Machine Learning Module

Week 7

Lecture Notes 13 & 14

Principal Component Analysis

Mark Girolami
girolami@dcs.gla.ac.uk
Department of Computing Science
University of Glasgow

February 20, 2006

1 Principal Component Analysis: Motivation

We have been asked to devise a linear classifier to discriminate between individuals who wear spectacles and those who do not based on a 64×64 dimensional 8-bit grey-scale image of their faces. There are 400 examples of various faces in the Matlab file `olivettifaces.mat` and some of these are shown in Figure (1).

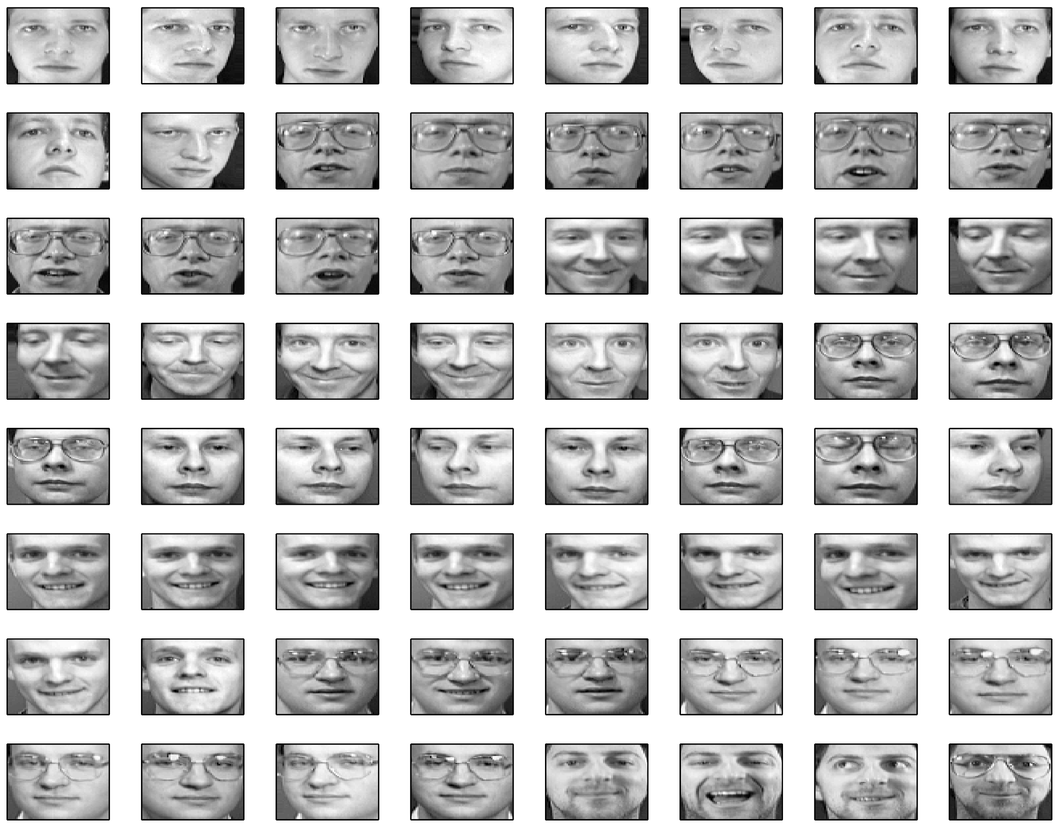

Figure 1: A sample of 68 faces from the file `olivettifaces.mat` showing the variability of the images based on, for example, pose and lighting.

Now as each 64×64 grey-scale image has $64^2 = 4096$ pixels taking integer values in the range $0 - 2^8(256)$ then each image can be represented as a $M = 4096 \times 1$ dimensional vector $\mathbf{x}$ and the whole collection of $N = 400$ images can be stored in the $N = 400 \times M = 4096$ dimensional matrix $\mathbf{X}$.

If we were to use a discriminative classifier of the form

$$\log \frac{P(C = 1|\mathbf{x})}{P(C = 0|\mathbf{x})} = \mathbf{w}^{\mathsf{T}}\mathbf{x}$$

,where $C = 1 \equiv \textsf{Spectacles}$ and $C = 0 \equiv \textsf{No Spectacles}$, then we see that as $\mathbf{w}$ has dimension $M \times 1$ there are a large number of parameters to estimate. Given that there are only 400 examples from which to identify these parameters it is clear that we will run the risk of overfitting our classifier to the available images (cast your mind back to Week 2 lectures on Generalisation). This is a problem.

Consider the variability observed in the images. From the small example shown in Figure (1) we can see that differences in pose (head on, facing diagonally, looking up, down), facial expression (grinning, smiling, scowling, open mouthed etc), wearing of glasses, presence of beard, shape of face, lighting and so forth, account for the inherent variability in this data. It would then appear that the variability in the image data available could be attributed to a relatively small number of degrees of freedom (pose, expression and so forth).

As we have 64×64 dimensional grey scale images the total number of possible images which could be produced from such a representation is 256^{4096}[1]. A few thousand example faces would be enough to be able to identify a face in general.

This leads us to consider the notion that the images of faces may well be described by a subspace of the 4096 dimensional pixel space. In other words the data lies in a lower-dimensional feature space which accounts for all of the information or variability in the images. If we can somehow extract these features from our original representation $\mathbf{X}$ then it may be possible to overcome the potential generalisation problem we face.

