ \text{subject to} & \dim U = k. \end{array} \tag{6.10}$$

Let $\mathbf{w}^1, \ldots, \mathbf{w}^k$ be a basis for a general space U expressed in the principal axes. We can then evaluate $J(U)$ as follows

$$\begin{aligned} J(U) &= \sum_{i=1}^{\ell} \|P_U(\boldsymbol{\phi}(\mathbf{x}_i))\|_2^2 = \sum_{i=1}^{\ell} \sum_{j=1}^{k} P_{\mathbf{w}^j}(\boldsymbol{\phi}(\mathbf{x}_i))^2 \\ &= \sum_{j=1}^{k} \sum_{i=1}^{\ell} P_{\mathbf{w}^j}(\boldsymbol{\phi}(\mathbf{x}_i))^2 = \sum_{j=1}^{k} \sum_{s=1}^{\ell} \left(\mathbf{w}_s^j\right)^2 \sum_{i=1}^{\ell} P_{\mathbf{u}_s}(\boldsymbol{\phi}(\mathbf{x}_i))^2 \\ &= \sum_{j=1}^{k} \sum_{s=1}^{\ell} \left(\mathbf{w}_s^j\right)^2 \lambda_s = \sum_{s=1}^{\ell} \lambda_s \sum_{j=1}^{k} \left(\mathbf{w}_s^j\right)^2 . \end{aligned}$$

Since, the $\mathbf{w}^j$ are orthogonal we must have

$$a_s = \sum_{j=1}^{k} \left(\mathbf{w}_s^j\right)^2 \leq 1,$$

for all s (consider extending to an orthonormal basis

$$\mathbf{W} = \left[\mathbf{w}^1 \cdots \mathbf{w}^k \mathbf{w}^{k+1} \cdots \mathbf{w}^{\ell}\right]$$

and observing that

$$\left(\mathbf{W}\mathbf{W}'\right)_{ss} = \sum_{j=1}^{\ell} \left(\mathbf{w}_s^j\right)^2 = 1$$

for all s), while

$$\sum_{s=1}^{\ell} a_s = \sum_{s=1}^{\ell} \sum_{j=1}^{k} \left(\mathbf{w}_s^j\right)^2 = \sum_{j=1}^{k} \sum_{s=1}^{\ell} \left(\mathbf{w}_s^j\right)^2 = k.$$

Therefore we have

$$J(U) = \sum_{s=1}^{\ell} \lambda_s a_s \leq \sum_{s=1}^{k} \lambda_s = J(U_k), \tag{6.11}$$

showing that U_k does indeed optimise both (6.7) and (6.10). The value of the optimum follows from (6.9), (6.11) and (6.6). □

Principal axes capturing variance If we take $k = \ell$ nothing is lost in the projection and so summing all the eigenvalues gives us the sum of the norms of the feature vectors

$$\sum_{i=1}^{\ell} \|\boldsymbol{\phi}(\mathbf{x}_i)\|^2 = \sum_{i=1}^{\ell} \lambda_i,$$

a fact that also follows from the invariance of the trace to the orthogonal transformation

$$\ell \mathbf{C} \longmapsto \ell \mathbf{U}'\mathbf{C}\mathbf{U} = \boldsymbol{\Lambda}_\ell.$$

The individual eigenvalues say how much of the sum of the norms squared lies in the space spanned by the ith eigenvector. By the above discussion the eigenvectors of the matrix $\mathbf{X}'\mathbf{X}$ give the directions of maximal variance of the data in descending order with the corresponding eigenvalues giving the size of the variance in that direction multiplied by ℓ. It is the fact that projection into the space U_k minimises the resulting average squared residual that motivates the use of these eigenvectors as a coordinate system.

We now consider how this analysis can be undertaken using only inner product information and hence exploiting a dual representation and kernels.

6.2.1 Kernel principal components analysis

Kernel PCA is the application of PCA in a kernel-defined feature space making use of the dual representation. Section 6.1 has demonstrated how projections onto the feature space eigenvectors can be computed through a dual representation computed from the eigenvectors and eigenvalues of the kernel matrix.

We now present the details of the kernel PCA algorithm before providing a stability analysis assessing when the resulting projection captures a stable pattern of the data. We continue to use U_k to denote the subspace spanned by the first k eigenvectors in the feature space. Using equation (6.4) we can compute the k-dimensional vector projection of new data into this subspace as

$$P_{U_k}\left(\boldsymbol{\phi}(\mathbf{x})\right) = \left(\mathbf{u}_j'\boldsymbol{\phi}(\mathbf{x})\right)_{j=1}^{k} = \left(\sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \kappa(\mathbf{x}_i, \mathbf{x})\right)_{j=1}^{k}, \tag{6.12}$$

where

$$\boldsymbol{\alpha}^j = \lambda_j^{-1/2}\mathbf{v}_j$$

is given in terms of the corresponding eigenvector and eigenvalue of the kernel matrix. Equation (6.12) forms the basis of kernel PCA.

Algorithm 6.13 [Kernel PCA] The kernel PCA algorithm performs the following computation:

input	Data $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, dimension k.
process	$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, \ldots, \ell$ $\mathbf{K} - \frac{1}{\ell}\mathbf{j}\mathbf{j}'\mathbf{K} - \frac{1}{\ell}\mathbf{K}\mathbf{j}\mathbf{j}' + \frac{1}{\ell^2}(\mathbf{j}'\mathbf{K}\mathbf{j})\,\mathbf{j}\mathbf{j}'$, $[\mathbf{V}, \boldsymbol{\Lambda}] = \mathrm{eig}(\mathbf{K})$ $\boldsymbol{\alpha}^j = \frac{1}{\sqrt{\lambda_j}}\mathbf{v}_j$, $j = 1, \ldots, k$. $\tilde{\mathbf{x}}_i = \left(\sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \kappa(\mathbf{x}_i, \mathbf{x})\right)_{j=1}^{k}$
output	Transformed data $\tilde{S} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_\ell\}$.

The Matlab code for this computation is given in Code Fragment 6.1. ∎

Figure 6.1 shows the first principal direction as a shading level for the sample data shown using primal PCA. Figure 6.2 shows the same data analysed using kernel PCA with a nonlinear kernel.

6.2.2 Stability of principal components analysis

The critical question for assessing the performance of kernel PCA is the extent to which the projection captures new data drawn according to the same distribution as the training data. The last line of the Matlab code in Code Fragment 6.1 computes the average residual of the test data. We would like to ensure that this is not much larger than the average residual of the training data given by the expression in the comment eight lines earlier. Hence, we assess the stability of kernel PCA through the pattern function

$$\begin{aligned} f(\mathbf{x}) &= \left\| P_{U_k}^{\perp}(\boldsymbol{\phi}(\mathbf{x})) \right\|^2 = \|\boldsymbol{\phi}(\mathbf{x}) - P_{U_k}(\boldsymbol{\phi}(\mathbf{x}))\|^2 \\ &= \|\boldsymbol{\phi}(\mathbf{x})\|^2 - \|P_{U_k}(\boldsymbol{\phi}(\mathbf{x}))\|^2, \end{aligned}$$

that is, the squared norm of the orthogonal (residual) projection for the subspace U_k spanned by the first k eigenvectors. As always we wish the expected value of the pattern function to be small

$$\mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] = \mathbb{E}_{\mathbf{x}}\left[\left\| P_{U_k}^{\perp}(\boldsymbol{\phi}(\mathbf{x})) \right\|^2\right] \approx 0.$$

```
 % K is the kernel matrix of the training points
 % inner products between ell training and t test points
 %   are stored in matrix Ktest of dimension (ell + 1) x t
 %   last entry in each column is inner product with self
 % k gives dimension of projection space
 % V is ell x k matrix storing the first k eigenvectors
  % L is k x k diagonal matrix with eigenvalues
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);
[V, L] = eigs(K, k, 'LM');
 invL = diag(1./diag(L));         % inverse of L
 sqrtL = diag(sqrt(diag(L)));     % sqrt of eigenvalues
 invsqrtL = diag(1./diag(sqrtL)); % inverse of sqrtL
 TestFeat = invsqrtL * V' * Ktest(1:20,:);
 TrainFeat = sqrtL * V'; % = invsqrtL * V' * K;
 % Note that norm(TrainFeat, 'fro') = sum-squares of
 %    norms of projections = sum(diag(L)).
 % Hence, average squared norm not captured (residual) =
 %    (sum(diag(K)) - sum(diag(L)))/ell
 % If we need the new inner product information:
 Knew = V * L * V'; % = TrainFeat' * TrainFeat;
 % between training and test
 Ktestnew = V * V' * Ktest(1:20,:);
 % and between test and test
 Ktestvstest = Ktest(1:20,:)'*V*invL*V'*Ktest(1:20,:);
 % The average sum-squared residual of the test points is
 (sum(Ktest(ell + 1,:)  - diag(Ktestvstest)')/t
```

Code Fragment 6.1. Matlab code for kernel PCA algorithm.

Our aim is to relate the empirical value of the residual given by the pattern function $f(\mathbf{x})$ to its expected value. Since the eigenvalues of $\ell\mathbf{C}$ and the kernel matrix $\mathbf{K}$ are the same, it follows from equation (6.8) that ℓ times the empirical average of the pattern function is just the sum of those eigenvalues from $k+1$ to ℓ. We introduce the notation $\lambda^{>t}(S) = \sum_{i=t+1}^{\ell} \lambda_i$ for these sums. Hence, the critical question is how much larger than the empirical expectation

$$\hat{\mathbb{E}}\left[\|P_{U_k}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\|^2\right] = \frac{1}{\ell}\lambda^{>t}(S)$$

is the true expectation

$$\mathbb{E}\left[\left\|P_{U_t}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right\|^2\right].$$

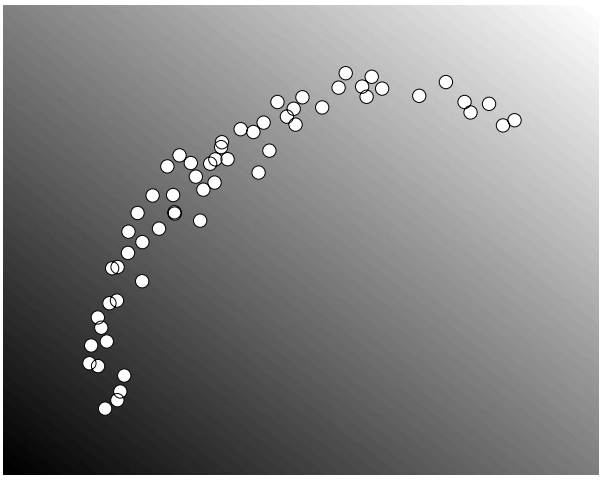

Fig. 6.1. The shading shows the value of the projection on to the first principal direction for linear PCA.

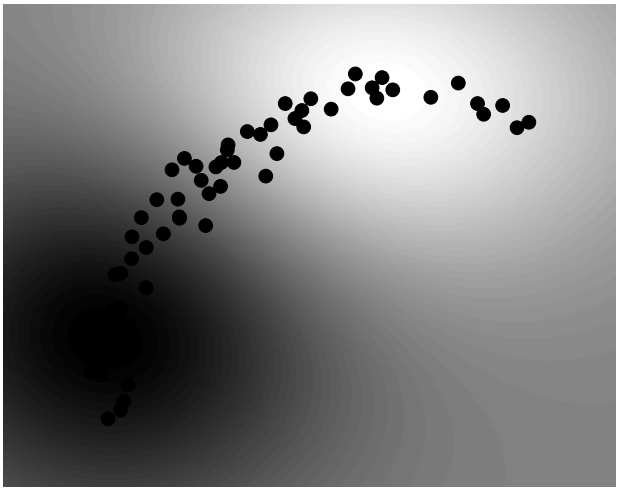

Fig. 6.2. The shading shows the the value of the projection on to the first principal direction for nonlinear PCA.

It is worth noting that if we can bound the difference between these for some value of t, for $k > t$ we have

$$\mathbb{E}\left[\left\|P_{U_k}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right\|^2\right] \leq \mathbb{E}\left[\left\|P_{U_t}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right\|^2\right],$$

so that the bound for t also applies to k-dimensional projections. This observation explains the min in the theorem below giving a bound on the difference between the two expectations.

Theorem 6.14 *If we perform PCA in the feature space defined by a kernel κ then with probability greater than $1-\delta$, for any $1 \leq k \leq \ell$, if we project new data onto the space U_k spanned by the first k eigenvectors in the feature space, the expected squared residual is bounded by*

$$\mathbb{E}\left[\left\|P_{U_k}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right\|^2\right] \leq \min_{1\leq t\leq k}\left[\frac{1}{\ell}\lambda^{>t}(S)+\frac{8}{\ell}\sqrt{(t+1)\sum_{i=1}^{\ell}\kappa(\mathbf{x}_i,\mathbf{x}_i)^2}\right] + 3R^2\sqrt{\frac{\ln(2\ell/\delta)}{2\ell}},$$

where the support of the distribution is in a ball of radius R in the feature space.

Remark 6.15 [The case of a Gaussian kernel] Reading of the theorem is simplified if we consider the case of a normalised kernel such as the Gaussian. In this case both R and $\kappa(\mathbf{x}_i,\mathbf{x}_i)$ are equal to 1 resulting in the bound

$$\mathbb{E}\left[\left\|P_{U_k}^{\perp}(\boldsymbol{\phi}(\mathbf{x}))\right\|^2\right] \leq \min_{1\leq t\leq k}\left[\frac{1}{\ell}\lambda^{>t}(S)+8\sqrt{\frac{(t+1)}{\ell}}\right]+3\sqrt{\frac{\ln(2\ell/\delta)}{2\ell}}.$$

Hence, Theorem 6.14 indicates that the expected squared residual of a test point will be small provided the residual eigenvalues are small for some value $t \leq k$, which is modest compared to ℓ. Hence, we should only use kernel PCA when the eigenvalues become small at an early stage in the spectrum. Provided we project into a space whose dimension exceeds the index of this stage, we will with high probability capture most of the variance of unseen data. ■

The overall message is that capturing a high proportion of the variance of the data in a number of dimensions significantly smaller than the samples size indicates that a reliable pattern has been detected and that the same subspace will, with high probability, capture most of the variance of the test data. We can therefore view the theorem as stating that the percentage of variance captured by low-dimensional eigenspaces is concentrated and hence reliably estimated from the training sample.

A proof of this theorem appears in Appendix A.2. The basis for the statistical analysis are the Rademacher complexity results of Chapter 4. The

difficulty in applying the method is that the function class does not appear to be linear, but interestingly it can be viewed as linear in the feature space defined by the quadratic kernel

$$\hat{\kappa}(\mathbf{x}, \mathbf{z}) = \kappa(\mathbf{x}, \mathbf{z})^2.$$

Hence, the use of kernels not only defines a feature space and provides the algorithmic tool to compute in that space, but also resurfaces as a proof technique for analysing the stability of principal components analysis. Though this provides an interesting and distinctive use of kernels we have preferred not to distract the reader from the main development of this chapter and have moved the proof details to an appendix.

Whitening PCA computed the directions of maximal variance and used them as the basis for dimensionality reduction. The resulting covariance matrix of the projected data retains the same eigenvalues corresponding to the eigenvectors used to define the projection space, but has a diagonal structure. This follows from the observation that given a centred data matrix $\mathbf{X}$, the projected data $\mathbf{X}\mathbf{U}_k$ has covariance matrix

$$\frac{1}{\ell}\mathbf{U}_k'\mathbf{X}'\mathbf{X}\mathbf{U}_k = \frac{1}{\ell}\mathbf{U}_k'\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}'\mathbf{U}_k = \frac{1}{\ell}\boldsymbol{\Lambda}_k.$$

Whitening is a technique that transforms the projected data to make the resulting covariance matrix equal to the identity by rescaling the projection directions by $\boldsymbol{\Lambda}_k^{-1/2}$ to obtain $\mathbf{X}\mathbf{U}_k\boldsymbol{\Lambda}_k^{-1/2}$, so that the covariance becomes

$$\begin{aligned}\frac{1}{\ell}\boldsymbol{\Lambda}_k^{-1/2}\mathbf{U}_k'\mathbf{X}'\mathbf{X}\mathbf{U}_k\boldsymbol{\Lambda}_k^{-1/2} &= \frac{1}{\ell}\boldsymbol{\Lambda}_k^{-1/2}\mathbf{U}_k'\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}'\mathbf{U}_k\boldsymbol{\Lambda}_k^{-1/2} = \frac{1}{\ell}\boldsymbol{\Lambda}_k^{-1/2}\boldsymbol{\Lambda}_k\boldsymbol{\Lambda}_k^{-1/2} \\ &= \frac{1}{\ell}\mathbf{I}.\end{aligned}$$

This is motivated by the desire to make the different directions have equal weight, though we will see a further motivation for this in Chapter 12. The transformation can be implemented as a variant of kernel PCA.

