Lecture 14: Principal Component Analysis

Principal Component Analysis, or simply PCA, is a statistical procedure concerned with elucidating the covariance structure of a set of variables. In particular it allows us to identify the principal directions in which the data varies.

For example, in figure 1, suppose that the triangles represent a two variable data set which we have measured in the X-Y coordinate system. The principal direction in which the data varies is shown by the U axis and the second most important direction is the V axis orthogonal to it. If we place the $U - V$ axis system at the mean of the data it gives us a compact representation. If we transform each (X, Y) coordinate into its corresponding (U, V) value, the data is de-correlated, meaning that the co-variance between the U and V variables is zero. For a given set of data, principal component analysis finds the axis system defined by the principal directions of variance (ie the $U - V$ axis system in figure 1). The directions U and V are called the principal components.
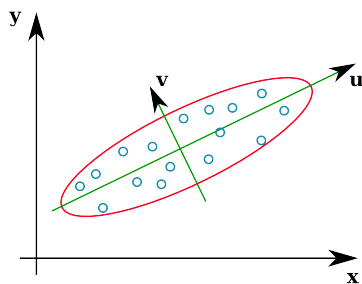

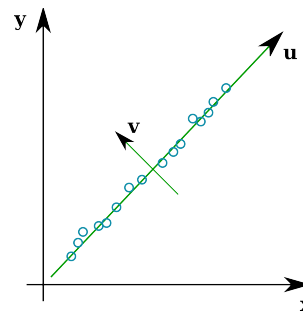Figure 1: PCA for Data Representation

Figure 2: PCA for Dimension Reduction

If the variation in a data set is caused by some natural property, or is caused by random experimental error, then we may expect it to be normally distributed. In this case we show the nominal extent of the normal distribution by a hyper-ellipse (the two dimensional ellipse in the example). The hyper ellipse encloses data points that are thought of as belonging to a class. It is drawn at a distance beyond which the probability of a point belonging to the class is low, and can be thought of as a class boundary.

If the variation in the data is caused by some other relationship then PCA gives us a way of reducing the dimensionality of a data set. Consider two variables that are nearly related linearly as shown in figure 2. As in figure 1 the principal direction in which the data varies is shown by the U axis, and the secondary direction by the V axis. However in this case all the V coordinates are all very close to zero. We may assume, for example, that they are only non zero because of experimental noise. Thus in the $U - V$ axis system we can represent the data set by one variable U and discard V. Thus we have reduced the dimensionality of the problem by 1.

Computing the Principal Components

In computational terms the principal components are found by calculating the eigenvectors and eigenvalues of the data covariance matrix. This process is equivalent to finding the axis system in which the co-variance matrix is diagonal. The eigenvector with the largest eigenvalue is the direction of greatest variation, the one with the second largest eigenvalue is the (orthogonal) direction with the next highest variation and so on. To see how the computation is done we will give a brief review on eigenvectors/eigenvalues.

Let A be an $n \times n$ matrix. The eigenvalues of A are defined as the roots of:

$$determinant(\boldsymbol{A} - \lambda \boldsymbol{I}) = |(\boldsymbol{A} - \lambda \boldsymbol{I})| = 0$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix. This equation is called the characteristic equation (or characteristic polynomial) and has n roots.

Let λ be an eigenvalue of A. Then there exists a vector $\boldsymbol{x}$ such that:

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

The vector $\boldsymbol{x}$ is called an eigenvector of $\boldsymbol{A}$ associated with the eigenvalue λ. Notice that there is no unique solution for $\boldsymbol{x}$ in the above equation. It is a direction vector only and can be scaled to any magnitude. To find a

numerical solution for x we need to set one of its elements to an arbitrary value, say 1, which gives us a set of simultaneous equations to solve for the other other elements. If there is no solution we repeat the process with another element. Ordinarily we normalise the final values so that x has length one, that is $x \cdot x^T = 1$.