[1]This is a staggeringly large number i.e. $256^{4096} = 2^{8 \times 4096}$ which is vastly larger than the number of atoms in the entire universe 2^{784}.

2 PCA Linear Sub-Spaces

We are assuming that our M dimensional data actually lies within a P dimensional subspace where $P << M$. It will be further assumed that the subspace is linear such that a set of linearly independent orthonormal[2] basis vectors (coordinates) span this subspace i.e. $\{\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_P\}$ where each $\boldsymbol{\beta}_p \in \mathbb{R}^D$. So each data point $\mathbf{x}$ can be approximated by a linear combination of these basis vectors

$$\mathbf{x}_n \approx \sum_{p=1}^{P} u_{np}\boldsymbol{\beta}_p = \mathbf{B}\mathbf{u}_n$$

where the $D \times P$ dimensional matrix $\mathbf{B} = [\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_P]$ and $\mathbf{u}_n$ is a $P \times 1$ dimensional vector. We can consider this as a descriptive model of our data where points $\mathbf{u}_n$ in a P-dimensional space defined by $\{\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_P\}$ generate corresponding points $\mathbf{x}_n$ in the original M-dimensional data space.

2.1 PCA Derivation

Consider the limiting case where $P = 1$, i.e. that the data $\mathbf{X}$ is modeled as residing around a 1-dimensional linear subspace $\boldsymbol{\beta}_1$. Assume that the data $\mathbf{X}$ has zero mean.

The squared reconstruction error incurred when making the approximation $\mathbf{x}_n = u_{1n}\boldsymbol{\beta}_1$ is defined as

$$\mathcal{E} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - u_{1n}\boldsymbol{\beta}_1)^2$$

taking derivatives with respect to each u_{1n} and setting to zero gives

$$\frac{\partial \mathcal{E}}{\partial u_{1n}} = -\frac{2}{N}(\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n - u_{1n}) = 0 \Rightarrow u_{1n} = \boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n$$

[2]For a set of vectors, $\{\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_P\}$, to be orthonormal they must satisfy two conditions, (1) that the norm (vector length) of each vector is one i.e. $\boldsymbol{\beta}_i^{\mathsf{T}}\boldsymbol{\beta}_i = 1$. (2) that $\boldsymbol{\beta}_i^{\mathsf{T}}\boldsymbol{\beta}_j = 0 \;\forall\; i \neq j$

plugging this value back into the expression for $\mathcal{E}$ yields

$$\begin{aligned}
\mathcal{E} &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - u_{1n}\boldsymbol{\beta}_1)^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - 2u_{1n}\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n + u_{1n}^2\boldsymbol{\beta}_1^{\mathsf{T}}\boldsymbol{\beta}_1 \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - 2u_{1n}^2 + u_{1n}^2\boldsymbol{\beta}_1^{\mathsf{T}}\boldsymbol{\beta}_1 \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - u_{1n}^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - \boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n\mathbf{x}_n^{\mathsf{T}}\boldsymbol{\beta}_1
\end{aligned}$$

So to minimise our reconstruction error we require to maximise

$$\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n\mathbf{x}_n^{\mathsf{T}}\boldsymbol{\beta}_1 = \frac{1}{N}\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta}_1 = \boldsymbol{\beta}_1^{\mathsf{T}}\widehat{\mathbf{C}}\boldsymbol{\beta}_1$$

subject to $\boldsymbol{\beta}_1^{\mathsf{T}}\boldsymbol{\beta}_1 = 1$ where the sample covariance matrix is denoted as $\widehat{\mathbf{C}}$ (remember that each $\mathbf{X}$ is zero mean).

2.2 Variance Maximisation

It is interesting to note that minimisation of reconstruction error by maximisation of

$$\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n\mathbf{x}_n^{\mathsf{T}}\boldsymbol{\beta}_1 = \frac{1}{N}\sum_{n=1}^{N}u_{1n}^2$$

provides projections which are maximally informative.

Remember that we are restricting each basis-vector to have unit norm in which case we require to create the Langrangian (Refer to the Week 5 notes)

$$\boldsymbol{\beta}_1^{\mathsf{T}}\widehat{\mathbf{C}}\boldsymbol{\beta}_1 - \lambda_1\boldsymbol{\beta}_1^{\mathsf{T}}\boldsymbol{\beta}_1$$

and maximise with respect to $\boldsymbol{\beta}_1$. The corresponding vector of partial derivatives gives

$$\frac{\partial}{\partial \boldsymbol{\beta}_1} = \widehat{\mathbf{C}}\boldsymbol{\beta}_1 - \lambda_1 \boldsymbol{\beta}_1$$

and setting to zero then we have a standard eigenvalue problem to solve i.e.

$$\widehat{\mathbf{C}}\boldsymbol{\beta}_1 = \lambda_1 \boldsymbol{\beta}_1$$

As the variance of the projection is defined $\boldsymbol{\beta}_1^\mathsf{T}\widehat{\mathbf{C}}\boldsymbol{\beta}_1$ then for $\boldsymbol{\beta}_1^\mathsf{T}\boldsymbol{\beta}_1 = 1$ it should be clear that the variance of the projection is equal to λ_1 the associated eigenvalue. We will discuss how to numerically solve this eigenvalue problem in the following section.

We have now found the direction $\boldsymbol{\beta}_1$ which maximises the variance of the projection $\boldsymbol{\beta}_1^\mathsf{T}\mathbf{x}$ and correspondingly minimises the reconstruction error

$$\mathcal{E} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - u_{1n}\boldsymbol{\beta}_1)^2$$

This is referred to as the First Principal Direction and the projections of the data in this direction are the Principal Components in this direction.