Algorithm 6.16 [Whitening] The whitening algorithm is given in Code Fragment 6.2. ■

6.3 Directions of maximum covariance

Principal components analysis measures the variance in the data by identifying the so-called principal axes that give the directions of maximal variance in decreasing importance. PCA sets a threshold and discards the principal directions for which the variance is below that threshold.

input	Data $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, dimension k.
process	$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, \ldots, \ell$ $\mathbf{K} - \frac{1}{\ell}\mathbf{j}\mathbf{j}'\mathbf{K} - \frac{1}{\ell}\mathbf{K}\mathbf{j}\mathbf{j}' + \frac{1}{\ell^2}(\mathbf{j}'\mathbf{K}\mathbf{j})\mathbf{j}\mathbf{j}'$, $[\mathbf{V}, \mathbf{\Lambda}] = \text{eig}(\mathbf{K})$ $\boldsymbol{\alpha}^j = \frac{1}{\lambda_j}\mathbf{v}_j$, $j = 1, \ldots, k$. $\tilde{\mathbf{x}}_i = \left(\sum_{i=1}^{\ell} \boldsymbol{\alpha}_i^j \kappa(\mathbf{x}_i, \mathbf{x})\right)_{j=1}^{k}$
output	Transformed data $\tilde{S} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_\ell\}$.

Code Fragment 6.2. Pseudocode for the whitening algorithm.

Consider for a moment that we are tackling a regression problem. Performing PCA as a precursor to finding a linear regressor is referred to as *principal components regression (PCR)* and is motivated mainly through its potential for denoising and hence reducing the variance of the resulting regression error. There is, however, a danger inherent in this approach in that what is important for the regression estimation is not the size of the variance of the data, but how well it can be used to predict the output. It might be that the high variance directions identified by PCA are uncorrelated with the target, while a direction with relatively low variance nonetheless has high predictive potential.

In this section we will begin to examine methods for measuring when directions carry information useful for prediction. This will allow us again to isolate directions that optimise the derived criterion. The key is to look for relationships between two random variables.

In Section 5.3 we defined the covariance of two zero-mean univariate random variables x and y as $\mathbb{E}[xy]$. This is in contrast to the correlation coefficient which normalises with respect to the variances of the two variables. We now consider extending our consideration to multidimensional random vectors.

Consider two multivariate random vectors giving rise to a dataset S containing pairs $(\mathbf{x}, \mathbf{y})$ from two different spaces X and Y. We call such a dataset *paired* in the sense that the process generating the data generates items in pairs, one from X and one from Y.

Example 6.17 For example, if we have a set of labelled examples for a supervised learning task, we can view it as a paired dataset by letting the input space be X and the output space be Y. If the labels are binary this makes examples from Y a Bernoulli sequence, but more generally for

regression $Y = \mathbb{R}$, and of course we can consider the case where $Y = \mathbb{R}^n$ or indeed has a more complex structure.

We are interested in studying the covariance between the two parts of a paired dataset even though those two parts live in different spaces. We achieve this by using an approach similar to that adopted to study the variance of random vectors. There we projected the data onto a direction vector $\mathbf{w}$ to create a univariate random variable, whose mean and standard deviation could subsequently be computed. Here we project the two parts onto two separate directions specified by unit vectors $\mathbf{w}_x$ and $\mathbf{w}_y$, to obtain two random variables $\mathbf{w}_x'\mathbf{x}$ and $\mathbf{w}_y'\mathbf{y}$ that are again univariate and hence whose covariance can be computed. In this way we can assess the relation between $\mathbf{x}$ and $\mathbf{y}$. Note that for the purposes of this exposition we are assuming that the input space is the feature space. When we come to apply this analysis in Section 6.7.1, we will introduce a kernel-defined feature space for the first component only. We give a definition of a paired dataset in which the two components correspond to distinct kernel mappings in Section 6.5.

Again following the analogy with the unsupervised case, given two directions $\mathbf{w}_x$ and $\mathbf{w}_y$, we can measure the covariance of the corresponding random variables as

$$\hat{\mathbb{E}}\left[\mathbf{w}_x'\mathbf{x}\mathbf{w}_y'\mathbf{y}\right] = \hat{\mathbb{E}}\left[\mathbf{w}_x'\mathbf{x}\mathbf{y}'\mathbf{w}_y\right] = \mathbf{w}_x'\hat{\mathbb{E}}\left[\mathbf{x}\mathbf{y}'\right]\mathbf{w}_y = \mathbf{w}_x'\mathbf{C}_{xy}\mathbf{w}_y,$$

where we have used $\mathbf{C}_{xy}$ to denote the sample covariance matrix $\hat{\mathbb{E}}\left[\mathbf{x}\mathbf{y}'\right]$ between X and Y. If we consider two matrices $\mathbf{X}$ and $\mathbf{Y}$ whose ith rows are the feature vectors of corresponding examples $\mathbf{x}_i$ and $\mathbf{y}_i$, we can write

$$\mathbf{C}_{xy} = \hat{\mathbb{E}}\left[\mathbf{x}\mathbf{y}'\right] = \frac{1}{\ell}\sum_{i=1}^{\ell}\mathbf{x}_i\mathbf{y}_i' = \frac{1}{\ell}\mathbf{X}'\mathbf{Y}.$$

Now that we are able to measure the covariance for a particular choice of directions, it is natural to ask if we can choose the directions to maximise this quantity. Hence, we would like to solve the following optimisation.

Computation 6.18 [Maximising Covariance] The directions $\mathbf{w}_x$, $\mathbf{w}_y$ of maximal covariance can be found as follows

$$\begin{array}{ll} \max_{\mathbf{w}_x,\mathbf{w}_y} & C(\mathbf{w}_x,\mathbf{w}_y) = \mathbf{w}_x'\mathbf{C}_{xy}\mathbf{w}_y = \frac{1}{\ell}\mathbf{w}_x'\mathbf{X}'\mathbf{Y}\mathbf{w}_y, \\ \text{subject to} & \|\mathbf{w}_x\|_2 = \|\mathbf{w}_y\|_2 = 1. \end{array} \tag{6.13}$$

We can again convert this to maximising a quotient by introducing an in-

variance to scaling

$$\max_{\mathbf{w}_x,\mathbf{w}_y} \frac{C(\mathbf{w}_x,\mathbf{w}_y)}{\|\mathbf{w}_x\|\,\|\mathbf{w}_y\|} = \max_{\mathbf{w}_x,\mathbf{w}_y} \frac{\mathbf{w}_x'\mathbf{C}_{xy}\mathbf{w}_y}{\|\mathbf{w}_x\|\,\|\mathbf{w}_y\|}. \tag{6.14}$$

■

Remark 6.19 [Relation to Rayleigh quotient] Note the similarity to the Rayleigh quotient considered above, but in this case $\mathbf{C}_{xy}$ is not a square matrix since its row dimension is equal to the dimension of X, while its column dimension is given by the dimension of Y. Furthermore, even if these dimensions were equal, $\mathbf{C}_{xy}$ would not be symmetric and here we are optimising over two vectors. ■

Proposition 6.20 *The directions that solve the maximal covariance optimisation (6.13) are the first singular vectors* $\mathbf{w}_x = \mathbf{u}_1$ *and* $\mathbf{w}_y = \mathbf{v}_1$ *of the singular value decomposition of* $\mathbf{C}_{xy}$

$$\mathbf{C}_{xy} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}';$$

the value of the covariance is given by the corresponding singular value σ_1.

Proof Using the singular value decomposition of $\mathbf{C}_{xy}$ and taking into account that $\mathbf{U}$ and $\mathbf{V}$ are orthornormal matrices so that, for example, $\|\mathbf{V}\mathbf{w}\| = \|\mathbf{w}\|$ and any $\mathbf{w}_x$ can be expressed as $\mathbf{U}\mathbf{u}_x$ for some $\mathbf{u}_x$, the solution to problem (6.13) becomes

$$\begin{aligned}
\max_{\mathbf{w}_x,\mathbf{w}_y:\|\mathbf{w}_x\|_2=\|\mathbf{w}_y\|_2=1} C(\mathbf{w}_x,\mathbf{w}_y) &= \max_{\mathbf{u}_x,\mathbf{v}_y:\|\mathbf{U}\mathbf{u}_x\|_2=\|\mathbf{V}\mathbf{v}_y\|_2=1} (\mathbf{U}\mathbf{u}_x)'\,\mathbf{C}_{xy}\mathbf{V}\mathbf{v}_y \\
&= \max_{\mathbf{u}_x,\mathbf{v}_y:\|\mathbf{u}_x\|_2=\|\mathbf{v}_y\|_2=1} \mathbf{u}_x'\mathbf{U}'\mathbf{U}\mathbf{\Sigma}\mathbf{V}'\mathbf{V}\mathbf{v}_y \\
&= \max_{\mathbf{u}_x,\mathbf{v}_y:\|\mathbf{u}_x\|_2=\|\mathbf{v}_y\|_2=1} \mathbf{u}_x'\mathbf{\Sigma}\mathbf{v}_y.
\end{aligned}$$

The last line clearly has a maximum of the largest singular value σ_1, when we take $\mathbf{u}_x = \mathbf{e}_1$ and $\mathbf{v}_y = \mathbf{e}_1$ the first unit vector (albeit of different dimensions). Hence, the original problem is solved by taking $\mathbf{w}_x = \mathbf{u}_1 = \mathbf{U}\mathbf{e}_1$ and $\mathbf{w}_y = \mathbf{v}_1 = \mathbf{V}\mathbf{e}_1$, the first columns of $\mathbf{U}$ and $\mathbf{V}$ respectively. □

Proposition 6.20 shows how to compute the directions that maximise the covariance. If we wish to identify more than one direction, as we did for example with the principal components, we must apply the same strategy of projecting the data onto the orthogonal complement by deflation. From equation (5.8), this corresponds to the operations

$$\mathbf{X} \longleftarrow \mathbf{X}\left(\mathbf{I} - \mathbf{u}_1\mathbf{u}_1'\right) \text{ and } \mathbf{Y} \longleftarrow \mathbf{Y}\left(\mathbf{I} - \mathbf{v}_1\mathbf{v}_1'\right).$$

The resulting covariance matrix is therefore

$$\begin{aligned}\frac{1}{\ell}\left(\mathbf{I}-\mathbf{u}_1\mathbf{u}_1'\right)\mathbf{X}'\mathbf{Y}\left(\mathbf{I}-\mathbf{v}_1\mathbf{v}_1'\right) &= \left(\mathbf{I}-\mathbf{u}_1\mathbf{u}_1'\right)\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}'\left(\mathbf{I}-\mathbf{v}_1\mathbf{v}_1'\right)\\ &= \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}'-\sigma_1\mathbf{u}_1\mathbf{v}_1'\\ &= \mathbf{C}_{xy}-\sigma_1\mathbf{u}_1\mathbf{v}_1',\end{aligned}$$

implying that this corresponds to the deflation procedure for singular value decomposition given in equation (6.1). The next two directions of maximal covariance will now be given by the second singular vectors $\mathbf{u}_2$ and $\mathbf{v}_2$ with the value of the covariance given by σ_2. Proceeding in similar fashion we see that the singular vectors give the orthogonal directions of maximal covariance in descending order. This provides a series of directions in X and in Y that have the property of being maximally covariant resulting in the singular value decomposition of $\mathbf{C}_{xy}$

$$\mathbf{C}_{xy}=\sum_{i=1}^{\ell}\sigma_i\mathbf{u}_i\mathbf{v}_i'.$$

Computation and dual form If we wish to avoid performing a singular value decomposition of $\mathbf{C}_{xy}$, for example when working in a kernel-defined feature space, we can find the singular vectors through an eigenanalysis of the matrix $\mathbf{C}_{xy}\mathbf{C}_{xy}'$, to obtain $\mathbf{U}$, and of $\mathbf{C}_{xy}'\mathbf{C}_{xy}$, to obtain $\mathbf{V}$. Incidentally, this also reminds us that the singular directions are orthogonal, since they are the eigenvectors of a symmetric matrix. Now observe that

$$\mathbf{C}_{xy}'\mathbf{C}_{xy}=\frac{1}{\ell^2}\mathbf{Y}'\mathbf{X}\mathbf{X}'\mathbf{Y}=\frac{1}{\ell^2}\mathbf{Y}'\mathbf{K}_x\mathbf{Y},$$

where $\mathbf{K}_x$ is the kernel matrix associated with the space X. The dimension of this system will be N_y, the same as that of the Y space. It follows from a direct comparison with PCA that

$$\mathbf{u}_j=\frac{1}{\sigma_j}\mathbf{C}_{xy}\mathbf{v}_j.$$

Hence, the projection of a new point $\boldsymbol{\phi}(\mathbf{x})$ onto $\mathbf{u}_j$ is given by

$$\mathbf{u}_j'\boldsymbol{\phi}(\mathbf{x})=\frac{1}{\ell\sigma_j}\mathbf{v}_j'\mathbf{Y}'\mathbf{X}\boldsymbol{\phi}(\mathbf{x})=\sum_{i=1}^{\ell}\boldsymbol{\alpha}_i^j\kappa(\mathbf{x}_i,\mathbf{x}),$$

where

$$\boldsymbol{\alpha}^j=\frac{1}{\ell\sigma_j}\mathbf{Y}\mathbf{v}_j.$$

Remark 6.21 [On stability analysis] We do not provide a stability analysis for the features selected by maximising the covariance, though it is clear that we can view them as eigenvectors of a corresponding eigen-decomposition based on a sample estimation of covariances. Hence, similar techniques to those used in Appendix A.2 could be used to show that provided the number of features extracted is small compared to the size of the sample, we can expect the test example performance to mimic closely that of the training sample. ■

Alternative characterisation There is another characterisation of the largest singular vectors that motivates their use in choosing a prediction function from X to Y in the case of a supervised learning problem with $Y = \mathbb{R}^n$. We will discuss multi-variate regression in more detail at the end of the chapter, but present the characterisation here to complement the covariance approach presented above. The approach focuses on the choice of the orthogonal matrices of the singular value decomposition.

Suppose that we seek orthogonal matrices $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ such that the columns of $\mathbf{S} = \mathbf{X}\hat{\mathbf{U}}$ and $\mathbf{T} = \mathbf{Y}\hat{\mathbf{V}}$ are as similar as possible. By this we mean that we seek to minimise a simple discrepancy D between $\mathbf{S}$ and $\mathbf{T}$ defined as

$$D(\hat{\mathbf{U}}, \hat{\mathbf{V}}) = \sum_{i=1}^{m} |\mathbf{s}_i - \mathbf{t}_i|^2 + \sum_{i=m+1}^{n} |\mathbf{s}_i|^2, \tag{6.15}$$

where we have assumed that $\mathbf{S}$ has more columns than $\mathbf{T}$. If we let $\bar{\mathbf{T}} = [\mathbf{T}, \mathbf{0}]$, or in other words $\mathbf{T}$ is padded with 0s to the size of $\mathbf{S}$, we have

$$\begin{aligned} D(\hat{\mathbf{U}}, \hat{\mathbf{V}}) &= \left\|\mathbf{S} - \bar{\mathbf{T}}\right\|_F^2 = \left\langle \mathbf{S} - \bar{\mathbf{T}}, \mathbf{S} - \bar{\mathbf{T}} \right\rangle_F \\ &= \left\langle \mathbf{S}, \mathbf{S} \right\rangle_F - 2\left\langle \mathbf{S}, \bar{\mathbf{T}} \right\rangle_F + \left\langle \bar{\mathbf{T}}, \bar{\mathbf{T}} \right\rangle_F \\ &= \operatorname{tr} \mathbf{S}'\mathbf{S} - 2\operatorname{tr} \mathbf{S}'\mathbf{T} + \operatorname{tr} \mathbf{T}'\mathbf{T} \\ &= \operatorname{tr} \hat{\mathbf{U}}'\mathbf{X}'\mathbf{X}\hat{\mathbf{U}} - 2\operatorname{tr} \hat{\mathbf{U}}'\mathbf{X}'\mathbf{Y}\hat{\mathbf{V}} + \operatorname{tr} \hat{\mathbf{V}}'\mathbf{Y}'\mathbf{Y}\hat{\mathbf{V}} \\ &= \operatorname{tr} \mathbf{X}'\mathbf{X} + \operatorname{tr} \mathbf{Y}'\mathbf{Y} - 2\operatorname{tr} \hat{\mathbf{U}}'\mathbf{X}'\mathbf{Y}\hat{\mathbf{V}}. \end{aligned}$$

Hence, the maximum of D is obtained when $\operatorname{tr} \hat{\mathbf{U}}'\mathbf{X}'\mathbf{Y}\hat{\mathbf{V}}$ is minimised. But we have

$$\operatorname{tr} \hat{\mathbf{U}}'\mathbf{X}'\mathbf{Y}\hat{\mathbf{V}} = \ell \operatorname{tr} \hat{\mathbf{U}}'\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}'\hat{\mathbf{V}} = \ell \operatorname{tr} \tilde{\mathbf{V}}\tilde{\mathbf{U}}'\boldsymbol{\Sigma},$$

for appropriately sized orthogonal matrices $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{U}}$. Since multiplying by an orthogonal matrix from the left will not change the two-norm of the columns, the value of the expression is clearly maximised when $\tilde{\mathbf{V}}\tilde{\mathbf{U}}' = \mathbf{I}$,

the identity matrix. Hence, the choice of $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ that minimises $D(\hat{\mathbf{U}}, \hat{\mathbf{V}})$ is the orthogonal matrices of the singular value decomposition.