Suppose we have a 3×3 matrix A with eigenvectors x_1, x_2, x_3, and eigenvalues $\lambda_1, \lambda_2, \lambda_3$ so:

$$Ax_1 = \lambda_1 x_1 \qquad Ax_2 = \lambda_2 x_2 \qquad Ax_3 = \lambda_3 x_3$$

Putting the eigenvectors as the columns of a matrix gives:

$$A \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

writing:

$$\Phi = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

gives us the matrix equation:

$$A\Phi = \Phi\Lambda$$

We normalised the eigenvectors to unit magnitude, and they are orthogonal, so:

$$\Phi\Phi^T = \Phi^T\Phi = I$$

which means that:

$$\Phi^T A \Phi = \Lambda$$

and:

$$A = \Phi\Lambda\Phi^T$$

Now let us consider how this applies to the covariance matrix in the PCA process. Let Σ be an $n \times n$ covariance matrix. There is an orthogonal $n \times n$ matrix Φ whose columns are eigenvectors of Σ and a diagonal matrix Λ whose diagonal elements are the eigenvalues of Σ, such that

$$\Phi^T \Sigma \Phi = \Lambda$$

We can look on the matrix of eigenvectors Φ as a linear transformation which, in the example of figure 1 transforms data points in the $[X, Y]$ axis system into the $[U, V]$ axis system. In the general case the linear transformation given by Φ transforms the data points into a data set where the variables are uncorrelated. The correlation matrix of the data in the new coordinate system is Λ which has zeros in all the off diagonal elements.
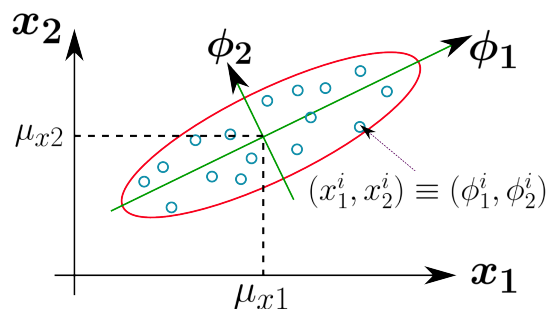
PCA in practice

Figure 3: The PCA Transformation

Figure 3 gives a geometric illustration of the process in two dimensions. Using all the data points we find the mean values of the variables (μ_{x1}, μ_{x2}) and the covariance matrix Σ which is a 2×2 matrix in this case. If we calculate the eigenvectors of the co-variance matrix we get the direction vectors indicated by ϕ_1 and ϕ_2.

Putting the two eigenvectors as columns in the matrix $\Phi = [\phi_1, \phi_2]$ we create a transformation matrix which takes our data points from the $[x_1, x_2]$ axis system to the axis $[\phi_1, \phi_2]$ system with the equation:

$$p_\phi = (p_x - \mu_x) \cdot \Phi$$

where p_x is any point in the $[x_1, x_2]$ axis system, $\mu_x = (\mu_{x1}, \mu_{x2})$ is the data mean, and p_ϕ is the coordinate of the point in the $[\phi_1, \phi_2]$ axis system.

Face Recognition

Face recognition is one example where principal component analysis has been extensively used, primarily for reducing the number of variables. Let us consider the 2D case where we have an input image, and wish to compare this with a set of data base images to find the best match. We assume that the images are all of the same resolution and are all equivalently framed. (ie the faces appear at the same place and same scale in the images). Each pixel can be considered a variable thus we have a very high dimensional problem which can be simplified by PCA.

Formally, in image recognition an input image with n pixels can be treated as a point in an n-dimensional space called the image space. The individual ordinates of this point represent the intensity values of each pixel of the image and form a row vector: $p_x = (i_1, i_2, i_3, ...i_n)$ This vector is formed by concatenating each row of image pixels, so for a modest sized image, say 128 by 128 resolution it will dimension 16384. For example:

$$= \begin{bmatrix} 150 & 152 & \cdot & 151 \\ 131 & 133 & \cdot & 72 \\ \cdot & \cdot & \cdot & \cdot \\ 144 & 171 & \cdot & 67 \end{bmatrix} \; 128 \times 128$$

becomes the vector:
$$[150, 152, \cdots 151, 131, 133, \cdots 72, \cdots 144, 171, \cdots 67]_{16K}$$

Clearly this number of variables is far more than is required for the problem. Most of the image pixels will be highly correlated. For example if the background pixels are all equal then adjacent background pixels are exactly correlated. Thus we need to consider how to achieve a reduction in the number of the variables.