2.3 Finding Additional Directions

Now we want to find another direction vector $\boldsymbol{\beta}_2$ which will satisfy $\boldsymbol{\beta}_1^\mathsf{T}\boldsymbol{\beta}_2 = 0$ and $\boldsymbol{\beta}_2^\mathsf{T}\boldsymbol{\beta}_2 = 1$ such that the two vectors $\{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2\}$ will form a 2-D orthonormal basis for our data. The approximations of points in data space will now take the form of

$$\mathbf{x}_n \approx \sum_{p=1}^{P=2} u_{np}\boldsymbol{\beta}_p =$$

and so we wish that the reconstruction error will be maximally reduced

$$\mathcal{E} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - u_{1n}\boldsymbol{\beta}_1 - u_{2n}\boldsymbol{\beta}_2)^2$$

it is straightforward to see that $u_{2n} = \boldsymbol{\beta}_2^\mathsf{T}\mathbf{x}_n$ and so following on from above the reconstruction error can now be obtained as the following where the orthonormal characteristics of both directions has been exploited

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - \boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{x}_n\mathbf{x}_n^{\mathsf{T}}\boldsymbol{\beta}_1 - \boldsymbol{\beta}_2^{\mathsf{T}}\mathbf{x}_n\mathbf{x}_n^{\mathsf{T}}\boldsymbol{\beta}_2$$

It is clear that given $\boldsymbol{\beta}_1$ then we require to obtain a solution for

$$\widehat{\mathbf{C}}\boldsymbol{\beta}_2 = \lambda_2\boldsymbol{\beta}_2$$

subject to the orthonormal constraints imposed. The following sub-section considers how we can achieve this in a practical way by exploiting some simple geometry.

2.4 Projection and Deflation

If $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are orthonormal then

$$\begin{aligned}\mathbf{x} &= u_1\boldsymbol{\beta}_1 + u_2\boldsymbol{\beta}_2\\ &= (\mathbf{x}^{\mathsf{T}}\boldsymbol{\beta}_1)\boldsymbol{\beta}_1 + (\mathbf{x}^{\mathsf{T}}\boldsymbol{\beta}_2)\boldsymbol{\beta}_2\end{aligned}$$

where $(\mathbf{x}^{\mathsf{T}}\boldsymbol{\beta}_2)\boldsymbol{\beta}_2$ is the projection orthogonal to $(\mathbf{x}^{\mathsf{T}}\boldsymbol{\beta}_1)\boldsymbol{\beta}_1$ in which case we can write the projection orthogonal to that of the first principal direction as

$$(\mathbf{x}^{\mathsf{T}}\boldsymbol{\beta}_2)\boldsymbol{\beta}_2 = (\mathbf{I} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}})\mathbf{x}$$

Applying this to all of the data gives

$$\mathbf{X}(\mathbf{I} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}})^{\mathsf{T}} = \mathbf{X}(\mathbf{I} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}})$$

We can think of this operation as removing from the D-dimensional data the component that lies in the direction of the first principal direction. In other words we are deflating the matrix $\mathbf{X}$ and thus reducing its rank from D to $D-1$ i.e. removing one direction component, the principal direction.

Consider then the covariance of this deflated data matrix $\widetilde{\mathbf{X}} = \mathbf{X}(\mathbf{I} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}})$

$$\begin{aligned}\frac{1}{N}\widetilde{\mathbf{X}}^{\mathsf{T}}\widetilde{\mathbf{X}} &= \frac{1}{N}(\mathbf{I} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}})\widetilde{\mathbf{X}}^{\mathsf{T}}\widetilde{\mathbf{X}}(\mathbf{I} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}})\\ &= \frac{1}{N}\left(\mathbf{X}^{\mathsf{T}}\mathbf{X} - \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X} - \mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}} + \boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}}\right)\end{aligned}$$

Taking this expression term by term we see that the right hand term can be written as

$$\boldsymbol{\beta}_1 \left(\boldsymbol{\beta}_1^{\mathsf{T}} \mathbf{X}^{\mathsf{T}} \mathbf{X} \boldsymbol{\beta}_1\right) \boldsymbol{\beta}_1^{\mathsf{T}} = \boldsymbol{\beta}_1 (N\lambda_1) \boldsymbol{\beta}_1^{\mathsf{T}} = N\lambda_1 \boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}}$$

For

$$\boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}} \mathbf{X}^{\mathsf{T}} \mathbf{X} = \boldsymbol{\beta}_1 (N\lambda_1 \boldsymbol{\beta}_1^{\mathsf{T}}) = N\lambda_1 \boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}}$$

and

$$\mathbf{X}^{\mathsf{T}} \mathbf{X} \boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}} = N\lambda_1 \boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}}$$

plugging these into the expression for the covariance we obtain

$$\begin{aligned} \widetilde{\mathbf{C}} &= \frac{1}{N} \widetilde{\mathbf{X}}^{\mathsf{T}} \widetilde{\mathbf{X}} \\ &= \frac{1}{N} \mathbf{X}^{\mathsf{T}} \mathbf{X} - \lambda_1 \boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}} \\ &= \widehat{\mathbf{C}} - \lambda_1 \boldsymbol{\beta}_1 \boldsymbol{\beta}_1^{\mathsf{T}} \end{aligned}$$

If we then find the principal direction associated with the deflated covariance matrix $\widetilde{\mathbf{C}}$ by solving the eigenvalue problem

$$\widetilde{\mathbf{C}} \boldsymbol{\beta}_2 = \lambda_2 \boldsymbol{\beta}_2$$

then of course $\boldsymbol{\beta}_2^{\mathsf{T}} \boldsymbol{\beta}_2 = 1$ and as the data $\widetilde{\mathbf{X}}$ resides in the $D-1$ dimensional subspace orthogonal to the first principal direction $\boldsymbol{\beta}_1$ then by definition $\boldsymbol{\beta}_1^{\mathsf{T}} \boldsymbol{\beta}_2 = 0$ must hold.