Before we continue our exploration of patterns that can be identified using eigen-decompositions, we must consider a more expanded class of techniques that solve the so-called generalised eigenvector problem.

6.4 The generalised eigenvector problem

A number of problems in kernel-based pattern analysis can be reduced to solving a generalised eigenvalue problem, a standard problem in multivariate statistics

$$\mathbf{A}\mathbf{w} = \lambda \mathbf{B}\mathbf{w}$$

with $\mathbf{A}$, $\mathbf{B}$ symmetric matrices, $\mathbf{B}$ positive definite. Hence, the normal eigenvalue problem is a special case obtained by taking $\mathbf{B} = \mathbf{I}$, the identity matrix. The problem arises as the solution of the maximisation of a generalised Rayleigh quotient

$$\rho(\mathbf{w}) = \frac{\mathbf{w}'\mathbf{A}\mathbf{w}}{\mathbf{w}'\mathbf{B}\mathbf{w}},$$

which has a positive quadratic form rather than a simple norm squared in the denominator. Since the ratio is invariant to rescaling of the vector $\mathbf{w}$, we can maximise the ratio by constraining the denominator to have value 1. Hence, the maximum quotient problem can be cast as the optimization problem

$$\begin{array}{ll} \max & \mathbf{w}'\mathbf{A}\mathbf{w} \\ \text{subject to} & \mathbf{w}'\mathbf{B}\mathbf{w} = 1. \end{array} \tag{6.16}$$

Applying the Lagrange multiplier technique and differentiating with respect to $\mathbf{w}$ we arrive at the generalised eigenvalue problem

$$\mathbf{A}\mathbf{w} - \lambda \mathbf{B}\mathbf{w} = \mathbf{0}. \tag{6.17}$$

Since by assumption $\mathbf{B}$ is positive definite we can convert to a standard eigenvalue problem by premultiplying by $\mathbf{B}^{-1}$ to obtain

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{w} = \lambda \mathbf{w}.$$

But note that although both $\mathbf{A}$ and $\mathbf{B}$ are assumed to be symmetric, $\mathbf{B}^{-1}\mathbf{A}$ need not be. Hence we cannot make use of the main results of Section 3.1. In particular the eigenvectors will not in general be orthogonal. There is, however, a related symmetric eigenvalue problem that reveals something

about the structure of the eigenvectors of (6.16). Since $\mathbf{B}$ is positive definite it possesses a symmetric square root $\mathbf{B}^{1/2}$ with the property that

$$\mathbf{B}^{1/2}\mathbf{B}^{1/2} = \mathbf{B}.$$

Consider premultiplying (6.17) by $\mathbf{B}^{1/2}$ and reparametrise the solution vector $\mathbf{w}$ as $\mathbf{B}^{-1/2}\mathbf{v}$. We obtain the standard eigenvalue problem

$$\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}\mathbf{v} = \lambda\mathbf{v}, \tag{6.18}$$

where now the matrix $\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2} = (\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2})'$ is symmetric. Applying the results of Chapter 3, we can find a set of orthonormal eigenvector solutions of (6.18) λ_i, $\mathbf{v}_i$. Hence, the solutions of (6.17) have the form

$$\mathbf{w}_i = \mathbf{B}^{-1/2}\mathbf{v}_i,$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ are the orthonormal eigenvectors of (6.18) with the associated eigenvalues being the same. Since $\mathbf{B}^{1/2}$ is a bijection of the space $\mathbb{R}^\ell$ we can write

$$\rho = \frac{\mathbf{w}'\mathbf{A}\mathbf{w}}{\mathbf{w}'\mathbf{B}\mathbf{w}} = \frac{\left(\mathbf{B}^{1/2}\mathbf{w}\right)'\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}\left(\mathbf{B}^{1/2}\mathbf{w}\right)}{\left\|\mathbf{B}^{1/2}\mathbf{w}\right\|^2}$$

the generalised Rayleigh quotient is given by the associated Rayleigh quotient for the standard eigenvalue problem (6.18) after the bijection $\mathbf{B}^{1/2}$ has been applied. We can therefore see the generalised eigenvalue problem as an eigenvalue problem in a transformed space. The following propositions are simple consequences of these observations.

Proposition 6.22 *Any vector* $\mathbf{v}$ *can be written as a linear combination of the eigenvectors* $\mathbf{w}_i$, $i = 1, \ldots, \ell$. *The generalised eigenvectors of the problem* $\mathbf{A}\mathbf{w} = \lambda\mathbf{B}\mathbf{w}$ *have the following generalised orthogonality properties: if the eigenvalues are distinct, then in the metrics defined by* $\mathbf{A}$ *and* $\mathbf{B}$, *the eigenvectors are orthonormal*

$$\begin{aligned} \mathbf{w}_i'\mathbf{B}\mathbf{w}_j &= \delta_{ij} \\ \mathbf{w}_i'\mathbf{A}\mathbf{w}_j &= \delta_{ij}\lambda_i. \end{aligned}$$

Proof For $i \neq j$ we have (assuming without loss of generality that $\lambda_j \neq 0$)

$$0 = \mathbf{v}_i'\mathbf{v}_j = \mathbf{w}_i'\mathbf{B}^{1/2}\mathbf{B}^{1/2}\mathbf{w}_j = \mathbf{w}_i'\mathbf{B}\mathbf{w}_j = \frac{1}{\lambda_j}\mathbf{w}_i'\mathbf{A}\mathbf{w}_j,$$

which gives the result for $i \neq j$. Now consider

$$\lambda_i = \lambda_i\mathbf{v}_i'\mathbf{v}_i = \lambda_i\mathbf{w}_i'\mathbf{B}^{1/2}\mathbf{B}^{1/2}\mathbf{w}_i = \lambda_i\mathbf{w}_i'\mathbf{B}\mathbf{w}_i = \mathbf{w}_i'\mathbf{A}\mathbf{w}_i,$$

which covers the case of $i = j$. □

Definition 6.23 [Conjugate vectors] The first property

$$\mathbf{w}_i' \mathbf{B} \mathbf{w}_j = \delta_{ij}, \text{ for } i, j = 1, \ldots, \ell,$$

is also referred to as *conjugacy with respect to* $\mathbf{B}$, or equivalently that the vectors $\mathbf{w}_i$ are *conjugate*. ■

Proposition 6.24 *There is a global maximum and minimum of the generalised Rayleigh quotient. The quotient is bounded by the smallest and the largest eigenvalue*

$$\rho_\ell \leq \rho \leq \rho_1,$$

so that the global maximum ρ_1 *is attained by the associated eigenvector.*

Remark 6.25 [Second derivatives] We can also study the stationary points, by examining the second derivative or Hessian at the eigenvectors

$$\mathbf{H} = \frac{\partial^2 \rho}{\partial \mathbf{w}^2}|_{\mathbf{w}=\mathbf{w}_i} = \frac{2}{\mathbf{w}_i' \mathbf{B} \mathbf{w}_i}(\mathbf{A} - \rho_i \mathbf{B}).$$

For all $1 < i < \ell$, $\mathbf{H}$ has positive and negative eigenvalues, since

$$\left(\mathbf{B}^{-1/2}\mathbf{v}_1\right)' \left(\mathbf{A} - \rho_i \mathbf{B}\right) \mathbf{B}^{-1/2}\mathbf{v}_1 = \mathbf{w}_1' \mathbf{A} \mathbf{w}_1 - \rho_i = \rho_1 - \rho_i > 0,$$

while

$$\left(\mathbf{B}^{-1/2}\mathbf{v}_\ell\right)' \left(\mathbf{A} - \rho_i \mathbf{B}\right) \mathbf{B}_{\mathbf{v}_\ell}^{-1/2} = \mathbf{w}_\ell' \mathbf{A} \mathbf{w}_\ell - \rho_i = \rho_\ell - \rho_i < 0.$$

It follows that all the eigensolutions besides the largest and smallest are saddle points. ■

Proposition 6.26 *If* λ_i, $\mathbf{w}_i$ *are the eigenvalues and eigenvectors of the generalised eigenvalue problem*

$$\mathbf{A}\mathbf{w} = \lambda \mathbf{B}\mathbf{w},$$

then the matrix $\mathbf{A}$ *can be decomposed as*

$$\mathbf{A} = \sum_{i=1}^{\ell} \lambda_i \mathbf{B}\mathbf{w}_i \left(\mathbf{B}\mathbf{w}_i\right)'.$$

Proof We can decompose

$$\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2} = \sum_{i=1}^{\ell} \lambda_i \mathbf{v}_i \mathbf{v}_i',$$

implying that

$$\mathbf{A} = \sum_{i=1}^{\ell} \lambda_i \mathbf{B}^{1/2}\mathbf{v}_i \left(\mathbf{B}^{1/2}\mathbf{v}_i\right)' = \sum_{i=1}^{\ell} \lambda_i \mathbf{B}\mathbf{w}_i \left(\mathbf{B}\mathbf{w}_i\right)',$$

as required. □

Definition 6.27 [Generalised deflation] The final proposition suggests how we can deflate the matrix $\mathbf{A}$ in an iterative direct solution of the generalised eigenvalue problem

$$\mathbf{A}\mathbf{w} = \lambda \mathbf{B}\mathbf{w}.$$

After finding a non-zero eigenvalue–eigenvector pair λ, $\mathbf{w}$ we deflate $\mathbf{A}$ by

$$\mathbf{A} \longleftarrow \mathbf{A} - \lambda \mathbf{B}\mathbf{w}\left(\mathbf{B}\mathbf{w}\right)' = \mathbf{A} - \lambda \mathbf{B}\mathbf{w}\mathbf{w}'\mathbf{B}',$$

leaving $\mathbf{B}$ unchanged. ■

6.5 Canonical correlation analysis

We have looked at two ways of detecting stable patterns through the use of eigen-decompositions firstly to optimise variance of the training data in kernel PCA and secondly to maximise the covariance between two views of the data typically input and output vectors. We now again consider the case in which we have two views of the data which are paired in the sense that each example as a pair of representations. This situation is sometimes referred to as a paired dataset. We will show how to find correlations between the two views.

An extreme case would be where the second view is simply the labels of the examples. In general we are interested here in cases where we have a more complex 'output' that amounts to a different representation of the same object.

Example 6.28 A set of documents containing each document in two different languages is a paired dataset. The two versions give different views of the same underlying object, in this case the semantic content of the document. Such a dataset is known as a parallel corpus. By seeking correlations between the two views, we might hope to extract features that bring out

the underlying semantic content. The fact that a pattern has been found in both views suggests that it is not related to the irrelevant representation specific aspects of one or other view, but rather to the common underlying semantic content. This example will be explored further in Chapter 10.

This section will develop the methodology for finding these common patterns in different views through seeking correlations between projection values from the two views. Using an appropriate regularisation technique, the methods are extended to kernel-defined feature spaces.

Recall that in Section 5.3 we defined the correlation between two zero-mean univariate random variables x and y to be

$$\rho = \text{corr}(x, y) = \frac{\mathbb{E}[xy]}{\sqrt{\mathbb{E}[xx]\,\mathbb{E}[yy]}} = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)}\sqrt{\text{var}(y)}}.$$

Definition 6.29 [Paired dataset] A paired dataset is created when each object $\mathbf{x} \in X$ can be viewed through two distinct projections into two feature spaces

$$\boldsymbol{\phi}_a : \mathbf{x} \longrightarrow F_a \quad \text{and } \boldsymbol{\phi}_b : \mathbf{x} \longrightarrow F_b,$$

where F_a is the feature space associated with one representation and F_b the feature space for the other. Figure 6.3 illustrates this configuration. The corresponding kernel functions are denoted κ_a and κ_b. Hence, we have a multivariate random vector $(\boldsymbol{\phi}_a(\mathbf{x}), \boldsymbol{\phi}_b(\mathbf{x}))$. Assume we are given a training set

$$S = \{(\boldsymbol{\phi}_a(\mathbf{x}_1), \boldsymbol{\phi}_b(\mathbf{x}_1)), \ldots, (\boldsymbol{\phi}_a(\mathbf{x}_\ell), \boldsymbol{\phi}_b(\mathbf{x}_\ell))\}$$

drawn independently at random according to the underlying distribution. We will refer to such a set as a *paired or aligned dataset* in the feature space defined by the kernels κ_a and κ_b. ■

We now seek to maximise the empirical correlation between $x_a = \mathbf{w}_a' \boldsymbol{\phi}_a(\mathbf{x})$ and $x_b = \mathbf{w}_b' \boldsymbol{\phi}_b(\mathbf{x})$ over the projection directions $\mathbf{w}_a$ and $\mathbf{w}_b$

$$\begin{aligned}
\max\ \rho &= \frac{\hat{\mathbb{E}}[x_a x_b]}{\sqrt{\hat{\mathbb{E}}[x_a x_a]\,\hat{\mathbb{E}}[x_b x_b]}} \\
&= \frac{\hat{\mathbb{E}}\left[\mathbf{w}_a' \boldsymbol{\phi}_a(\mathbf{x})\, \boldsymbol{\phi}_b(\mathbf{x})' \mathbf{w}_b\right]}{\sqrt{\hat{\mathbb{E}}\left[\mathbf{w}_a' \boldsymbol{\phi}_a(\mathbf{x})\, \boldsymbol{\phi}_a(\mathbf{x})' \mathbf{w}_a\right] \hat{\mathbb{E}}\left[\mathbf{w}_b' \boldsymbol{\phi}_b(\mathbf{x})\, \boldsymbol{\phi}_b(\mathbf{x})' \mathbf{w}_b\right]}}
\end{aligned}$$

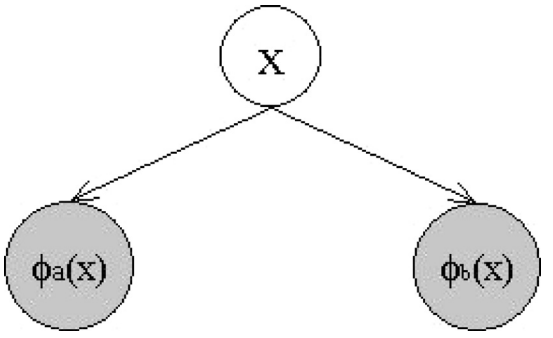

Fig. 6.3. The two embeddings of a paired dataset.

$$= \frac{\mathbf{w}_a' \mathbf{C}_{ab} \mathbf{w}_b}{\sqrt{\mathbf{w}_a' \mathbf{C}_{aa} \mathbf{w}_a \mathbf{w}_b' \mathbf{C}_{bb} \mathbf{w}_b}}, \tag{6.19}$$

where we have decomposed the empirical covariance matrix as follows

$$\begin{aligned}
\mathbf{C} &= \frac{1}{\ell} \sum_{i=1}^{\ell} (\boldsymbol{\phi}_a(\mathbf{x}), \boldsymbol{\phi}_b(\mathbf{x}))(\boldsymbol{\phi}_a(\mathbf{x}), \boldsymbol{\phi}_b(\mathbf{x}))' \\
&= \begin{pmatrix} \frac{1}{\ell}\sum_{i=1}^{\ell} \boldsymbol{\phi}_a(\mathbf{x})\boldsymbol{\phi}_a(\mathbf{x})' & \frac{1}{\ell}\sum_{i=1}^{\ell} \boldsymbol{\phi}_b(\mathbf{x})\boldsymbol{\phi}_a(\mathbf{x})' \\ \frac{1}{\ell}\sum_{i=1}^{\ell} \boldsymbol{\phi}_a(\mathbf{x})\boldsymbol{\phi}_b(\mathbf{x})' & \frac{1}{\ell}\sum_{i=1}^{\ell} \boldsymbol{\phi}_b(\mathbf{x})\boldsymbol{\phi}_b(\mathbf{x})' \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{C}_{aa} & \mathbf{C}_{ba} \\ \mathbf{C}_{ab} & \mathbf{C}_{bb} \end{pmatrix}.
\end{aligned}$$

This optimisation is very similar to that given in (6.14). The only difference is that here the denominator of the quotient measures the norm of the projection vectors differently from the covariance case. In the current optimisation the vectors $\mathbf{w}_a$ and $\mathbf{w}_b$ are again only determined up to direction since rescaling $\mathbf{w}_a$ by λ_a and $\mathbf{w}_b$ by λ_b results in the quotient

$$\begin{aligned}
\frac{\lambda_a \lambda_b \mathbf{w}_a' \mathbf{C}_{ab} \mathbf{w}_b}{\sqrt{\lambda_a^2 \mathbf{w}_a' \mathbf{C}_{aa} \mathbf{w}_a \lambda_b^2 \mathbf{w}_b' \mathbf{C}_{bb} \mathbf{w}_b}} &= \frac{\lambda_a \lambda_b \mathbf{w}_a' \mathbf{C}_{ab} \mathbf{w}_b}{\lambda_a \lambda_b \sqrt{\mathbf{w}_a' \mathbf{C}_{aa} \mathbf{w}_a \mathbf{w}_b' \mathbf{C}_{bb} \mathbf{w}_b}} \\
&= \frac{\mathbf{w}_a' \mathbf{C}_{ab} \mathbf{w}_b}{\sqrt{\mathbf{w}_a' \mathbf{C}_{aa} \mathbf{w}_a \mathbf{w}_b' \mathbf{C}_{bb} \mathbf{w}_b}}.
\end{aligned}$$

This implies that we can constrain the two terms in the denominator to individually have value 1. Hence, the problem is solved by the following optimisation problem.