Dimension Reduction

Lets consider an application where we have N images each with n pixels. We can write our entire data set as an $N \times n$ data matrix D. Each row of D represents one image of our data set. For example we may have:

$$D = \begin{bmatrix} 150 & 152 & \cdots & 254 & 255 & \cdots & 252 \\ 131 & 133 & \cdots & 221 & 223 & \cdots & 241 \\ \cdot & \cdot & & \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & \cdot & & \cdot \\ 144 & 171 & \cdots & 244 & 245 & \cdots & 223 \end{bmatrix} \; N \times n$$

The first step in PCA is to move the origin to mean of the data. In this application we achieve this by finding a mean image μ by averaging the columns of D. We then subtract the mean image from each image of the data set (ie each row of D) to create the mean centered data vector which we write as U. Let us suppose that the mean centered image is:

$$[120 \;\; 140 \;\; \cdots \;\; 230 \;\; 230 \;\; \cdots \;\; 240]$$

Then we have that:
$$U = \begin{bmatrix} 30 & 12 & \cdots & 24 & 25 & \cdots & 12 \\ 11 & -7 & \cdots & -9 & -7 & \cdots & 1 \\ \cdot & \cdot & & \cdot & \cdot & & \cdot \\ 24 & 31 & \cdots & 14 & 15 & \cdots & -17 \end{bmatrix} \; N \times n$$

It is very easy to compute the covariance matrix from the mean centered data matrix. It is just

$$\Sigma = \boldsymbol{U}^T \boldsymbol{U}/(N-1)$$

and has dimension $n \times n$. We now calculate the eigenvectors and eigenvalues of Σ using the standard technique outlined above. That is to say we solve for Φ and Λ that satisfy:

$$\Sigma = \Phi \Lambda \Phi^T$$

If we normalise the eigenvectors, then the system of vectors Φ forms an orthonormal basis, that is to say:

$$\forall \phi_i \; \phi_j \in \Phi, \; \phi_i \cdot \phi_j = \left\{ \begin{array}{ll} 1 & if \; i = j \\ 0 & if \; i \neq j \end{array} \right.$$

It is in effect an axis system in which we we can represent our data in a compact form.

We can achieve size reduction by choosing to represent our data we fewer dimensions. Normally we choose to use the set of $m(m \leq n)$ eigenvectors of Σ which have the m largest eigenvalues. Typically for face recognition system m will be quite small (around 20-50 in number). We can compose these in an $n \times m$ matrix $\Phi_{pca} = [\phi_1, \phi_2, \phi_3 \cdots \phi_m]$ which performs the PCA projection. For any given image $\boldsymbol{p_x} = (i_1, i_2, i_3, ... i_n)$ we can find a corresponding point in the PCA space by computing

$$\boldsymbol{p_\phi} = (\boldsymbol{p_x} - \boldsymbol{\mu_x}) \cdot \Phi_{pca}$$

The m-dimension vector $\boldsymbol{p_\phi}$ is all we need to represent the image. We have achieved a massive reduction in data size since typically n will be at least 16K and m as small as 20. We can store all our data base images in the PCA space and can easily search the data base to find the closest match to a test image. We can also reconstruct any image with the inverse transform:

$$\boldsymbol{p_x} = \boldsymbol{p_\phi} \cdot \Phi_{pca}^T + \boldsymbol{\mu_x}$$

It can be shown that choosing the m eigenvectors of Σ that have the largest eigenvalues minimises the mean square reconstruction error over all choices of m orthonormal bases.

Clearly we would like m to be as small as possible compatible with accurate recognition and reconstruction, but this problem is data dependent. We can make a decision on the basis of the amount of the total variance accounted for by the m principal components that we have chosen. This can be assessed by looking at the eigenvalues. Let the sum of all the n eigenvalues be written $\sum_{j=1}^{n} \lambda_j$. (The Σ denoting summation in this case, not co-variance.) We can express the percentage of the variance accounted for by the ith eigenvector as:

$$r_i = 100 \times \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

We can then choose m to meet a heuristic condition that will be application dependent. For example, we could ensure that we account for a minimum percentage of the total variance, say 95%, by making $\sum_{j=1}^{m} r_j \geq 95$. Alternatively we could remove all eigenvectors whose eigenvalues account for less than 1% of the total variance. Unfortunately there is no definitive rule for deciding how to choose the number of eigenvectors to retain.

Few samples and many variables

Typically in face recognition we have a very large number of variables n (16K) but far fewer images N. This means that although the dimension of the covariance matrix is $n \times n$ its rank can be at most $N-1$, and consequently the number of non zero eigenvalues will be limited to at most $N-1$. The rank may well be less if the data points are not linearly independent. In the extreme case of the data points all being co-linear, there is just one eigenvector with a non zero eigenvalue, which is in the direction of the line joining the points, and the rank of the covariance matrix will be 1.

Since the number of variables n is very large, the calculation of the eigenvectors of Σ is problematic. We can save considerable amounts of computation time in finding these $N-1$ non-zero eigenvalues by employing a trick first published by Karhunen and Loeve.