We will see further on that continuing this joint matrix deflation and solving of the associated eigenvalue problems will provide a set of eigenvectors $\{\boldsymbol{\beta}_1 \cdots \boldsymbol{\beta}_D\}$ and associated eigenvalues $\{\lambda_1 \cdots \lambda_D\}$ which provide an orthonormal basis for the data which when truncated at $P << D$ will provide the minimum reconstruction error, in the least squares sense, of the data.

2.5 Components of Reconstruction Error

The overall data reconstruction error can be written as

$$\begin{aligned}
\mathcal{E} &= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}_n - \sum_{p=1}^{P} u_{pn}\boldsymbol{\beta}_p\right)^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - \sum_{p=1}^{P}\boldsymbol{\beta}_p^{\mathsf{T}}\mathbf{x}_n\mathbf{x}_n^{\mathsf{T}}\boldsymbol{\beta}_p\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - \sum_{p=1}^{P}\lambda_p
\end{aligned}$$

Now if there is no truncation and $P = D$ then $\mathcal{E}$ is clearly zero in which case

$$\begin{aligned}
0 &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n^{\mathsf{T}}\mathbf{x}_n - \sum_{p=1}^{P}\lambda_p - \sum_{p'=P+1}^{D}\lambda_{p'} \\
&= \mathcal{E} - \sum_{p'=P+1}^{D}\lambda_{p'} \\
&\Rightarrow \mathcal{E} = \sum_{p'=P+1}^{D}\lambda_{p'}
\end{aligned}$$

and therefore the reconstruction error is composed of the sum of the eigenvalues associated with the principal components discarded in the truncation. As the first principal component provides the largest reduction in error and the second principal component (PC) is obtained from the deflated covariance matrix $\widehat{\mathbf{C}} - \lambda_1\boldsymbol{\beta}_1\boldsymbol{\beta}_1^{\mathsf{T}}$ then the reduction in error obtained by the second PC will be smaller than that obtained from the first as such $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \geq \lambda_D$.

This means that by studying the distribution of the eigenvalues we can potentially identify the intrinsic dimension of the data by assessing which dimensions incur the main contributions to the overall reconstruction error.

It also follows that

$$\widetilde{\mathbf{C}} = \mathbf{0} = \widehat{\mathbf{C}} - \sum_{p=1}^{D} \lambda_p \boldsymbol{\beta}_p \boldsymbol{\beta}_p^{\mathsf{T}} \Rightarrow \widehat{\mathbf{C}} = \sum_{p=1}^{D} \lambda_p \boldsymbol{\beta}_p \boldsymbol{\beta}_p^{\mathsf{T}}$$

so if we define the $D \times D$ matrix $\mathbf{B}$ whose columns are $\boldsymbol{\beta}_p$ and the $D \times D$ diagonal matrix $\mathbf{D}$ whose elements are each λ_p then the covariance matrix can be represented in terms of the associated eigenvalues and eigenvectors as

$$\widehat{\mathbf{C}} = \mathbf{B}\mathbf{D}\mathbf{B}^{\mathsf{T}}$$

2.6 Illustrative Example

Consider 200 samples of 2-dimensional data denoted by the matrix $\mathbf{X}$. The data is drawn from two 2-D isotropic Gaussian distributions centered at [-2, -2] and [+2, +2]. A plot is given in Figure(2) and the data was produced using the Matlab command

```
X=[randn(100,2)+2.*ones(100,2);2.*randn(100,2)-2.*ones(100,2)];
```

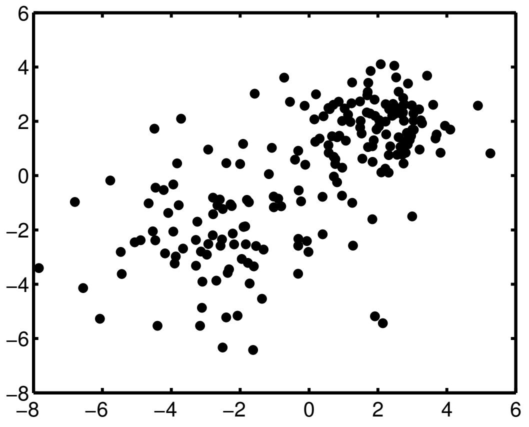

Figure 2: A scatter diagram of the 2-D data.

Now we generate a random 10×2 matrix $\mathbf{A}$ and apply the transformation $\widetilde{\mathbf{Y}} = \mathbf{X}\mathbf{A}$ such that the data has now been projected from the original 2-D space into a 10-D representation. Finally we set $\mathbf{Y} = \widetilde{\mathbf{Y}} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is isotropic noise with variance 2.