Computation 6.30 [CCA] Given a paired dataset with covariance matrix

$\mathbf{C}_{ab}$, canonical correlation analysis finds the directions $\mathbf{w}_a$,$\mathbf{w}_b$ that maximise the correlation of corresponding projections by solving

$$\begin{array}{ll} \max_{\mathbf{w}_a,\mathbf{w}_b} & \mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b \\ \text{subject to} & \mathbf{w}_a'\mathbf{C}_{aa}\mathbf{w}_a = 1 \text{ and } \mathbf{w}_b'\mathbf{C}_{bb}\mathbf{w}_b = 1. \end{array} \tag{6.20}$$

■

Solving CCA Applying the Lagrange multiplier technique to the optimisation (6.20) gives

$$\max \quad \mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b - \frac{\lambda_a}{2}\left(\mathbf{w}_a'\mathbf{C}_{aa}\mathbf{w}_a - 1\right) - \frac{\lambda_y}{2}\left(\mathbf{w}_b'\mathbf{C}_{bb}\mathbf{w}_b - 1\right).$$

Taking derivatives with respect to $\mathbf{w}_a$ and $\mathbf{w}_b$ we obtain the equations

$$\mathbf{C}_{ab}\mathbf{w}_b - \lambda_a\mathbf{C}_{aa}\mathbf{w}_a = \mathbf{0} \quad \text{and} \quad \mathbf{C}_{ba}\mathbf{w}_a - \lambda_b\mathbf{C}_{bb}\mathbf{w}_b = \mathbf{0}. \tag{6.21}$$

Subtracting $\mathbf{w}_a'$ times the first from $\mathbf{w}_b'$ times the second we have

$$\lambda_a\mathbf{w}_a'\mathbf{C}_{aa}\mathbf{w}_a - \lambda_b\mathbf{w}_b'\mathbf{C}_{bb}\mathbf{w}_b = 0,$$

which, taking into account the two constraints, implies $\lambda_a = \lambda_b$. Using λ to denote this value we obtain the following algorithm for computing the correlations.

Algorithm 6.31 [Primal CCA] The following method finds the directions of maximal correlation:

Input	covariance matrices $\mathbf{C}_{aa}$, $\mathbf{C}_{bb}$, $\mathbf{C}_{ba}$ and $\mathbf{C}_{ab}$
Process	solve the generalised eigenvalue problem: $\begin{pmatrix} \mathbf{0} & \mathbf{C}_{ab} \\ \mathbf{C}_{ba} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{pmatrix} = \lambda\begin{pmatrix} \mathbf{C}_{aa} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{bb} \end{pmatrix}\begin{pmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{pmatrix}$
Output	eigenvectors and eigenvalues $\mathbf{w}_a^j$, $\mathbf{w}_b^j$ and $\lambda_j > 0$, $j = 1, \ldots, \ell$.

(6.22)

■

This is an example of a generalised eigenvalue problem described in the last section. Note that the value of the eigenvalue for a particular eigenvector gives the size of the correlation since $\mathbf{w}_a'$ times the top portion of (6.22) gives

$$\rho = \mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b = \lambda_a\mathbf{w}_a'\mathbf{C}_{aa}\mathbf{w}_a = \lambda.$$

Hence, we have all eigenvalues lying in the interval $[-1, +1]$, with each λ_i and eigenvector

$$\begin{pmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{pmatrix}$$

paired with an eigenvalue $-\lambda_i$ with eigenvector

$$\begin{pmatrix} \mathbf{w}_a \\ -\mathbf{w}_b \end{pmatrix}.$$

We are therefore only interested in half the spectrum which we can take to be the positive eigenvalues. The eigenvectors corresponding to the largest eigenvalues are those that identify the strongest correlations. Note that in this case by Proposition 6.22 the eigenvectors will be conjugate with respect to the matrix

$$\begin{pmatrix} \mathbf{C}_{aa} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{bb} \end{pmatrix},$$

so that for $i \neq j$ we have

$$0 = \begin{pmatrix} \mathbf{w}_a^j \\ \mathbf{w}_b^j \end{pmatrix}' \begin{pmatrix} \mathbf{C}_{aa} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{bb} \end{pmatrix} \begin{pmatrix} \mathbf{w}_a^i \\ \mathbf{w}_b^i \end{pmatrix} = \left(\mathbf{w}_a^j\right)' \mathbf{C}_{aa} \mathbf{w}_a^i + \left(\mathbf{w}_b^j\right)' \mathbf{C}_{bb} \mathbf{w}_b^i$$

and

$$0 = \begin{pmatrix} \mathbf{w}_a^j \\ \mathbf{w}_b^j \end{pmatrix}' \begin{pmatrix} \mathbf{C}_{aa} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{bb} \end{pmatrix} \begin{pmatrix} \mathbf{w}_a^i \\ -\mathbf{w}_b^i \end{pmatrix} = \left(\mathbf{w}_a^j\right)' \mathbf{C}_{aa} \mathbf{w}_a^i - \left(\mathbf{w}_b^j\right)' \mathbf{C}_{bb} \mathbf{w}_b^i$$

yielding

$$\left(\mathbf{w}_a^j\right)' \mathbf{C}_{aa} \mathbf{w}_a^i = 0 = \left(\mathbf{w}_b^j\right)' \mathbf{C}_{bb} \mathbf{w}_b^i.$$

This implies that, as with PCA, we obtain a diagonal covariance matrix if we project the data into the coordinate system defined by the eigenvectors, whether we project each view independently or simply the sum of the projections of the two views in the common space. The directions themselves will not, however, be orthogonal in the standard inner product of the feature space.

Dual form of CCA Naturally we wish to solve the problem in the dual formulation. Hence, we consider expressing $\mathbf{w}_a$ and $\mathbf{w}_b$ in terms of their respective parts of the training sample by creating a matrix $\mathbf{X}_a$ whose rows are the vectors $\phi_a(\mathbf{x}_i)$, $i = 1, \ldots, \ell$ and the matrix $\mathbf{X}_b$ with rows $\phi_b(\mathbf{x}_i)$

$$\mathbf{w}_a = \mathbf{X}_a' \boldsymbol{\alpha}_a \text{ and } \mathbf{w}_b = \mathbf{X}_b' \boldsymbol{\alpha}_b.$$

Substituting into (6.20) gives

$$\begin{array}{ll} \max & \boldsymbol{\alpha}_a' \mathbf{X}_a \mathbf{X}_a' \mathbf{X}_b \mathbf{X}_b' \boldsymbol{\alpha}_b \\ \text{subject to} & \boldsymbol{\alpha}_a' \mathbf{X}_a \mathbf{X}_a' \mathbf{X}_a \mathbf{X}_a' \boldsymbol{\alpha}_a = 1 \text{ and } \boldsymbol{\alpha}_b' \mathbf{X}_b \mathbf{X}_b' \mathbf{X}_b \mathbf{X}_b' \boldsymbol{\alpha}_b = 1, \end{array}$$

or equivalently the following optimisation problem.

Computation 6.32 [Kernel CCA] Given a paised dataset with respect to kernels κ_a and κ_b, kernel canonical correlation analysis finds the directions of maximal correlation by solving

$$\begin{array}{ll} \max_{\boldsymbol{\alpha}_a, \boldsymbol{\alpha}_b} & \boldsymbol{\alpha}_a' \mathbf{K}_a \mathbf{K}_b \boldsymbol{\alpha}_b \\ \text{subject to} & \boldsymbol{\alpha}_a' \mathbf{K}_a^2 \boldsymbol{\alpha}_a = 1 \text{ and } \boldsymbol{\alpha}_b' \mathbf{K}_b^2 \boldsymbol{\alpha}_b = 1, \end{array}$$

where $\mathbf{K}_a$ and $\mathbf{K}_b$ are the kernel matrices for the two representations. ■

Figure 6.4 shows the two feature spaces with the projections of 7 points. The shading corresponds to the value of the projection on the first correlation direction using a Gaussian kernel in each feature space.

Overfitting in CCA Again applying the Lagrangian techniques this leads to the equations

$$\mathbf{K}_a \mathbf{K}_b \boldsymbol{\alpha}_b - \lambda \mathbf{K}_a^2 \boldsymbol{\alpha}_a = \mathbf{0} \text{ and } \mathbf{K}_b \mathbf{K}_a \boldsymbol{\alpha}_a - \lambda \mathbf{K}_b^2 \boldsymbol{\alpha}_b = \mathbf{0}.$$

These equations highlight the potential problem of overfitting that arises in high-dimensional feature spaces. If the dimension N_a of the feature space F_a satisfies $N_a \gg \ell$, it is likely that the data will be linearly independent in the feature space. For example this is always true for a Gaussian kernel. But if the data are linearly independent in F_a the matrix $\mathbf{K}_a$ will be full rank and hence invertible. This gives

$$\boldsymbol{\alpha}_a = \frac{1}{\lambda} \mathbf{K}_a^{-1} \mathbf{K}_b \boldsymbol{\alpha}_b \tag{6.23}$$

and so

$$\mathbf{K}_b^2 \boldsymbol{\alpha}_b - \lambda^2 \mathbf{K}_b^2 \boldsymbol{\alpha}_b = \mathbf{0}.$$

This equation will hold for all vectors $\boldsymbol{\alpha}_b$ with $\lambda = 1$. Hence, we are able to find perfect correlations between arbitrary projections in F_b and an appropriate choice of the projection in F_a. Clearly these correlations are failing to distinguish spurious features from those capturing the underlying semantics. This is perhaps most clearly demonstrated if we consider a random permutation σ of the examples for the second projections to create the vectors

$$\left(\boldsymbol{\phi}_a\left(\mathbf{x}_i\right), \boldsymbol{\phi}_b\left(\mathbf{x}_{\sigma(i)}\right)\right), \; i = 1, \ldots, \ell.$$

The kernel matrix $\mathbf{K}_a$ will be unchanged and hence still invertible. We are therefore still able to find perfect correlations even though the underlying semantics are no longer correlated in the two representations.

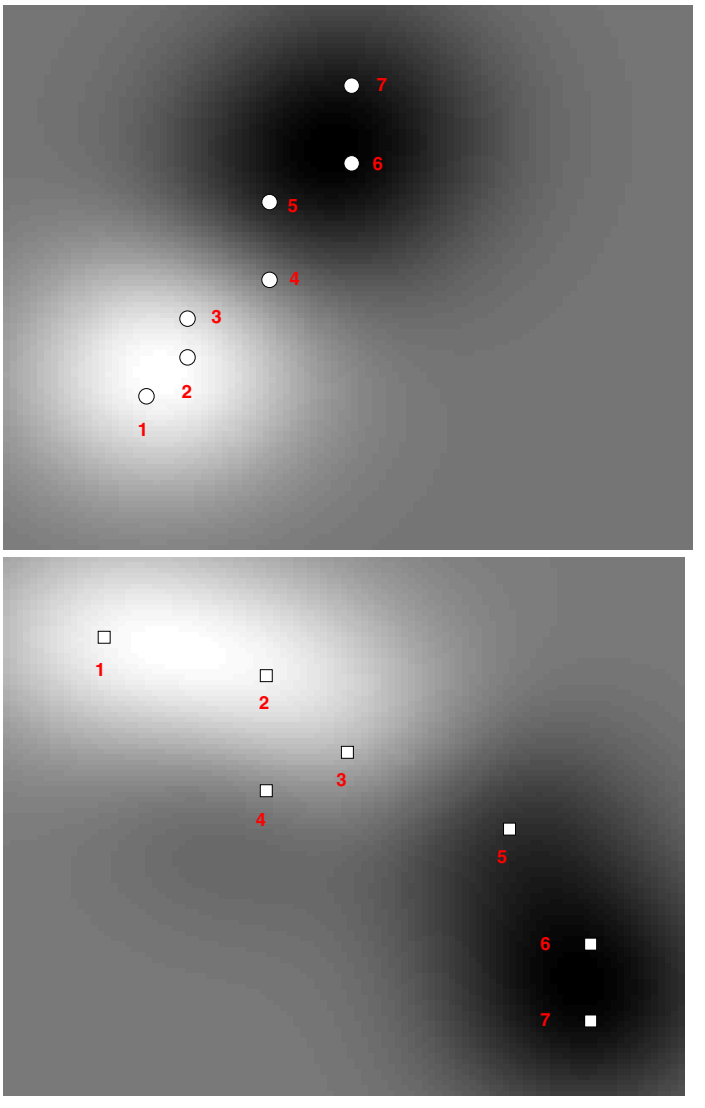

Fig. 6.4. Two feature spaces for a paired dataset with shading indicating the value of the projection onto the first correlation direction.

These observations show that the class of pattern functions we have selected are too flexible. We must introduce some regularisation to control the flexibility. We must, therefore, investigate the statistical stability of CCA, if we are to ensure that meaningful patterns are found.