We noted above that the covariance matrix could be written in terms of the mean centred data matrix as:

$$\Sigma = \boldsymbol{U}^T \boldsymbol{U} / (N - 1)$$

and has dimension $n \times n$. However, the matrix $\boldsymbol{U}\boldsymbol{U}^T$ has the much lower dimension of $N \times N$. Suppose we calculate the eigenvectors of $\boldsymbol{U}\boldsymbol{U}^T$. These satisfy the equation:

$$\boldsymbol{U}\boldsymbol{U}^T \Phi' = \Phi' \Lambda$$

If we multiply both sides of this equation by $\boldsymbol{U}^T$ we get:

$$\boldsymbol{U}^T \boldsymbol{U}\boldsymbol{U}^T \Phi' = \boldsymbol{U}^T \Phi' \Lambda$$

Adding brackets to this equation:

$$\boldsymbol{U}^T \boldsymbol{U}(\boldsymbol{U}^T \Phi') = (\boldsymbol{U}^T \Phi') \Lambda$$

makes it clear to see that $\boldsymbol{U}^T \Phi'$ are the eigenvectors of $\boldsymbol{U}^T \boldsymbol{U}$, which is what we wanted to find. Since $\boldsymbol{U}\boldsymbol{U}^T$ has dimension $N \times N$ and is a considerably smaller matrix than $\boldsymbol{U}^T \boldsymbol{U}$, we have solved a the much smaller eigenvector problem, and can calculate the $N-1$ eigenvectors of $\boldsymbol{U}^T \boldsymbol{U}$ with one further matrix multiply. $\boldsymbol{U}\boldsymbol{U}^T$ has at most $N-1$ eigenvectors, but, as noted above, all other eigenvectors of $\boldsymbol{U}^T \boldsymbol{U}$ have zero eigenvalues. The resulting eigenvectors are not orthonormal, and so should be normalised.

The method of PCA is sometimes also called the Karhunen-Loeve transform, and occasionally the Householder transform. It is a specific application of the general mathematical technique called single value decomposition in which an $n \times m$ matrix is projected to a diagonal form. It is closely related to the technique called single value decomposition which is a more general diagonalistion method that applies to matrices that are not square.

Correspondence in PCA

In 2D face images the pixels are not usually in correspondence. That is to say a given pixel $[x_i, y_i]$ may be part of the cheek in one image, part of the hair in another and so on. This causes a lot of problems in face recognition and also in reconstructing faces. This means that any linear combination of eigenfaces does not represent a true face but a composition of face parts. A true face is only created when the eigenfaces are added together in exactly the right proportions to re-create one of the original face images of the training set.

However, if we represent a face in 3D, then it is possible to establish a correspondence between each point on the surface map. We can organise the data in such a way that each anatomical location (for example the tip of the nose) has the same index in each different face data set. Figure **??** shows two different faces whose surface points are in correspondence.
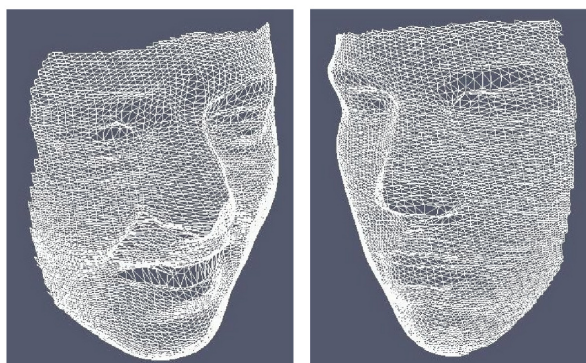

Figure 4: 3D surfaces in correspondence

PCA can be carried out on 3D surface maps of faces (and other anatomical structures). The variables are the individual co-ordinates of the surface points. A data set is of the form:

$$[x_1, y_1, z_1, x_2, y_2, z_2, x_3, \cdots x_N, y_N, z_N]$$

The face maps in figure **??** have about 5,000 surface points, hence the number of variables is 15,000.

If the points of each subject are in correspondence, and the different subjects are aligned as closely as possible in 3D before calculating the PCA, then any reasonable combination of the eigenvectors will represent a valid face. This means that PCA on 3D surface maps has the potential for constructing and manipulating faces that are different from any example in the training set. This has many of applications in film and media. The set of eigenvectors representing surface data in this manner is called an active shape model

Texture can be mapped onto a 3D surface map to give it a realistic face appearance. The texture values (like pixel values in 2D) can also be included as variables in PCA. Models of this sort are called active appearance models.