Given this 10-D data let us perform PCA on the data and study how the errors are distributed throughout the ten dimensions by plotting the 10 eigenvalues $\lambda_1 \cdots \lambda_{10}$. The matlab command `eig` performs the required eigenvalue decomposition[3] and the little segment of code below will produce the bar chart in Figure (3).

```
X=[randn(100,2)+2.*ones(100,2);2.*randn(100,2)-2.*ones(100,2)];
plot(X(:,1),X(:,2),'r.')
A=randn(10,2);
Y=X*A';
Y=Y+randn(200,10).*2;
[V,S]=eig(cov(Y));
bar(flipud(diag(S)))
```

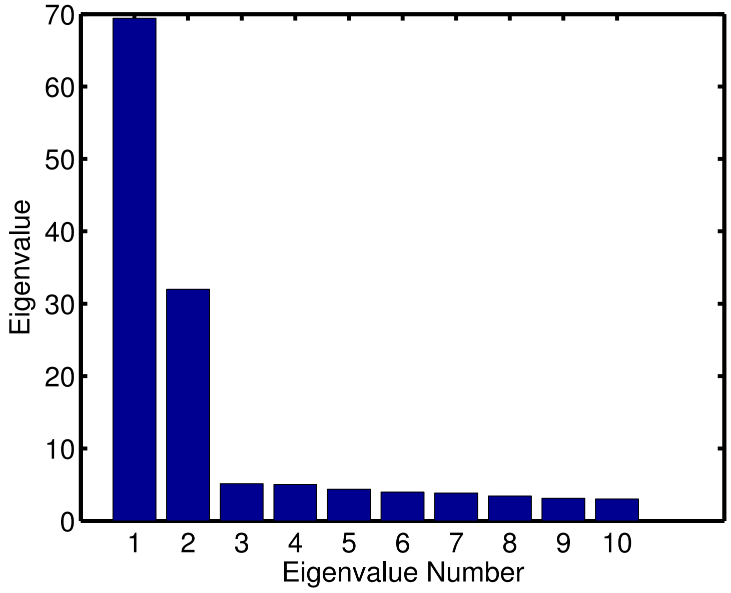

Figure 3: A bar chart of the eigenvalues $\lambda_1 \cdots \lambda_{10}$ for the 10-D data created from the procedure given above.

[3]We will be introduced to an algorithm for computing the eigenvectors and eigenvalues of a covariance matrix in the following section - just in case you were getting worried about being too reliant on the suite of functions which Matlab provides

The eigenvalues are [69.4287, 31.9839, 5.1432, 5.0361, 4.3781, 3.9894, 3.8538, 3.4494, 3.1129, 3.0384]. It should be clear that there are two principal directions which account for the majority of the reconstruction error with the first two accounting for over 76% of the total error, or put another way the total variance and hence information in the data. This strongly suggests that the data exists around a 2-D subspace and hence the main structure in the data has an intrinsic dimension of two.

Let us now revisit our collection of images of faces. Our data matrix $\mathbf{X}$ has dimension 400×4096 and so the covariance matrix will have dimension 4096×4096 which is huge relative to the number of examples available. Of course this matrix will only have rank $N = 400$ and so we will only be able to extract 400 principal components in which case we employ the following trick

$$\begin{aligned}
\widehat{\mathbf{C}} &= \mathbf{B}\mathbf{D}\mathbf{B}^{\mathsf{T}} \\
\Rightarrow & \frac{1}{N}\mathbf{X}^{\mathsf{T}}\mathbf{X} = \mathbf{B}\mathbf{D}\mathbf{B}^{\mathsf{T}} \\
\Rightarrow & \frac{1}{N}\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{B} = \mathbf{B}\mathbf{D} \\
\Rightarrow & \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{B} = \mathbf{X}\mathbf{B}\mathbf{D} \\
\Rightarrow & \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathsf{T}}\mathbf{U} = \mathbf{U}\mathbf{D}
\end{aligned}$$

where we have defined $\mathbf{U} = \mathbf{X}\mathbf{B}$. Now as there are only N non-zero eigenvalues then we can see that

$$\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathsf{T}}\mathbf{U} = \mathbf{U}\mathbf{D}$$

solves the eigenvalue problem for the matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathsf{T}}$ [4] where $\mathbf{U}$ has dimension $N \times N$ and $\mathbf{D}$ is an $N \times N$ dimensional diagonal matrix. So having obtained $\mathbf{U}$ and $\mathbf{D}$ by noting that

$$\frac{1}{N}\mathbf{X}\mathbf{X}^{\mathsf{T}}\mathbf{U}\mathbf{D}^{-1} = \mathbf{U} = \mathbf{X}\mathbf{B}$$

[4] You may notice that this is a matrix of inner products $\mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j$ which could of course be generalised to some kernel defined inner-product $k(\mathbf{x}_i, \mathbf{x}_j)$ and so opens up the possibility of performing PCA in a nonlinear space defined by the kernel function $k(.,.)$.

Then the $D \times N$ dimensional matrix of eigenvectors $\mathbf{B}$ follows simply as

$$\mathbf{B} = \frac{1}{N}\mathbf{X}^{\mathsf{T}}\mathbf{U}\mathbf{D}^{-1}$$

Remember that throughout we have assumed that $\mathbf{X}$ has zero-mean.

Armed with this alternative method for computing a PCA when $D >> N$ let us now consider the 400 non-zero eigenvalues for the database of images. The Matlab script `faces_demo.m` is available for download from the class webpage and it illustrates PCA applied to the images of faces in the file `olivettifaces.mat`.