Stability analysis of CCA Maximising correlation corresponds to minimising the empirical expectation of the pattern function

$$g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x}) = \left\|\mathbf{w}_a'\boldsymbol{\phi}_a(\mathbf{x}) - \mathbf{w}_b'\boldsymbol{\phi}_b(\mathbf{x})\right\|^2,$$

subject to the same conditions, since

$$\begin{aligned}\hat{\mathbb{E}}\left[\left\|\mathbf{w}_a'\boldsymbol{\phi}_a(\mathbf{x}) - \mathbf{w}_b'\boldsymbol{\phi}_b(\mathbf{x})\right\|^2\right] &= \hat{\mathbb{E}}\left[\left\|\mathbf{w}_a'\boldsymbol{\phi}_a(\mathbf{x})\right\|^2\right] + \hat{\mathbb{E}}\left[\left\|\mathbf{w}_b'\boldsymbol{\phi}_b(\mathbf{x})\right\|^2\right] - \\ &\qquad 2\hat{\mathbb{E}}\left[\left\langle\mathbf{w}_a'\boldsymbol{\phi}_a(\mathbf{x}), \mathbf{w}_b'\boldsymbol{\phi}_b(\mathbf{x})\right\rangle\right] \\ &= 2\left(1 - \mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b\right).\end{aligned}$$

The function $g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x}) \approx 0$ captures the property of the pattern that we are seeking. It assures us that the feature $\mathbf{w}_a'\phi_a(\mathbf{x})$ that can be obtained from one view of the data is almost identical to $\mathbf{w}_b'\phi_b(\mathbf{x})$ computable from the second view. Such pairs of features are therefore able to capture underlying properties of the data that are present in both views. If our assumption is correct, that what is essential is common to both views, then these features must be capturing some important properties. We can obtain a stability analysis of the function by simply viewing $g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x})$ as a regression function, albeit with special structure, attempting to learn the constant 0 function. Applying the standard Rademacher bound, observe that the empirical expected value of $g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x})$ is simply $2(1 - \mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b)$. Furthermore, we can use the same technique as that described in Theorem A.3 of Appendix A.2 to represent the function as a linear function in the feature space determined by the quadratic kernel

$$\hat{\kappa}(\mathbf{x},\mathbf{z}) = \left(\kappa_a(\mathbf{x},\mathbf{z}) + \kappa_a(\mathbf{x},\mathbf{z})\right)^2,$$

with norm-squared

$$2\left\|\mathbf{w}_a\mathbf{w}_b'\right\|_F^2 = 2\,\mathrm{tr}\left(\mathbf{w}_b\mathbf{w}_a'\mathbf{w}_a\mathbf{w}_b'\right) = \|\mathbf{w}_a\|^2\|\mathbf{w}_b\|^2.$$

This gives the following theorem.

Theorem 6.33 *Fix A and B in $\mathbb{R}^+$. If we obtain a feature given by the pattern function $g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x})$ with $\|\mathbf{w}_a\| \leq A$ and $\|\mathbf{w}_b\| \leq B$, on a paired training set S of size ℓ in the feature space defined by the kernels κ_a and κ_b drawn i.i.d. according to a distribution $\mathcal{D}$, then with probability greater than $1-\delta$ over the generation of S, the expected value of $g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x})$ on new data is bounded by*

$$\mathbb{E}_{\mathcal{D}}\left[g_{\mathbf{w}_a,\mathbf{w}_b}(\mathbf{x})\right] \leq 2\left(1 - \mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b\right) +$$

$$AB\sqrt{\frac{2}{\ell}\sum_{i=1}^{\ell}\left(\kappa_a(\mathbf{x}_i,\mathbf{x}_i)+\kappa_b(\mathbf{x}_i,\mathbf{x}_i)\right)^2}+3R^2\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

where

$$R^2=\max_{\mathbf{x}\in\,\mathrm{supp}(\mathcal{D})}\left(\kappa_a(\mathbf{x},\mathbf{x})+\kappa_b(\mathbf{x},\mathbf{x})\right).$$

The theorem indicates that the empirical value of the pattern function will be close to its expectation, provided that the norms of the two direction vectors are controlled. Hence, we must trade-off between finding good correlations while not allowing the norms to become too large.

Regularisation of CCA Theorem 6.33 shows that the quality of the generalisation of the associated pattern function is controlled by the product of the norms of the weight vectors $\mathbf{w}_a$ and $\mathbf{w}_b$. We therefore introduce a penalty on the norms of these weight vectors. This gives rise to the primal optimisation problem.

Computation 6.34 [Regularised CCA] The regularised version of CCA is solved by the optimisation:

$$\begin{aligned}&\max_{\mathbf{w}_a,\mathbf{w}_b}\rho\left(\mathbf{w}_a,\mathbf{w}_b\right) \qquad (6.24)\\ =\;&\frac{\mathbf{w}_a'\mathbf{C}_{ab}\mathbf{w}_b}{\sqrt{\left((1-\tau_a)\,\mathbf{w}_a'\mathbf{C}_{aa}\mathbf{w}_a+\tau_a\left\|\mathbf{w}_a\right\|^2\right)\left((1-\tau_b)\,\mathbf{w}_b'\mathbf{C}_{bb}\mathbf{w}_b+\tau_b\left\|\mathbf{w}_b\right\|^2\right)}},\end{aligned}$$

where the two regularisation parameters τ_a and τ_b control the flexibility in the two feature spaces. ■

Notice that τ_a, τ_b interpolate smoothly between the maximisation of the correlation and the maximisation of the covariance described in Section 6.3. Dualising we arrive at the following optimisation problem.

Computation 6.35 [Kernel regularised CCA] The dual regularised CCA is solved by the optimisation

$$\begin{array}{ll}\max_{\boldsymbol{\alpha}_a,\boldsymbol{\alpha}_b} & \boldsymbol{\alpha}_a'\mathbf{K}_a\mathbf{K}_b\boldsymbol{\alpha}_b\\ \text{subject to} & (1-\tau_a)\,\boldsymbol{\alpha}_a'\mathbf{K}_a^2\boldsymbol{\alpha}_a+\tau_a\boldsymbol{\alpha}_a'\mathbf{K}_a\boldsymbol{\alpha}_a=1\\ & \text{and } (1-\tau_b)\,\boldsymbol{\alpha}_b'\mathbf{K}_b^2\boldsymbol{\alpha}_b+\tau_b\boldsymbol{\alpha}_b'\mathbf{K}_b\boldsymbol{\alpha}_b=1.\end{array}$$

■

Note that as with ridge regression we regularised by penalising the norms of the weight vectors. Nonetheless, the resulting form of the equations obtained does not in this case correspond to a simple addition to the diagonal of the kernel matrix, the so-called ridge of ridge regression.

Solving dual regularised CCA Using the Lagrangian technique, we can now obtain the equations

$$\begin{aligned} \mathbf{K}_a\mathbf{K}_b\boldsymbol{\alpha}_b - \lambda\left(1-\tau_a\right)\mathbf{K}_a^2\boldsymbol{\alpha}_a - \lambda\tau_a\mathbf{K}_a\boldsymbol{\alpha}_a &= \mathbf{0} \\ \text{and } \mathbf{K}_b\mathbf{K}_a\boldsymbol{\alpha}_a - \lambda\left(1-\tau_b\right)\mathbf{K}_b^2\boldsymbol{\alpha}_b - \lambda\tau_b\mathbf{K}_b\boldsymbol{\alpha}_b &= \mathbf{0}, \end{aligned}$$

hence forming the generalised eigenvalue problem

$$\begin{pmatrix} \mathbf{0} & \mathbf{K}_a\mathbf{K}_b \\ \mathbf{K}_b\mathbf{K}_a & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{pmatrix} = \lambda \begin{pmatrix} \left(1-\tau_a\right)\mathbf{K}_a^2 + \tau_a\mathbf{K}_a & \mathbf{0} \\ \mathbf{0} & \left(1-\tau_b\right)\mathbf{K}_b^2 + \tau_b\mathbf{K}_b \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}_a \\ \boldsymbol{\alpha}_b \end{pmatrix}.$$

One difficulty with this approach can be the size of the resulting generalised eigenvalue problem, since it will be twice the size of the training set. A method of tackling this is to use the partial Gram–Schmidt orthonormalisation of the data in the feature space to form a lower-dimensional approximation to the feature representation of the data. As described in Section 5.2 this is equivalent to performing an incomplete Cholesky decomposition of the kernel matrices

$$\mathbf{K}_a = \mathbf{R}_a'\mathbf{R}_a \text{ and } \mathbf{K}_b = \mathbf{R}_b'\mathbf{R}_b,$$

with the columns of $\mathbf{R}_a$ and $\mathbf{R}_b$ being the new feature vectors of the training points in the orthonormal basis created by the Gram–Schmidt process. Performing an incomplete Cholesky decomposition ensures that $\mathbf{R}_a \in \mathbb{R}^{n_a\times\ell}$ has linearly independent rows so that $\mathbf{R}_a\mathbf{R}_a'$ is invertible. The same holds for $\mathbf{R}_b\mathbf{R}_b'$ with $\mathbf{R}_b \in \mathbb{R}^{n_b\times\ell}$.

We can now view our problem as a primal canonical correlation analysis with the feature vectors given by the columns of $\mathbf{R}_a$ and $\mathbf{R}_b$. This leads to the equations

$$\begin{aligned} \mathbf{R}_a\mathbf{R}_b'\mathbf{w}_b - \lambda\left(1-\tau_a\right)\mathbf{R}_a\mathbf{R}_a'\mathbf{w}_a - \lambda\tau_a\mathbf{w}_a &= \mathbf{0} \\ \text{and } \mathbf{R}_b\mathbf{R}_a'\mathbf{w}_a - \lambda\left(1-\tau_b\right)\mathbf{R}_b\mathbf{R}_b'\mathbf{w}_b - \lambda\tau_b\mathbf{w}_b &= \mathbf{0}. \end{aligned} \tag{6.25}$$

From the first equation, we can now express $\mathbf{w}_a$ as

$$\mathbf{w}_a = \frac{1}{\lambda}\left(\left(1-\tau_a\right)\mathbf{R}_a\mathbf{R}_a' + \tau_a\mathbf{I}\right)^{-1}\mathbf{R}_a\mathbf{R}_b'\mathbf{w}_b,$$

which on substitution in the second gives the normal (albeit non-symmetric) eigenvalue problem

$$\left((1-\tau_b)\,\mathbf{R}_b\mathbf{R}_b' + \tau_b\mathbf{I}\right)^{-1}\mathbf{R}_b\mathbf{R}_a'\left((1-\tau_a)\,\mathbf{R}_a\mathbf{R}_a' + \tau_a\mathbf{I}\right)^{-1}\mathbf{R}_a\mathbf{R}_b'\mathbf{w}_b = \lambda^2\mathbf{w}_b$$

of dimension $n_b \times n_b$. After performing a full Cholesky decomposition

$$\mathbf{R}'\mathbf{R} = \left((1-\tau_b)\,\mathbf{R}_b\mathbf{R}_b' + \tau_b\mathbf{I}\right)$$

of the non-singular matrix on the right hand side, we then take

$$\mathbf{u}_b = \mathbf{R}\mathbf{w}_b,$$

which using the fact that the transpose and inversion operations commute leads to the equivalent symmetric eigenvalue problem

$$\left(\mathbf{R}'\right)^{-1}\mathbf{R}_b\mathbf{R}_a'\left((1-\tau_a)\,\mathbf{R}_a\mathbf{R}_a' + \tau_a\mathbf{I}\right)^{-1}\mathbf{R}_a\mathbf{R}_b'\mathbf{R}^{-1}\mathbf{u}_b = \lambda^2\mathbf{u}_b.$$

By symmetry we could have created an eigenvalue problem of dimension $n_a \times n_a$. Hence, the size of the eigenvalue problem can be reduced to the smaller of the two partial Gram–Schmidt dimensions.

We can of course recover the full unapproximated kernel canonical correlation analysis if we simply choose $n_a = \operatorname{rank}(\mathbf{K}_a)$ and $n_b = \operatorname{rank}(\mathbf{K}_b)$. Even in this case we have avoided the need to solve a generalised eigenvalue problem, while at the same time reducing the dimension of the problem by at least a factor of two since $\min(n_a, n_b) \leq \ell$. The overall algorithm is as follows.

Algorithm 6.36 [Kernel CCA] Kernel canonical correlation analysis can be solved as shown in Code Fragment 6.3. ∎

This means that we can have two views of an object that together create a paired dataset S through two different representations or kernels. We use this procedure to compute correlations between the two sets that are stable in the sense that they capture properties of the underlying distribution rather than of the particular training set or view.

Remark 6.37 [Bilingual corpora] Example 6.28 has already mentioned as examples of paired datasets so-called parallel corpora in which each document appears with its translation to a second language. We can apply the kernel canonical correlation analysis to such a corpus using kernels for text that will be discussed in Chapter 10. This will provide a means of projecting documents from either language into a common semantic space. ∎

Input	kernel matrices $\mathbf{K}_a$ and $\mathbf{K}_b$ with parameters τ_a and τ_b
Process	Perform (incomplete) Cholesky decompositions: $\mathbf{K}_a = \mathbf{R}'_a\mathbf{R}_a$ and $\mathbf{K}_b = \mathbf{R}'_b\mathbf{R}_b$ of dimensions n_a and n_b; perform a complete Cholesky decomposition: $(1-\tau_b)\,\mathbf{R}_b\mathbf{R}'_b + \tau_b\mathbf{I} = \mathbf{R}'\mathbf{R}$ solve the eigenvalue problem: $(\mathbf{R}')^{-1}\,\mathbf{R}_b\mathbf{R}'_a\left((1-\tau_a)\,\mathbf{R}_a\mathbf{R}'_a + \tau_a\mathbf{I}\right)^{-1}\mathbf{R}_a\mathbf{R}'_b\mathbf{R}^{-1}\mathbf{u}_b = \lambda^2\mathbf{u}_b$. to give each λ_j, $\mathbf{u}_b^j$ compute $\mathbf{w}_b^j = \mathbf{R}^{-1}\mathbf{u}_b$, $\mathbf{w}_b^j = \mathbf{w}_b^j/\|\mathbf{w}_b^j\|$ $\mathbf{w}_a^j = \frac{1}{\lambda_j}\left((1-\tau_a)\,\mathbf{R}_a\mathbf{R}'_a + \tau_a\mathbf{I}\right)^{-1}\mathbf{R}_a\mathbf{R}'_b\mathbf{w}_b^j$ $\mathbf{w}_a^j = \mathbf{w}_a^j/\|\mathbf{w}_a^j\|$
Output	eigenvectors and values $\mathbf{w}_a^j$, $\mathbf{w}_b^j$ and $\lambda_j > 0$,. $j = 1, \ldots, \min(n_a, n_b)$

Code Fragment 6.3. Pseudocode for the kernel CCA algorithm.

Remark 6.38 [More than 2 representations] Notice that a simple manipulation of equation (6.22) gives the alternative formulation

$$\begin{pmatrix} C_{aa} & C_{ab} \\ C_{ba} & C_{bb} \end{pmatrix}\begin{pmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{pmatrix} = (1+\lambda)\begin{pmatrix} C_{aa} & 0 \\ 0 & C_{bb} \end{pmatrix}\begin{pmatrix} \mathbf{w}_a \\ \mathbf{w}_b \end{pmatrix}$$

which suggests a natural generalisation, namely seeking correlations between three or more views. Given k multivariate random variables, it reduces to the generalised eigenvalue problem

$$\begin{aligned} &\begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & \cdots & \mathbf{C}_{1k} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{k1} & \cdots & \cdots & \mathbf{C}_{kk} \end{pmatrix}\begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \vdots \\ \mathbf{w}_k \end{pmatrix} \\ = \quad &\rho\begin{pmatrix} \mathbf{C}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_{kk} \end{pmatrix}\begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \vdots \\ \mathbf{w}_k \end{pmatrix}, \end{aligned}$$

where we use C_{ij} to denote the covariance matrix between the ith and jth views. Note that for $k > 2$ there is no obvious way of reducing such a generalised eigenvalue problem to a lower-dimensional eigenvalue problem as was possible using the Cholesky decomposition in the case $k = 2$. ∎

6.6 Fisher discriminant analysis II

We considered the Fisher discriminant in Section 5.4, arriving at a dual formulation that could be solved by solving a set of linear equations. We revisit it here to highlight the fact that it can also be viewed as the solution of a generalised eigenvalue problem and so is closely related to the correlation and covariance analysis we have been studying in this chapter. Recall that Computation 5.14 characterised the regularised Fisher discriminant as choosing its discriminant vector to maximise the quotient

$$\frac{(\boldsymbol{\mu}_{\mathbf{w}}^{+} - \boldsymbol{\mu}_{\mathbf{w}}^{-})^2}{\left(\sigma_{\mathbf{w}}^{+}\right)^2 + \left(\sigma_{\mathbf{w}}^{-}\right)^2 + \lambda \|\mathbf{w}\|^2}.$$

This can be expressed using the notation of Section 5.4 as

$$\max_{\mathbf{w}} \frac{\mathbf{w}'\mathbf{X}'\mathbf{y}\mathbf{y}'\mathbf{X}\mathbf{w}}{\lambda \mathbf{w}'\mathbf{w} + \frac{\ell}{2\ell^+\ell^-}\mathbf{w}'\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{w}} = \max_{\mathbf{w}} \frac{\mathbf{w}'\mathbf{E}\mathbf{w}}{\mathbf{w}'\mathbf{F}\mathbf{w}},$$

where

$$\mathbf{E} = \mathbf{X}'\mathbf{y}\mathbf{y}'\mathbf{X} \text{ and } \mathbf{F} = \lambda \mathbf{I} + \frac{\ell}{2\ell^+\ell^-}\mathbf{X}'\mathbf{B}\mathbf{X}.$$

Hence, the solution is the eigenvector corresponding to the largest eigenvalue of the generalised eigenvalue problem

$$\mathbf{E}\mathbf{w} = \mu \mathbf{F}\mathbf{w},$$

as outlined in Section 6.4. Note that the matrix $\mathbf{E}$ has rank 1 since it can be decomposed as

$$\mathbf{E} = \left(\mathbf{X}'\mathbf{y}\right)\left(\mathbf{y}'\mathbf{X}\right),$$

where $\mathbf{X}'\mathbf{y}$ has just one column. This implies that only the first eigenvector contains useful information and that it can be found by the matrix inversion procedure described in Section 5.4.

