Machine Learning and Pattern Recognition
Principal Component Analysis

Course Lecturer:Amos J Storkey[*]
Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh
10 Crichton Street, Edinburgh UK
a.storkey@ed.ac.uk
Course page : `http://www.inf.ed.ac.uk/teaching/courses/mlpr/`

[*] Notes are edited versions of notes originally created by David Barber

1 High Dimensional Data

Often in machine learning, the data is very high dimensional. In the case of the hand-written digits, the data is 784 dimensional. Images are a good example of high dimensional data, and a good place where some of the basic motivations and assumptions about machine learning come to light. For simplicity, consider the case of the handwritten digits in which each pixel is binary – either 1 or 0. In this case, the total possible number of images that could ever exist is $2^{784} \approx 10^{236}$ – this is an extremely large number (very much larger than the number of atoms in the universe). However, it is clear that only perhaps at most a hundred or so examples of a digit 7 would be sufficient (to a human) to understand how to recognise a 7. Indeed, the world of digits must therefore lie in a highly constrained subspace of the 784 dimensions. It is certainly not true that each dimension is independent of the other in the case of digits. In other words, certain directions in the space will be more important than others for describing digits. This is exactly the hope, in general, for machine learning – that only a relatively small number of directions are relevant for describing the true process underlying the data generating mechanism. That is, any model of the data will have a relatively low number of effective degrees of freedom. These lower dimensional independent representations are often called 'feature' representations, since it is these quintessential features which succinctly describe the data.

A hook for machine learning

Features

Linear Dimension Reduction

In general, it seems clear that the way dimensions depend on each other is, for a general machine learning problem (and certainly the digits data) very complex – certain dimensions being 'on' means that others are likely to be 'off'. This suggests that non-linear effects will, in general, be important for the efficient description of data. However, finding non-linear representations of data is numerically difficult. Here, we concentrate on linear dimension reduction in which a high dimensional datapoint $\mathbf{x}$ is represented by $\mathbf{y} = \mathbf{F}\mathbf{x}$ where the non-square matrix $\mathbf{F}$ has dimensions $dim(\mathbf{y}) \times dim(\mathbf{x})$, $dim(\mathbf{y}) < dim(\mathbf{x})$. The matrix $\mathbf{F}$ represents a linear projection from the higher dimensional $\mathbf{x}$ space to the lower dimensional $\mathbf{y}$ space. The form of this matrix determines what kind of linear projection is performed and, classically, there are several popular choices. The two most popular correspond to Principal Components Analysis (PCA) and Linear Discriminants. The first is an unsupervised and the latter a supervised projection. We concentrate in this chapter on the more generic PCA, leaving linear discriminants to a later chapter. Note that, again, these methods do not describe any model from which we could generate data and, are also non-probabilistic. However, probabilistic data generating versions do exist which are model based but are beyond the scope of this course.

2 Principal Components Analysis

If data lies in a high dimensional space, we might hope that it lies close to a hyperplane, as in Figure 1. We then can approximate each data point by using the vectors that span the hyperplane alone. I will sometimes refer to this small set of vectors as the "basis" set. Strictly speaking this is not a basis for the whole space, rather is is a 'basis' which approximately spans the space where the data is concentrated. Effectively, we are trying to choose a more appropriate low dimensional co-ordinate system that will approximately represent the data. Mathematically, we write

$$\mathbf{x} \approx \mathbf{c} + \sum_{i=1}^{M} w_i \mathbf{b}^i \tag{2.1}$$

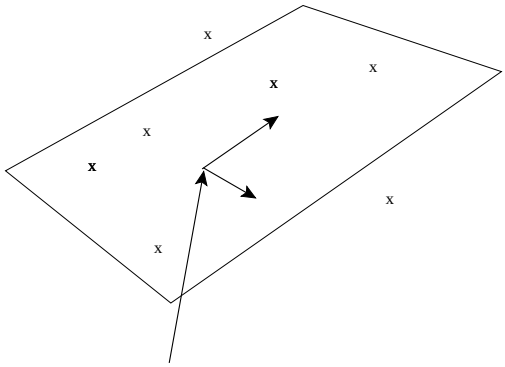

Figure 1: In linear dimension reduction we hope that data that lies in a high dimensional space lies close to a hyperplane that can be spanned by a smaller number of vectors.