Figure (4) shows the cumulative reduction in reconstruction error when increasing the number of principal components used to model the images.

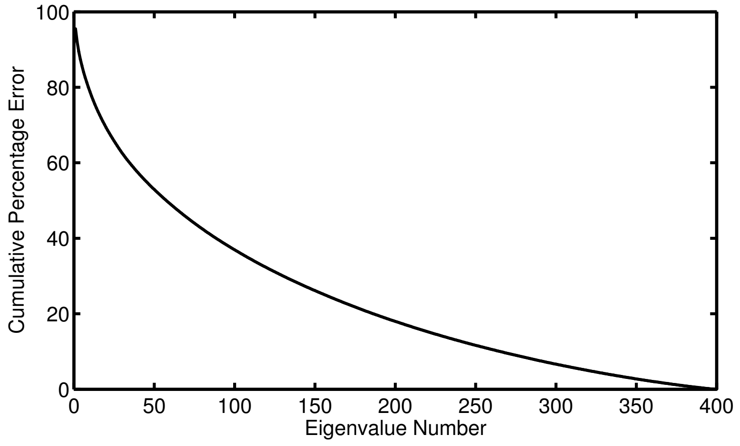

Figure 4: The percentage reconstruction error as principal components are included within the image representation.

The first thing to note is that an error of 20% is achieved when only 200 eigenvectors are employed. So this means that of the 4096 dimensions available a feature space of dimension less than 5% of this will capture around 80% of the variance in the images or will incur a 20% error. This is a massive amount of representation compression.

In Figure (5) the original image of a single face from the database is shown, the middle plot shows the reconstructed facial image i.e. $\mathbf{x}_n = \sum_{p=1}^{P} u_{pn}\boldsymbol{\beta}_p$ when the first ten ($P = 10$) PC's are employed. Features such as spectacles around the eyes have been captured but there are still more specific details required.

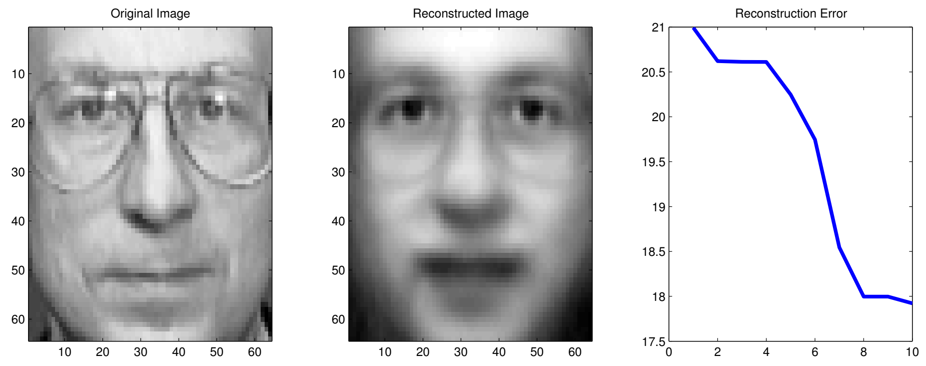

Figure 5: The original image (left) and the reconstructed image (middle) after ten principal components have been employed. The right hand plot shows how the error has decreased for this particular face over the ten PC's employed.

Figure (6) now shows how the reconstruction has progressed when $P = 100$, it is much more recognizable as the original face, however rather than using a 4096 dimensional feature representation a smaller 100 dimensional feature representation is required.

3 PCA Applications

It is clear that PCA can be employed in extracting linear feature sets from high-dimensional data is such a way that the maximum amount of data variance can be retained. This is a particularly important and in many cases useful application of PCA.

3.1 Improving Generalisation Performance

Recall that the variance of predictions made by linear regression models on data points $\mathbf{x}_*$ can be given as

$$\sigma^2 \mathbf{x}_*^\mathsf{T} (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{x}_*$$

and as

$$\mathbf{X}^\mathsf{T}\mathbf{X} = N\mathbf{B}\mathbf{D}\mathbf{B}^\mathsf{T}$$

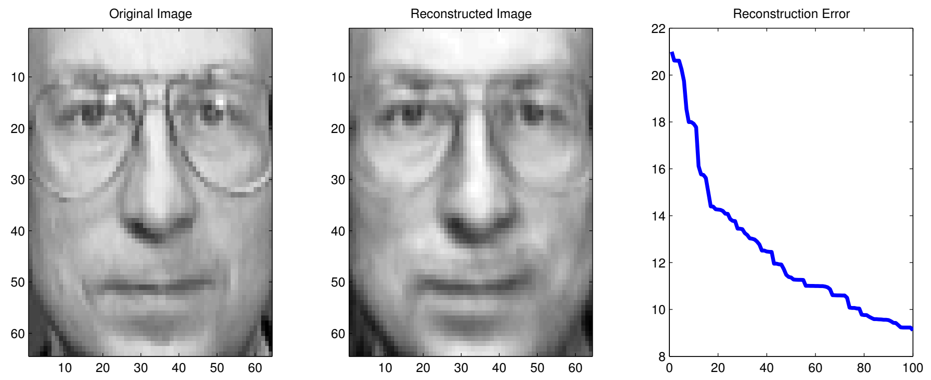

Figure 6: The original image (left) and the reconstructed image (middle) after one hundred principal components have been employed. The right hand plot shows how the error has decreased for this particular face over the one hundred PC's employed.

then given that $\mathbf{B}$ is an orthonormal matrix such that $\mathbf{B}^\mathsf{T}\mathbf{B} = \mathbf{I}$ then $\mathbf{B}^{-1} = \mathbf{B}^\mathsf{T}$ we can write

$$(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1} = \frac{1}{N}\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\mathsf{T} = \frac{1}{N}\sum_{p=1}^{D}\frac{1}{\lambda_p}\boldsymbol{\beta}_p\boldsymbol{\beta}_p^\mathsf{T}$$

So we see that directions which have small eigenvalues λ_p will make a large contribution to the variance of the estimate due to the $\frac{1}{\lambda_p}$ term occurring in the expansion. Therefore employing a smaller number of PC's by discarding the directions with small eigenvalues will reduce the prediction variance and hence improve the generalisation performance.