6.7 Methods for linear regression

The previous section showed how the Fisher discriminant is equivalent to choosing a feature by solving a generalised eigenvalue problem and then defining a threshold in that one-dimensional space. This section will return to the problem of regression and consider how the feature spaces derived from solving eigenvalue problems might be used to enhance regression accuracy.

We first met regression in Chapter 2 when we considered simple linear

regression subsequently augmented with a regularisation of the regression vector $\mathbf{w}$ to create so-called ridge regression defined in Computation 7.21. In this section we consider performing linear regression using a new set of coordinates that has been extracted from the data with the methods presented above. This will lead to an easier understanding of some popular regression algorithms.

First recall the optimisation of least squares regression. We seek a vector $\mathbf{w}$ that solves

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2,$$

where as usual the rows of $\mathbf{X}$ contain the feature vectors of the examples and the desired outputs are stored in the vector $\mathbf{y}$. If we wish to consider a more general multivariate regression both $\mathbf{w}$ and $\mathbf{y}$ become matrices $\mathbf{W}$ and $\mathbf{Y}$ and the norm is taken as the Frobenius matrix norm

$$\min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_F^2,$$

since this is equivalent to summing the squared norms of the individual errors.

Principal components regression Perhaps the simplest method to consider is the use of the features returned by PCA. If we were to use the first k eigenvectors of $\mathbf{X}'\mathbf{X}$ as our features and leave $\mathbf{Y}$ unchanged, this would correspond to performing PCA and regressing in the feature space given by the first k principal axes, so the data matrix now becomes $\mathbf{X}\mathbf{U}_k$, where $\mathbf{U}_k$ contains the first k columns of the matrix $\mathbf{U}$ from the singular value decomposition $\mathbf{X}' = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'$. Using the fact that premultiplying by an orthogonal matrix does not affect the norm, we obtain

$$\begin{aligned}\min_{\mathbf{B}} \|\mathbf{X}\mathbf{U}_k\mathbf{B} - \mathbf{Y}\|_F^2 &= \min_{\mathbf{B}} \left\|\mathbf{V}'\mathbf{V}\mathbf{\Sigma}'\mathbf{U}'\mathbf{U}_k\mathbf{B} - \mathbf{V}'\mathbf{Y}\right\|_F^2 \\ &= \min_{\mathbf{B}} \left\|\mathbf{\Sigma}'_k\mathbf{B} - \mathbf{V}'\mathbf{Y}\right\|_F^2,\end{aligned}$$

where $\mathbf{\Sigma}_k$ is the matrix obtained by taking the first k rows of $\mathbf{\Sigma}$. Letting $\mathbf{\Sigma}_k^{-1}$ denote the matrix obtained from $\mathbf{\Sigma}_k$ by inverting its diagonal elements, we have $\mathbf{\Sigma}_k^{-1}\mathbf{\Sigma}'_k = \mathbf{I}_k$, so the solution $\mathbf{B}$ with minimal norm is given by

$$\mathbf{B} = \mathbf{\Sigma}_k^{-1}\mathbf{V}'\mathbf{Y} = \bar{\mathbf{\Sigma}}_k^{-1}\mathbf{V}'_k\mathbf{Y},$$

where $\mathbf{V}_k$ contains the first k columns of $\mathbf{V}$ and $\bar{\mathbf{\Sigma}}_k^{-1}$ is the square matrix containing the first k columns of $\mathbf{\Sigma}_k^{-1}$. It follows from the singular value decomposition that

$$\mathbf{V}'_k = \bar{\mathbf{\Sigma}}_k^{-1}\mathbf{U}'_k\mathbf{X}', \tag{6.26}$$

so we can also write

$$\mathbf{B} = \bar{\mathbf{\Sigma}}_k^{-2}\mathbf{U}_k'\mathbf{X}'\mathbf{Y}.$$

This gives the primal form emphasising that the components are computed by an inner product between the corresponding feature vectors $\mathbf{u}_j$ that form the columns of $\mathbf{U}$ and the data matrix $\mathbf{X}'\mathbf{Y}$ weighted by the inverse of the corresponding eigenvalue.

If we recall that $\mathbf{V}$ contains the eigenvectors $\mathbf{v}_j$ of the kernel matrix and that kernel PCA identifies the dual variables of the directions $\mathbf{u}_j$ as

$$\frac{1}{\sigma_j}\mathbf{v}_j,$$

it follows from equation (6.26) that the regression coefficient for the jth principal component is given by the inner product between its dual representation and the target outputs again with an appropriate weighting of the inverse of the corresponding singular value. We can therefore write the resulting regression function for the univariate case in the dual form as

$$f(\mathbf{x}) = \sum_{j=1}^{k}\frac{1}{\sigma_j}\sum_{s=1}^{\ell}\mathbf{v}_{js}y_s\sum_{i=1}^{\ell}\frac{1}{\sigma_j}\mathbf{v}_{ji}\kappa(\mathbf{x}_i,\mathbf{x}),$$

where $\mathbf{v}_{js}$ denotes the sth component of the jth eigenvector $\mathbf{v}_j$. Hence

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

where

$$\boldsymbol{\alpha} = \sum_{j=1}^{k}\frac{1}{\lambda_j}\left(\mathbf{v}_j'\mathbf{y}\right)\mathbf{v}_j.$$

The form of the solution has an intuitive feel in that we work out the covariances with the target values of the different eigenvectors and weight their contribution to $\boldsymbol{\alpha}$ proportionately. This also implies that we can continue to add additional dimensions without recomputing the previous coefficients in the primal space but by simply adding in a vector to $\boldsymbol{\alpha}$ in the dual resprresentation. This is summarised in Algorithm 6.39.

Algorithm 6.39 [Principal components regression] The dual principal components regression (PCR) algorithm is given in Code Fragment 6.4. ■

input	Data $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, dimension k and target output vectors $\mathbf{y}^s$, $s = 1, \ldots, m$.
process	$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, \ldots, \ell$ $\mathbf{K} = \mathbf{K} - \frac{1}{\ell}\mathbf{j}\mathbf{j}'\mathbf{K} - \frac{1}{\ell}\mathbf{K}\mathbf{j}\mathbf{j}' + \frac{1}{\ell^2}(\mathbf{j}'\mathbf{K}\mathbf{j})\mathbf{j}\mathbf{j}'$, $[\mathbf{V}, \mathbf{\Lambda}] = \text{eig}(\mathbf{K})$ $\boldsymbol{\alpha}^s = \sum_{j=1}^{k} \frac{1}{\lambda_j}(\mathbf{v}_j'\mathbf{y}^s)\mathbf{v}_j$, $s = 1, \ldots, m$.
output	Regression functions $f_s(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i^s \kappa(\mathbf{x}_i, \mathbf{x})$, $s = 1, \ldots, m$.

Code Fragment 6.4. Pseudocode for dual principal components regression.

Regression features from maximal covariance We can see from the previous example that the critical measure for the different coordinates is their covariance with the matrix $\mathbf{X}'\mathbf{Y}$, since the regression coefficient is proportional to this quantity. This suggests that rather than using PCA to choose the features, we should select directions that maximise the covariance. Proposition 6.20 showed that the directions that maximise the covariance are given by the singular vectors of the matrix $\mathbf{X}'\mathbf{Y}$. Furthermore, the characterisation of the minimisers of equation (6.15) as the orthogonal matrices of the singular value decomposition of $\mathbf{X}'\mathbf{Y}$ suggests that they may provide a useful set of features when solving a regression problem from an input space $X = \mathbb{R}^n$ to an output space $Y = \mathbb{R}^m$. There is an implicit restriction as there are only m non-zero singular values of the matrix $\mathbf{X}'\mathbf{Y}$. We must therefore consider performing regression of the variables $\mathbf{Y}$ in terms of $\mathbf{X}\mathbf{U}_k$, where $\mathbf{U}_k$ is the matrix formed of the first $k \leq m$ columns of $\mathbf{U}$. We seek a $k \times m$ matrix of coefficients $\mathbf{B}$ that solves the optimisation

$$\begin{aligned}
\min_{\mathbf{B}} \|\mathbf{X}\mathbf{U}_k\mathbf{B} - \mathbf{Y}\|_F^2 &= \min_{\mathbf{B}} \langle \mathbf{X}\mathbf{U}_k\mathbf{B} - \mathbf{Y}, \mathbf{X}\mathbf{U}_k\mathbf{B} - \mathbf{Y} \rangle_F \\
&= \min_{\mathbf{B}} \big(\text{tr}(\mathbf{B}'\mathbf{U}_k'\mathbf{X}'\mathbf{X}\mathbf{U}_k\mathbf{B}) - 2\,\text{tr}(\mathbf{B}'\mathbf{U}_k'\mathbf{X}'\mathbf{Y}) \\
&\qquad + \text{tr}(\mathbf{Y}'\mathbf{Y}) \big) \\
&= \min_{\mathbf{B}} \big(\text{tr}(\mathbf{B}'\mathbf{U}_k'\mathbf{X}'\mathbf{X}\mathbf{U}_k\mathbf{B}) - 2\,\text{tr}(\mathbf{B}'\mathbf{U}_k'\mathbf{X}'\mathbf{Y}) \big) .
\end{aligned}$$

The final regression coefficients are given by $\mathbf{U}_k\mathbf{B}$. We seek the minimum by computing the gradient with respect to $\mathbf{B}$ and setting to zero. This results in the equation

$$\mathbf{U}_k'\mathbf{X}'\mathbf{X}\mathbf{U}_k\mathbf{B} = \mathbf{U}_k'\mathbf{X}'\mathbf{Y} = \mathbf{U}_k'\mathbf{U}\mathbf{\Sigma}\mathbf{V}' = \mathbf{\Sigma}_k\mathbf{V}_k'.$$

The solution for $\mathbf{B}$ can be computed using, for example, a Cholesky decomposition of $\mathbf{U}_k'\mathbf{X}'\mathbf{X}\mathbf{U}_k$, though for the case where $k = 1$, it is given by

$$\mathbf{B} = \frac{\sigma_1}{\mathbf{u}_1'\mathbf{X}'\mathbf{X}\mathbf{u}_1}\mathbf{v}_1'.$$

If we wish to compute the dual representation of this regression coefficient, we must express

$$\mathbf{u}_1\mathbf{B} = \frac{\sigma_1}{\mathbf{u}_1'\mathbf{X}'\mathbf{X}\mathbf{u}_1}\mathbf{u}_1\mathbf{v}_1' = \mathbf{X}'\boldsymbol{\alpha},$$

for some $\boldsymbol{\alpha}$. By observing that $\mathbf{u}_1 = \frac{1}{\sigma_1}\mathbf{X}'\mathbf{Y}\mathbf{v}_1$ we obtain

$$\boldsymbol{\alpha} = \frac{1}{\mathbf{u}_1'\mathbf{X}'\mathbf{X}\mathbf{u}_1}\mathbf{Y}\mathbf{v}_1\mathbf{v}_1'.$$

Note that the $\frac{1}{\sigma_1}\mathbf{Y}\mathbf{v}_1$ are the dual variables of $\mathbf{u}_1$, so that we again see the dual variables of the feature playing a role in determining the dual representation of the regression coefficients. For $k > 1$, there is no avoiding solving a system of linear equations.

When we compare PCR and the use of maximal covariance features, PCR has two advantages. Firstly, the coefficients can be obtained by simple inner products rather than solving linear equations, and secondly, the restriction to take $k \leq m$ does not apply. The disadvantage of PCR is that the choice of features does not take into account the output vectors $\mathbf{Y}$ so that the features are unable to align with the maximal covariance directions. As discussed above the features that carry the regression information may be of relatively low variance and so could potentially be removed by the PCA phase of the algorithm.

The next section will describe an algorithm known as *partial least squares* that combines the advantages of both methods while further improving the covariance obtained and providing a simple method for iteratively computing the feature directions.

6.7.1 Partial least squares

When developing a regression algorithm, it appears that it may not be the variance of the inputs, but their covariance with the target that is more important. The partial least squares approach uses the covariance to guide the selection of features before performing least-squares regression in the derived feature space. It is very popular in the field of chemometrics, where high-dimensional and correlated representations are commonplace. This situation will also arise if we use kernels to project the data into spaces where

the new coordinates are far from uncorrelated and where the dimension of the space is much higher than the sample size. The combination of PLS with kernels produces a powerful algorithm that we will describe in the next subsection after first deriving the primal version here.

Our first goal is to find the directions of maximum covariance. Since we have already described in Section 6.3 that these are computed by the singular value decomposition of $\mathbf{X}'\mathbf{Y}$ and have further discussed the difficulties of using the resulting features at the end of the previous section, it seems a contradiction that we should be able to further improve the covariance. This is certainly true of the first direction and indeed the first direction that is chosen by the partial least squares algorithm is that given by the singular vector corresponding to the largest singular value. Consider now performing regression using only this first direction. The regression coefficient is the one for the case $k = 1$ given in the previous subsection as $b\mathbf{v}_1'$, where

$$b = \frac{\sigma_1}{\mathbf{u}_1'\mathbf{X}'\mathbf{X}\mathbf{u}_1},$$

while the approximation of $\mathbf{Y}$ will be given by

$$b\mathbf{X}\mathbf{u}_1\mathbf{v}_1'.$$

Hence, the values across the training set of the hidden feature that has been used are given in the vector $\mathbf{X}\mathbf{u}_1$. This suggests that rather than deflate $\mathbf{X}'\mathbf{Y}$ by $\sigma_1\mathbf{u}_1\mathbf{v}_1'$ as required for the singular value decomposition, we deflate $\mathbf{X}$ by projecting its columns into the space orthogonal to $\mathbf{X}\mathbf{u}_1$. Using equation (5.8) which gives the projection matrix for a normalised vector $\mathbf{w}$ as

$$\left(\mathbf{I} - \mathbf{w}\mathbf{w}'\right),$$

we obtain the deflation of $\mathbf{X} = \mathbf{X}_1$ as

$$\mathbf{X}_2 = \left(\mathbf{I} - \frac{\mathbf{X}_1\mathbf{u}_1\mathbf{u}_1'\mathbf{X}_1'}{\mathbf{u}_1'\mathbf{X}_1'\mathbf{X}_1\mathbf{u}_1}\right)\mathbf{X}_1 = \mathbf{X}_1 - \frac{\mathbf{X}_1\mathbf{u}_1\mathbf{u}_1'\mathbf{X}_1'\mathbf{X}_1}{\mathbf{u}_1'\mathbf{X}_1'\mathbf{X}_1\mathbf{u}_1} = \mathbf{X}_1\left(\mathbf{I} - \frac{\mathbf{u}_1\mathbf{u}_1'\mathbf{X}_1'\mathbf{X}_1}{\mathbf{u}_1'\mathbf{X}_1'\mathbf{X}_1\mathbf{u}_1}\right). \tag{6.27}$$

If we now recursively choose a new direction, the result will be that the vector of values of the next hidden feature will necessarily be orthogonal to $\mathbf{X}\mathbf{u}_1$ since it will be a linear combination of the columns of the deflated matrix all of which are othogonal to that vector.

Remark 6.40 [Conjugacy] It is important to distinguish between the orthogonality between the values of a feature across the training examples, and the orthogonality of the feature vectors. Vectors that satisfy the orthogonality considered here are referred to as *conjugate*. Furthermore, this

will imply that the coefficients can be computed iteratively at each stage since there can be no interaction between a set of conjugate features. ■

Remark 6.41 [Conjugacy of eigenvectors] It may seem surprising that deflating using $\mathbf{X}\mathbf{u}_1$ leads to orthogonal features when, for an eigenvalue decomposition, we deflate by the equivalent of $\mathbf{u}_1$; that is, the first eigenvector. The reason that the eigenvalue deflation leads to conjugate features is that for the eigenvalue case $\mathbf{X}\mathbf{u}_1 = \sigma_1\mathbf{v}_1$ is the first eigenvector of the kernel matrix. Hence, using the eigenvectors results in features that are automatically conjugate. ■

Since we have removed precisely the direction that contributed to the maximal covariance, namely $\mathbf{X}\mathbf{u}_1$, the maximal covariance of the deflated matrix must be at least as large as σ_2, the second singular value of the original matrix. In general, the covariance of the deflated matrix will be larger than σ_2. Furthermore, this also means that the restriction to $k \leq m$ no longer applies since we do not need to deflate $\mathbf{Y}$ at all. We summarise the PLS feature extraction in Algorithm 6.42.