However, as we have a set of target values it would seem sensible that rather than seeking variance maximising projections to seek projections which maximise the covariance between the projection and the target values. There are a number of methods such as Partial Least Squares and Canonical Correlation Analysis which identify such directions. Nevertheless PCA is a very useful method of feature extraction and prediction variance control.

3.2 High Dimensional Data Visualisation

If the PC's capture the variability in the data projection onto the first number of PC's may enable visualisation of inherent data structure such as data clumps and clusters. As an example consider the microarray data shown in Figure (7) which measures the differential expression level of 243 genes (from the yeast organism) at 7 different time points.

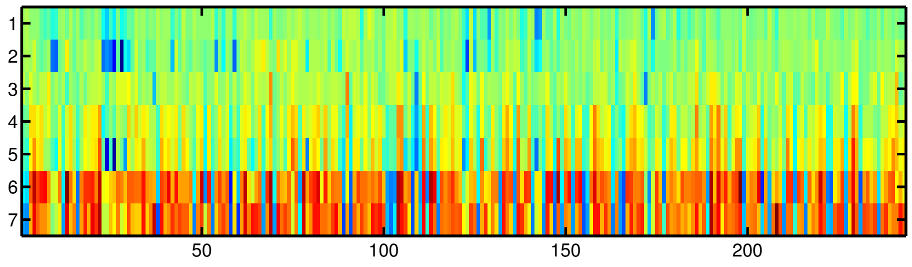

Figure 7: The differential gene expression levels of 243 genes measured at seven time points.

By performing PCA on the expression data and then plotting the projections of the temporal profile of each gene onto the first two PC's we obtain a scatter plot shown in Figure

We can see straightaway that there are two natural groupings of genes whose temporal profiles varies in a similar manner. It is difficult to identify this natural grouping from the original data.

4 Latent Semantic Analysis

Vector Space models for documents based Information Retrieval rely on a measure of similarity between a query $\mathbf{q}$ and some document $\mathbf{d}$ where both the document and query representations are based on weighted functions of the counts of the occurrence of dictionary terms and $\mathbf{q}$, $\mathbf{d} \in \mathbb{R}^{|\mathcal{D}|}$ where $|\mathcal{D}|$ denotes the number of terms in the dictionary. The cosine similarity is typically employed

$$sim(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q}^{\mathsf{T}}\mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$$

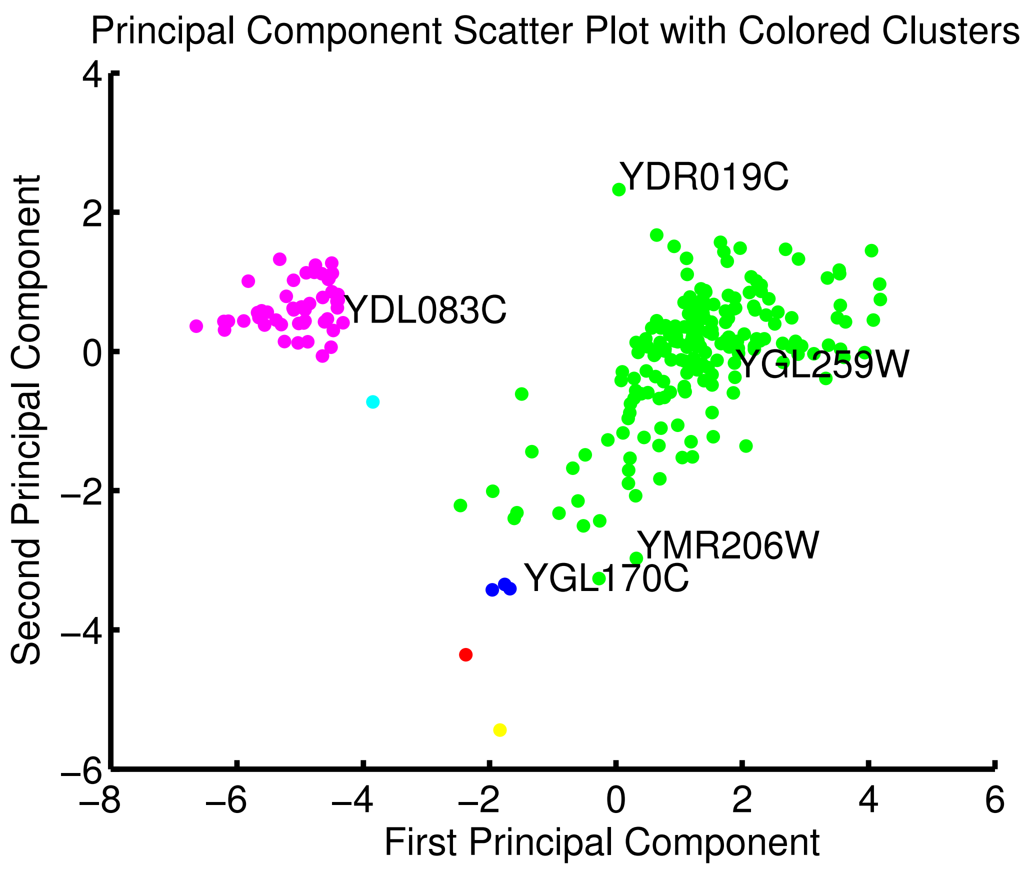

Figure 8: The projection of the differential gene expression levels of 243 genes onto the first two principal directions.