Algorithm 6.42 [PLS feature extraction] The PLS feature extraction algorithm is given in Code Fragment 6.5. ■

input	Data matrix $\mathbf{X} \in \mathbb{R}^{\ell \times N}$, dimension k, target vectors $\mathbf{Y} \in \mathbb{R}^{\ell \times m}$.
process	$\mathbf{X}_1 = \mathbf{X}$ for $j = 1, \ldots, k$ let $\mathbf{u}_j, \mathbf{v}_j, \sigma_j$ be the first singular vector/value of $\mathbf{X}'_j\mathbf{Y}$, $\mathbf{X}_{j+1} = \left(\mathbf{I} - \frac{\mathbf{X}_j\mathbf{u}_j\mathbf{u}'_j\mathbf{X}'_j}{\mathbf{u}'_j\mathbf{X}'_j\mathbf{X}_j\mathbf{u}_j}\right)\mathbf{X}_j$ end
output	Feature directions $\mathbf{u}_j$, $j = 1, \ldots, k$.

Code Fragment 6.5. Pseudocode for PLS feature extraction.

Remark 6.43 [Deflating $\mathbf{Y}$] We can if we wish use a similar deflation strategy for $\mathbf{Y}$ giving, for example

$$\mathbf{Y}_2 = \left(\mathbf{I} - \frac{\mathbf{X}_1\mathbf{u}_1\mathbf{u}'_1\mathbf{X}'_1}{\mathbf{u}'_1\mathbf{X}'_1\mathbf{X}_1\mathbf{u}_1}\right)\mathbf{Y}.$$

Surprisingly even if we do, the fact that we are only removing the explained covariance means it will have no effect on the extraction of subsequent features. An alternative way of seeing this is that we are projecting into the

space spanned by the columns of $\mathbf{X}_2$ and so are only removing components parallel to $\mathbf{X}_1\mathbf{u}_1$. This also ensures that we can continue to extract hidden features as long as there continues to be explainable variance in $\mathbf{Y}$, typically for values of $k > m$. Deflating $\mathbf{Y}$ will, however, be needed for dual partial least squares. ■

Remark 6.44 [Relation to Gram–Schmidt orthonormalisation] For one-dimensional outputs the PLS feature extraction can be viewed as a Gram–Schmidt orthonormalisation of the so-called Krylov space of vectors

$$\mathbf{X}'\mathbf{y}, \left(\mathbf{X}'\mathbf{X}\right)^1\mathbf{X}'\mathbf{y}, \ldots, \left(\mathbf{X}'\mathbf{X}\right)^{k-1}\mathbf{X}'\mathbf{y}$$

with respect to the inner product

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}'\left(\mathbf{X}'\mathbf{X}\right)\mathbf{b}.$$

It is also closely related to the conjugate gradient method as applied to minimising the expression

$$\frac{1}{2}\mathbf{u}'\left(\mathbf{X}'\mathbf{X}\right)\mathbf{u} - \mathbf{y}\mathbf{X}'\mathbf{u}.$$

■

Orthogonality and conjugacy of PLS features There are some nice properties of the intermediate quantities computed in the algorithm. For example the vectors $\mathbf{u}_i$ are not only conjugate but also orthogonal as vectors, as the following derivation demonstrates. Suppose $i < j$, then we can write

$$\mathbf{X}_j = \mathbf{Z}\left(\mathbf{X}_i - \frac{\mathbf{X}_i\mathbf{u}_i\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i}{\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}\right),$$

for some matrix $\mathbf{Z}$. Hence

$$\mathbf{X}_j\mathbf{u}_i = \mathbf{Z}\left(\mathbf{X}_i - \frac{\mathbf{X}_i\mathbf{u}_i\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i}{\mathbf{u}_i'\mathbf{X}_i'\mathbf{X}_i\mathbf{u}_i}\right)\mathbf{u}_i = \mathbf{0}. \tag{6.28}$$

Note that $\mathbf{u}_j$ is in the span of the rows of $\mathbf{X}_j$, that is $\mathbf{u}_j = \mathbf{X}_j'\boldsymbol{\alpha}$, for some $\boldsymbol{\alpha}$. It follows that

$$\mathbf{u}_j'\mathbf{u}_i = \boldsymbol{\alpha}'\mathbf{X}_j\mathbf{u}_i = 0.$$

Furthermore, if we let

$$\mathbf{p}_j = \frac{\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j},$$

we have $\mathbf{u}_i'\mathbf{p}_j = 0$ for $i < j$. This follows from

$$\mathbf{u}_i'\mathbf{p}_j = \frac{\mathbf{u}_i'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = 0, \tag{6.29}$$

again from equation (6.28). Furthermore, we clearly have $\mathbf{u}_j'\mathbf{p}_j = 1$. The projection of $\mathbf{X}_j$ can also now be expressed as

$$\mathbf{X}_{j+1} = \mathbf{X}_j\left(\mathbf{I} - \frac{\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right) = \mathbf{X}_j\left(\mathbf{I} - \mathbf{u}_j\mathbf{p}_j'\right). \tag{6.30}$$

Computing the regression coefficients If we consider a test point with feature vector $\boldsymbol{\phi}(\mathbf{x})$ the transformations that we perform at each step should also be applied to $\boldsymbol{\phi}_1(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})$ to create a series of feature vectors

$$\boldsymbol{\phi}_{j+1}(\mathbf{x})' = \boldsymbol{\phi}_j(\mathbf{x})'\left(\mathbf{I} - \mathbf{u}_j\mathbf{p}_j'\right).$$

This is the same operation that is performed on the rows of $\mathbf{X}_j$ in equation (6.30). We can now write

$$\boldsymbol{\phi}(\mathbf{x})' = \boldsymbol{\phi}_{k+1}(\mathbf{x})' + \sum_{j=1}^{k}\boldsymbol{\phi}_j(\mathbf{x})'\mathbf{u}_j\mathbf{p}_j'.$$

The feature vector that we need for the regression $\hat{\boldsymbol{\phi}}(\mathbf{x})$ has components

$$\hat{\boldsymbol{\phi}}(\mathbf{x}) = \left(\boldsymbol{\phi}_j(\mathbf{x})'\mathbf{u}_j\right)_{j=1}^{k},$$

since these are the projections of the residual vector at stage j onto the next feature vector $\mathbf{u}_j$. Rather than compute $\boldsymbol{\phi}_j(\mathbf{x})'$ iteratively, consider using the inner products between the original $\boldsymbol{\phi}(\mathbf{x})'$ and the feature vectors $\mathbf{u}_j$ stored as the columns of the matrix $\mathbf{U}$

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x})'\mathbf{U} &= \boldsymbol{\phi}_{k+1}(\mathbf{x})'\mathbf{U} + \sum_{j=1}^{k}\boldsymbol{\phi}_j(\mathbf{x})'\mathbf{u}_j\mathbf{p}_j'\mathbf{U} \\ &= \boldsymbol{\phi}_{k+1}(\mathbf{x})'\mathbf{U} + \hat{\boldsymbol{\phi}}(\mathbf{x})'\mathbf{P}'\mathbf{U},\end{aligned}$$

where $\mathbf{P}$ is the matrix whose columns are $\mathbf{p}_j$, $j = 1, \ldots, k$. Finally, since for $s > j$, $\left(\mathbf{I} - \mathbf{u}_s\mathbf{p}_s'\right)\mathbf{u}_j = \mathbf{u}_j$, we can write

$$\boldsymbol{\phi}_{k+1}(\mathbf{x})'\mathbf{u}_j = \boldsymbol{\phi}_k(\mathbf{x})'\left(\mathbf{I} - \mathbf{u}_k\mathbf{p}_k'\right)\mathbf{u}_j = 0, \text{ for } j = 1, \ldots, k.$$

It follows that the new feature vector can be expressed as

$$\hat{\boldsymbol{\phi}}(\mathbf{x})' = \boldsymbol{\phi}(\mathbf{x})'\mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}.$$

As observed above the regression coefficients for the jth dimension of the new feature vector is

$$\frac{\sigma_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\mathbf{v}_j',$$

where $\mathbf{v}_j$ is the complementary singular vector associated with $\mathbf{u}_j$ so that

$$\sigma_j\mathbf{v}_i = \mathbf{Y}'\mathbf{X}_i\mathbf{u}_i$$

It follows that the overall regression coefficients can be computed as

$$\mathbf{W} = \mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}', \tag{6.31}$$

where $\mathbf{C}$ is the matrix with columns

$$\mathbf{c}_j = \frac{\mathbf{Y}'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}.$$

This appears to need a matrix inversion, but equation (6.29) implies that the matrix $\mathbf{P}'\mathbf{U}$ is upper triangular with constant diagonal 1 so that the computation of

$$\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}'$$

only involves the solution of m sets of k linear equations in k unknowns with an upper triangular matrix.

Iterative computation of singular vectors The final promised ingredient of the new algorithm is an iterative method for computing the maximal singular value and associated singular vectors. The technique is known as the iterative power method and can also be used to find the largest eigenvalue of a matrix. It simply involves repeatedly multiplying a random initial vector by the matrix and then renormalising. Supposing that $\mathbf{Z}\mathbf{\Lambda}\mathbf{Z}'$ is the eigen-decomposition of a matrix $\mathbf{A}$, then the computation

$$\mathbf{A}^s\mathbf{x} = \left(\mathbf{Z}\mathbf{\Lambda}\mathbf{Z}'\right)^s\mathbf{x} = \mathbf{Z}\mathbf{\Lambda}^s\mathbf{Z}'\mathbf{x} \approx \mathbf{z}_1\lambda_1^s\mathbf{z}_1'\mathbf{x}$$

shows that the vector converges to the largest eigenvector and the renormalisation coefficient to the largest eigenvalue provided $\mathbf{z}_1'\mathbf{x} \neq 0$.

In general this is not very efficient, but in the case of low-rank matrices such as $\mathbf{C}_{xy}$ when the output dimension m is small, it proves very effective. Indeed for the case when $m = 1$ a single iteration is sufficient to find the exact solution. Hence, for solving a standard regression problem this is more efficient than performing an SVD of $\mathbf{X}'\mathbf{Y}$.

Algorithm 6.45 [Primal PLS] The primal PLS algorithm is given in Code Fragment 6.6. The repeat loop computes the first singular value by the

input	Data matrix $\mathbf{X} \in \mathbb{R}^{\ell \times N}$, dimension k, target outputs $\mathbf{Y} \in \mathbb{R}^{\ell \times m}$.
process	$\boldsymbol{\mu} = \frac{1}{\ell}\mathbf{X}'\mathbf{j}$ computes the means of components $\mathbf{X}_1 = \mathbf{X} - \mathbf{j}\boldsymbol{\mu}'$ centering the data $\hat{\mathbf{Y}} = \mathbf{0}$ for $j = 1, \ldots, k$ $\mathbf{u}_j$=first column of $\mathbf{X}_j'\mathbf{Y}$ $\mathbf{u}_j = \mathbf{u_j} / \|\mathbf{u}_j\|$ repeat $\mathbf{u}_j = \mathbf{X}_j'\mathbf{Y}\mathbf{Y}'\mathbf{X}_j\mathbf{u}_j$ $\mathbf{u}_j = \mathbf{u_j} / \|\mathbf{u}_j\|$ until convergence $\mathbf{p}_j = \frac{\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}$ $\mathbf{c}_j = \frac{\mathbf{Y}'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}$ $\hat{\mathbf{Y}} = \hat{\mathbf{Y}} + \mathbf{X}_j\mathbf{u}_j\mathbf{c}_j'$ $\mathbf{X}_{j+1} = \mathbf{X}_j\left(\mathbf{I} - \mathbf{u}_j\mathbf{p}_j'\right)$ end $\mathbf{W} = \mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}'$
output	Mean vector $\boldsymbol{\mu}$, training outputs $\hat{\mathbf{Y}}$, regression coefficients $\mathbf{W}$

Code Fragment 6.6. Pseudocode for the primal PLS algorithm.

iterative method. This results in $\mathbf{u}_j$ converging to the first right singular vector $\mathbf{Y}'\mathbf{X}_j$. Following the loop we compute $\mathbf{p}_j$ and $\mathbf{c}_j$, followed by the deflation of $\mathbf{X}_j$ given by

$$\mathbf{X} \to \mathbf{X} - \mathbf{X}\mathbf{u}_j\mathbf{p}_j'.$$

as required. We can deflate $\mathbf{Y}$ to its residual but it does not affect the correlations discovered since the deflation removes components in the space spanned by $\mathbf{X}_j\mathbf{u}_j$, to which $\mathbf{X}_{j+1}$ has now become orthogonal. From our observations above it is clear that the vectors $\mathbf{X}_j\mathbf{u_j}$ generated at each stage are orthogonal to each other.

We must now allow the algorithm to classify new data. The regression coefficients $\mathbf{W}$ are given in equation (6.31).

Code Fragment 6.7 gives Matlab code for the complete PLS algorithm in primal form. Note that it begins by centering the data since covariances are computed on centred data. ■

We would now like to show how this selection can be mimicked in the dual space.

```
% X is an ell x n matrix whose rows are the training inputs
% Y is ell x m containing the corresponding output vectors
% T gives the number of iterations to be performed
mux = mean(X); muy = mean(Y); jj = ones(size(X,1),1);
X = X - jj*mux; Y = Y - jj*muy;
for i=1:T
 YX = Y'*X;
 u(:,i) = YX(1,:)'/norm(YX(1,:));
 if size(Y,2) > 1, % only loop if dimension greater than 1
   uold = u(:,i) + 1;
   while norm(u(:,i) - uold) > 0.001,
     uold = u(:,i);
     tu = YX'*YX*u(:,i);
     u(:,i) = tu/norm(tu);
   end
 end
 t = X*u(:,i);
 c(:,i) = Y'*t/(t'*t);
 p(:,i) = X'*t/(t'*t);
 trainY = trainY + t*c(:,i)';
 trainerror = norm(Y - trainY,'fro')/sqrt(ell)
 X = X - t*p(:,i)';
 % compute residual Y = Y - t*c(:,i)';
end
% Regression coefficients for new data
W = u * ((p'*u)\c');
%  Xtest gives new data inputs as rows, Ytest true outputs
elltest = size(Xtest,1); jj = ones(elltest,1);
testY = (Xtest - jj*mux) * W + jj*muy;
testerror = norm(Ytest - testY,'fro')/sqrt(elltest)
```

Code Fragment 6.7. Matlab code for the primal PLS algorithm.