Now as the number of terms in the dictionary can be in the range of 10 to 30 thousand an argument can be made that there exists a lower-dimensional *semantic* space which captures the variability in the usage of terms which is in some way meaningful. If this is the case then the similarity matching would benefit from operating in the semantic space rather than the term space. In that case if we have a document collection represented as a vector space where there are N documents then the document by term matrix can be defined as $\mathbf{D} \in \mathbb{R}^{N\times|\mathcal{D}|}$. A query will be defined as $\mathbf{q} \in \mathbb{R}^{|\mathcal{D}|}$ and so the similarity of all documents to the query will be proportional to $\mathbf{q}^{\mathbf{D}}$.

Now the document by term matrix $\mathbf{D}$ can be approximated such that $\mathbf{D} \approx \mathbf{U}_\delta \mathbf{S}_\delta \mathbf{V}_\delta^{\mathsf{T}}$ where the subscript δ denotes the number of principal components which are retained. Projecting the document collection into the principal subspace of dimension δ gives $\mathbf{U}_\delta = \mathbf{D}\mathbf{V}_\delta\mathbf{S}_\delta^{-1}$ now a query can be projected into this subspace such that $\mathbf{q}_\delta = \mathbf{S}_\delta^{-1}\mathbf{V}_\delta^\mathsf{T}\mathbf{q}$ and so scoring of documents against the query would take place in the latent semantic space such that

$$\begin{aligned} sim(\mathbf{q}, \mathbf{D}) &\equiv \mathbf{q}_\delta^\mathsf{T}\mathbf{U}_\delta^\mathsf{T} \\ &= \mathbf{q}^\mathsf{T}\mathbf{V}_\delta\mathbf{S}_\delta^{-1}\mathbf{S}_\delta^{-1}\mathbf{V}_\delta^\mathsf{T}\mathbf{D}^\mathsf{T} \\ &= \mathbf{q}^\mathsf{T}\boldsymbol{\Delta}\mathbf{D}^\mathsf{T} \end{aligned}$$

where $\boldsymbol{\Delta} = \mathbf{V}_\delta\mathbf{S}_\delta^{-2}\mathbf{V}_\delta^\mathsf{T}$ acts to provide an appropriate scaling of the original coordinate axes in term space based on the structure of the latent semantic space as defined by the δ principal eigenvectors.

There is evidence to suggest that for some document collections and query sets Latent Sematic Analysis can provide superior scoring than a straightforward cosine measure in term space.

5 Computing a Principal Component Analysis

Up till now we have let the magic of Matlab compute our PCA for us. Now we will learn of one numerical method which will provide the eigenvectors and associated eigenvalues of a covariance matrix.

It can be shown that the following iteration will provide convergence to the principal eigenvector of the covariance matrix $\mathbf{C}$.

$$\begin{aligned} \mathbf{x}_t &= \mathbf{C}\mathbf{y}_{t-1} \\ \mathbf{y}_t &= \frac{\mathbf{x}_t}{\sqrt{\mathbf{x}_t^\mathsf{T}\mathbf{x}_t}} \end{aligned}$$

as $t \to \infty$ then $\mathbf{y}_t \to \boldsymbol{\beta}_1$ and $\sqrt{\mathbf{x}_t^\mathsf{T}\mathbf{x}_t} \to \lambda_1$. Now once the first eigenvector has been identified the covariance matrix is deflated as detailed previously

$$\mathbf{C} \leftarrow \mathbf{C} - \lambda_1\boldsymbol{\beta}_1\boldsymbol{\beta}_1^\mathsf{T}$$

and the above iteration is applied to the deflated matrix to obtain the second eigenvector and associated eigenvalue. This is repeated until all the eigenvector/value pairs are obtained.

The following Matlab script provides a simple implementation of the above algorithm.

```
function [B,D]=power_pca(C)
%A little routine to compute PCA given a covariance matrix C
N = size(C,1);

threshold = 1e-3; Max_Its = 1000;

%loop round all dimensions of the covariance matrix
for n=1:N

   %initialise the principal eigenvector and set norm to unity
   x = randn(N,1);
   y = x./sqrt(x'*x);
   %monitor convergence
   err = 1e20;
   its = 1;

   %main loop to compute single eigenvector
   while (err > threshold) | (its < Max_Its)
       x = C*y;
       y_new = x./sqrt(x'*x);
       err = sum((y_new - y).^2);
       y = y_new;
       %set eigenvalue
       D(n) = sqrt(x'*x);
       %increment counter
       its = its + 1;
   end

   %set the column vectors to be the found eigenvectors
   B(:,n) = y_new;
   %deflate the covariance matrix
   C = C - D(n)*y_new*y_new';
end
D=diag(D);
```