6.7.2 Kernel partial least squares

The projection direction in the feature space at each stage is given by the vector $\mathbf{u}_j$. This vector is in the primal space while we must work in the dual space. We therefore express a multiple of $\mathbf{u}_j$ as

$$a_j \mathbf{u}_j = \mathbf{X}'_j \boldsymbol{\beta}_j,$$

which is clearly consistent with the derivation of $\mathbf{u}_j$ in the primal PLS algorithm. For the dual PLS algorithm we must implement the deflation of $\mathbf{Y}$. This redundant step for the primal will be needed to get the required dual representations. We use the notation $\mathbf{Y}_j$ to denote the jth deflation. This leads to the following recursion for $\boldsymbol{\beta}$

$$\boldsymbol{\beta} = \mathbf{Y}_j \mathbf{Y}'_j \mathbf{X}_j \mathbf{X}'_j \boldsymbol{\beta} = \mathbf{Y}_j \mathbf{Y}'_j \mathbf{K}_j \boldsymbol{\beta}$$

$$\text{with the normalisation} \quad \boldsymbol{\beta} \;=\; \frac{\boldsymbol{\beta}}{\|\boldsymbol{\beta}\|}.$$

This converges to a dual representation $\boldsymbol{\beta}_j$ of a scaled version $a_j\mathbf{u}_j$ of $\mathbf{u}_j$, where note that we have moved to a kernel matrix $\mathbf{K}_j$. Now we need to compute a rescaled $\boldsymbol{\tau}_j = a_j\mathbf{X}_j\mathbf{u}_j$ and $\mathbf{c}_j$ from $\boldsymbol{\beta}_j$. We have

$$\boldsymbol{\tau}_j = a_j\mathbf{X}_j\mathbf{u}_j = \mathbf{X}_j\mathbf{X}_j'\boldsymbol{\beta}_j = \mathbf{K}_j\boldsymbol{\beta}_j,$$

while we work with a rescaled version $\hat{\mathbf{c}}_j$ of $\mathbf{c}_j$

$$\hat{\mathbf{c}}_j = \frac{\mathbf{Y}_j'\boldsymbol{\tau}_j}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j} = \frac{\mathbf{Y}_j'\mathbf{X}_j\mathbf{u}_j}{a_j\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = \frac{1}{a_j}\mathbf{c}_j,$$

so that we can consider $\boldsymbol{\tau}_j$ as a rescaled dual representation of the output vector $\mathbf{c}_j$. However, when we compute the contribution to the training output values

$$\boldsymbol{\tau}_j\hat{\mathbf{c}}_j' = \mathbf{X}_j\mathbf{u}_j\left(\frac{\mathbf{Y}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right)',$$

the rescalings cancel to give the correct result. Again with an automatic correction for the rescaling, Algorithm 6.42 gives the deflation of $\mathbf{X}_j$ as

$$\mathbf{X}_{j+1} = \left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right)\mathbf{X}_j,$$

with an equivalent deflation of the kernel matrix given by

$$\begin{aligned}
\mathbf{K}_{j+1} &= \mathbf{X}_{j+1}\mathbf{X}_{j+1}' \\
&= \left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right)\mathbf{X}_j\mathbf{X}_j'\left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right) \\
&= \left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right)\mathbf{K}_j\left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right),
\end{aligned}$$

all computable without explicit feature vectors. We also need to consider the vectors $\mathbf{p}_j$

$$\mathbf{p}_j = \frac{\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = a_j\frac{\mathbf{X}_j'\boldsymbol{\tau}_j}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}.$$

Properties of the dual computations We now consider the properties of these new quantities. First observe that the $\boldsymbol{\tau}_j$ are orthogonal since for $j > i$

$$\boldsymbol{\tau}_j' \boldsymbol{\tau}_i = a_j a_i \mathbf{u}_j' \mathbf{X}_j' \mathbf{X}_i \mathbf{u}_i = 0,$$

as the columns of $\mathbf{X}_j$ are all orthogonal to $\mathbf{X}_i \mathbf{u}_i$. This furthermore means that for $i < j$

$$\left(\mathbf{I} - \frac{\boldsymbol{\tau}_i \boldsymbol{\tau}_i'}{\boldsymbol{\tau}_i' \boldsymbol{\tau}_i} \right) \boldsymbol{\tau}_j = \boldsymbol{\tau}_j,$$

implying

$$\mathbf{X}_j' \boldsymbol{\tau}_j = \mathbf{X}' \boldsymbol{\tau}_j,$$

so that

$$\mathbf{p}_j = a_j \frac{\mathbf{X}' \boldsymbol{\tau}_j}{\boldsymbol{\tau}_j' \boldsymbol{\tau}_j}.$$

Note $\boldsymbol{\beta}_j$ can be written as $\mathbf{Y}_j \mathbf{x}_j$ for $\mathbf{x}_j = b_j \mathbf{Y}_j' \mathbf{K}_j \boldsymbol{\beta}_j$, for some scaling b_j. This implies that provided we deflate $\mathbf{Y}$ using

$$\mathbf{Y}_{j+1} = \left(\mathbf{I} - \frac{\boldsymbol{\tau}_j \boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j' \boldsymbol{\tau}_j} \right) \mathbf{Y}_j,$$

so the columns of $\mathbf{Y}_j$ are also orthogonal to $\mathbf{X}_i \mathbf{u}_i$ for $i < j$, it follows that

$$\boldsymbol{\beta}_j' \boldsymbol{\tau}_i = \mathbf{x}_j' \mathbf{Y}_j' \mathbf{X}_i \mathbf{u}_i = 0.$$

From this we have

$$\left(\mathbf{I} - \frac{\boldsymbol{\tau}_i \boldsymbol{\tau}_i'}{\boldsymbol{\tau}_i' \boldsymbol{\tau}_i} \right) \boldsymbol{\beta}_j = \boldsymbol{\beta}_j,$$

for $i < j$, so that

$$\mathbf{X}_j' \boldsymbol{\beta}_j = \mathbf{X}' \boldsymbol{\beta}_j$$

Computing the regression coefficients All that remains to be computed are the regression coefficients. These again must be computed in dual form, that is we require

$$\mathbf{W} = \mathbf{X}' \boldsymbol{\alpha},$$

so that a new input $\boldsymbol{\phi}(\mathbf{x})$ can be processed using

$$\boldsymbol{\phi}(\mathbf{x})' \mathbf{W} = \boldsymbol{\phi}(\mathbf{x})' \mathbf{X}' \boldsymbol{\alpha} = \mathbf{k}' \boldsymbol{\alpha},$$

where $\mathbf{k}$ is the vector of inner products between the test point and the training inputs. From the analysis of the primal PLS in equation (6.31) we have

$$\mathbf{W} = \mathbf{U}\left(\mathbf{P}'\mathbf{U}\right)^{-1}\mathbf{C}'.$$

Using $\mathbf{B}$ to denote the matrix with columns $\boldsymbol{\beta}_j$ and $\operatorname{diag}(\mathbf{a})$ for the diagonal matrix with entries $\operatorname{diag}(\mathbf{a})_{ii} = a_i$, we can write

$$\mathbf{U} = \mathbf{X}'\mathbf{B}\operatorname{diag}(\mathbf{a})^{-1}.$$

Similarly using $\mathbf{T}$ to denote the matrix with columns $\boldsymbol{\tau}_j$

$$\begin{aligned}\mathbf{P}'\mathbf{U} &= \operatorname{diag}(\mathbf{a})\operatorname{diag}\left(\boldsymbol{\tau}_i'\boldsymbol{\tau}_i\right)^{-1}\mathbf{T}'\mathbf{X}\mathbf{X}'\mathbf{B}\operatorname{diag}(\mathbf{a})^{-1}\\ &= \operatorname{diag}(\mathbf{a})\operatorname{diag}\left(\boldsymbol{\tau}_i'\boldsymbol{\tau}_i\right)^{-1}\mathbf{T}'\mathbf{K}\mathbf{B}\operatorname{diag}(\mathbf{a})^{-1}.\end{aligned}$$

Here $\operatorname{diag}(\boldsymbol{\tau}_i'\boldsymbol{\tau}_i)$ is the diagonal matrix with entries $\operatorname{diag}(\boldsymbol{\tau}_i'\boldsymbol{\tau}_i)_{ii} = \boldsymbol{\tau}_i'\boldsymbol{\tau}_i$. Finally, again using the orthogonality of $\mathbf{X}_j\mathbf{u}_j$ to $\boldsymbol{\tau}_i$, for $i < j$, we obtain

$$\mathbf{c}_j = \frac{\mathbf{Y}_j'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = \frac{\mathbf{Y}'\mathbf{X}_j\mathbf{u}_j}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j} = a_j\frac{\mathbf{Y}'\boldsymbol{\tau}_j}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j},$$

making

$$\mathbf{C} = \mathbf{Y}'\mathbf{T}\operatorname{diag}\left(\boldsymbol{\tau}_i'\boldsymbol{\tau}_i\right)^{-1}\operatorname{diag}(\mathbf{a}).$$

Putting the pieces together we can compute the dual regression variables as

$$\boldsymbol{\alpha} = \mathbf{B}\left(\mathbf{T}'\mathbf{K}\mathbf{B}\right)^{-1}\mathbf{T}'\mathbf{Y}.$$

Finally, the dual solution is given component-wise by

$$f_j(\mathbf{x}) = \sum_{i=1}^{\ell}\boldsymbol{\alpha}_i^j\kappa(\mathbf{x}_i, \mathbf{x}), \quad j = 1, \ldots, m.$$

Remark 6.46 [Rescaling matrices] Observe that

$$\mathbf{T}'\mathbf{K}\mathbf{B} = \operatorname{diag}\left(\boldsymbol{\tau}_i'\boldsymbol{\tau}_i\right)\operatorname{diag}(\mathbf{a})^{-\mathbf{1}}\mathbf{P}'\mathbf{U}\operatorname{diag}(\mathbf{a})$$

and so is also upper triangular, but with rows and columns rescaled. The rescaling caused by $\operatorname{diag}(\boldsymbol{\tau}_\mathbf{i}'\boldsymbol{\tau}_\mathbf{i})$ could be removed since we can easily compute this matrix. This might be advantageous to increase the numerical stability, since $\mathbf{P}'\mathbf{U}$ was optimally stable with diagonal entries 1, so the smaller the rescalings the better. The matrix $\operatorname{diag}(\mathbf{a})$ on the other hand is not readily accessible. ■

Remark 6.47 [Notation] The following table summarises the notation used in the above derivations:

$\mathbf{u}_j$	primal projection directions	$\boldsymbol{\beta}_j$	dual projection directions
$\mathbf{U}$	matrix with columns $\mathbf{u}_j$	$\mathbf{B}$	matrix with columns β_j
$\mathbf{c}_j$	primal output vector	τ_j	dual of scaled output vector
$\mathbf{C}$	matrix with columns $\mathbf{c}_j$	$\mathbf{T}$	matrix with columns $\boldsymbol{\tau}_j$
$\mathbf{W}$	primal regression coefficients	$\boldsymbol{\alpha}$	dual regression coefficients
$\mathbf{P}$	matrix with columns $\mathbf{p}_j$	$\mathbf{K}$	kernel matrix

■

Algorithm 6.48 [Kernel PLS] The kernel PLS algorithm is given in Code Fragment 6.8. Code Fragment 6.9 gives Matlab code for the complete PLS

input	Data $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, dimension k, target outputs $\mathbf{Y} \in \mathbb{R}^{\ell \times m}$.
process	$\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, \ldots, \ell$ $\mathbf{K}_1 = \mathbf{K}$ $\hat{\mathbf{Y}} = \mathbf{Y}$ for $j = 1, \ldots, k$ $\boldsymbol{\beta}_j$=first column of $\hat{\mathbf{Y}}$ $\boldsymbol{\beta}_j = \boldsymbol{\beta}_j / \|\boldsymbol{\beta}_j\|$ repeat $\boldsymbol{\beta}_j = \hat{\mathbf{Y}}\hat{\mathbf{Y}}'\mathbf{K}_j\boldsymbol{\beta}_j$ $\boldsymbol{\beta}_j = \boldsymbol{\beta}_j / \|\boldsymbol{\beta}_j\|$ until convergence $\boldsymbol{\tau}_j = \mathbf{K}_j\boldsymbol{\beta}_j$ $\mathbf{c}_j = \hat{\mathbf{Y}}'\boldsymbol{\tau}_j / \|\boldsymbol{\tau}_j\|^2$ $\hat{\mathbf{Y}} = \hat{\mathbf{Y}} - \boldsymbol{\tau}_j\mathbf{c}_j'$ $\mathbf{K}_{j+1} = \left(\mathbf{I} - \boldsymbol{\tau}_j\boldsymbol{\tau}_j' / \|\boldsymbol{\tau}_j\|^2\right)\mathbf{K}_j\left(\mathbf{I} - \boldsymbol{\tau}_j\boldsymbol{\tau}_j' / \|\boldsymbol{\tau}_j\|^2\right)$ end $\mathbf{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k]$ $\mathbf{T} = [\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_k]$ $\boldsymbol{\alpha} = \mathbf{B}\left(\mathbf{T}'\mathbf{K}\mathbf{B}\right)^{-1}\mathbf{T}'\mathbf{Y}$
output	Training outputs $\mathbf{Y} - \hat{\mathbf{Y}}$ and dual regression coefficients $\boldsymbol{\alpha}$

Code Fragment 6.8. Pseudocode for the kernel PLS algorithm.

algorithm in dual form. Note that it should also begin by centering the data but we have for brevity omitted this step (see Code Fragment 5.2 for Matlab code for centering). ■

```
 % K is an ell x ell kernel matrix
 % Y is ell x m containing the corresponding output vectors
 % T gives the number of iterations to be performed
KK = K; YY = Y;
for i=1:T
 YYK = YY*YY'*KK;
 beta(:,i) = YY(:,1)/norm(YY(:,1));
 if size(YY,2) > 1, % only loop if dimension greater than 1
   bold = beta(:,i) + 1;
   while norm(beta(:,i) - bold) > 0.001,
     bold = beta(:,i);
     tbeta = YYK*beta(:,i);
     beta(:,i) = tbeta/norm(tbeta);
   end
 end
 tau(:,i) = KK*beta(:,i);
 val = tau(:,i)'*t(:,i);
 c(:,i) = YY'*tau(:,i)/val;
 trainY = trainY + tau(:,i)*c(:,i)';
 trainerror = norm(Y - trainY,'fro')/sqrt(ell)
 w = KK*tau(:,i)/val;
 KK = KK - tau(:,i)*w' - w*tau(:,i)'
       + tau(:,i)*tau(:,i)'*(tau(:,i)'*w)/val;
 YY = YY - tau(:,i)*c(:,i)';
end
% Regression coefficients for new data
alpha = beta * ((tau'*K*beta)\tau')*Y;
%  Ktest gives new data inner products as rows, Ytest true outputs
elltest = size(Xtest,1);
testY = Ktest * alpha;
testerror = norm(Ytest - testY,'fro')/sqrt(elltest)
```

Code Fragment 6.9. Matlab code for the dual PLS algorithm.

6.8 Summary

- Eigenanalysis can be used to detect patterns within sets of vectors.
- Principal components analysis finds directions based on the variance of the data.
- The singular value decomposition of a covariance matrix finds directions of maximal covariance.
- Canonical correlation analysis finds directions of maximum correlation.
- Fisher discriminant analysis can also be derived as the solution of a generalised eigenvalue problem.
- The methods can be implemented in kernel-defined feature spaces.
- The patterns detected can also be used as feature selection methods for subsequent analysis, as for example principal components regression.

- The iterative use of directions of maximal covariance in regression gives the state-of-the-art partial least squares regression procedure, again implementable in kernel-defined feature spaces.

6.9 Further reading and advanced topics

The use of eigenproblems to solve statistical problems dates back to the 1930s. In 1936 Sir Ronald Fisher, the English statistician who pioneered modern data analysis, published 'The use of multiple measurements in taxonomic problems', where his linear discriminant algorithm is described [44]. The basic ideas behind principal components analysis (PCA) date back to Karl Pearson in 1901, but the general procedure as described in this book was developed by Harold Hotelling, whose pioneering paper 'Analysis of a Complex of Statistical Variables with Principal Component' appeared in 1933 [59]. A few years later in 1936, Hotelling [60] further introduced canonical correlation analysis (CCA), with the article 'Relations between two sets of variables'.

So in very few years much of multivariate statistics had been introduced, although it was not until the advent of modern computers that it could show its full power. All of these algorithms were linear and were not regularised. Classically they were justified under the assumption that the data was generated according to a Gaussian distribution, but the main computational steps are the same as the ones described and generalised in this chapter. For an introduction to classical multivariate statistics see [156]. The statistical analysis of PCA is based on the papers [125] and [124]. Many of these methods suffer from overfitting when directly applied to high-dimensional data. The need for regularisation was, for example, recognised by Vinod in [149]. A nice unified survey of eigenproblems in pattern recognition can be found in [15].

The development of the related algorithm of partial least squares has in contrast been rather different. It was introduced by Wold [159] in 1966 and developed in [161], [160], see also Höskuldsson [58] and Wold [162] for a full account. It has mostly been developed and applied in the field of chemometrics, where it is common to have very high-dimensional data. Based on ideas motivated by conjugate gradient methods in least squares problems (see for example conjugate gradient in [49]), it has been used in applications for many years. Background material on SVD and generalised eigenproblems can be found in many linear algebra books, for example [96].

The enhancement of these classical methods with the use of kernels has been a recurring theme over the last few years in the development of kernel

methods. Schölkopf *et al.* introduced it with kernel PCA [119]. Later several groups produced versions of kernel CCA [7], [81], [2], and of kernel FDA [98], [11]. Kernel PLS was introduced by Rosipal and Trejo [110].

Applications of kernel CCA in cross-lingual information retrieval are described in [149] while applications in bioinformatics are covered in [165], [147]. Kernel CCA is also described in the book [129].

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