The vectors $\mathbf{b}^i, i \in 1, \ldots M$ are chosen to be orthonormal. That is $(\mathbf{b}^i)^T\mathbf{b}^j = 0$ for $i \neq j$, and $(\mathbf{b}^i)^T\mathbf{b}^i = 1$. If the dimension of the data space, $dim(\mathbf{x}) = N$, our hope is that we can describe the data using only a small number M of vectors. If we can do so, we can reduce greatly the information needed to accurately describe the data. For example, if the data lies in a 784 dimensional space, we might hope that we can describe the data accurately using the above linear prescription with a much smaller dimensional representation.

One can show (see end of chapter) that the optimal lower dimensional representation (optimal in the sense of minimal squared reconstruction error) is given by projecting the data onto the eigenvectors of covariance matrix with the largest M eigenvalues. Algorithmically, this is :

1. Find the mean and covariance matrix of the data:

$$\mathbf{m} = \frac{1}{P}\sum_{\mu=1}^{P}\mathbf{x}^\mu, \qquad \mathbf{S} = \frac{1}{P-1}\sum_{\mu=1}^{P}(\mathbf{x}^\mu - \mathbf{m})(\mathbf{x}^\mu - \mathbf{m})^T \quad (2.2)$$

2. Find the eigenvectors $\mathbf{e}^1, \ldots, \mathbf{e}^M$ of the covariance matrix $\mathbf{S}$ which have the largest eigenvalues. Form the matrix $\mathbf{E} = [\mathbf{e}^1, \ldots, \mathbf{e}^M]$ which has the largest eigenvectors as its columns.

3. The lower dimensional representation of each data point $\mathbf{x}^\mu$ is given by $\mathbf{y}^\mu = \mathbf{E}^T(\mathbf{x}^\mu - \mathbf{m})$.

4. The approximate reconstruction of the original datapoint $\mathbf{x}^\mu$ is

$$\mathbf{x}^\mu \approx \mathbf{m} + \mathbf{E}\mathbf{y}^\mu \quad (2.3)$$

5. The total squared error over all the training data made by the approximation is $(P-1)\sum_{j=M+1}^{N}\lambda_j$ where λ_j, $j = M+1 \ldots N$ are the eigenvalues discarded in the projection.

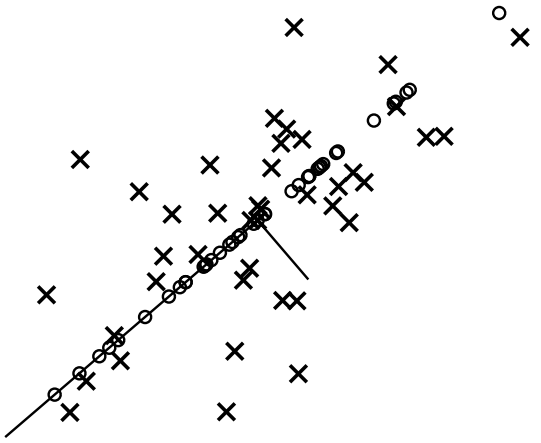

Figure 2: Projection of two dimensional data using one dimensional PCA. Plotted are the original datapoints (crosses) and their reconstructions using 1 dimensional PCA (circles). The two lines represent the eigenvectors and their lengths their corresponding eigenvalues.

One can view the PCA reconstructions (though there is usually little use for these except to check that they give an adequate representation of the original data) as orthogonal projections of the data onto the subspace spanned by the M largest eigenvectors of the covariance matrix, see Figure 2.

```
% PCA demo

p = 40; % number of training points
dimx = 10; % dimension of the data
xdata = randn(dimx,p) + 2*ones(dimx,p);
A = rand(dimx); xdata = A*xdata; % generate some training data, xdata

% perform PCA :
m = mean(xdata,2); x = xdata - repmat(m,1,p);  % subtract the mean
S = cov(x'); % covariance of the data

[Evec,Evalm] = eig(S); Eval = diag(Evalm); % find the e-vals and e-vecs
[evals,index]=sort(Eval); index=flipud(index); % find the largest e-vals
num_retain = 3; % number of eigenvectors to retain
Evec_retained = Evec(:,index(1:num_retain)); % get the largest e-vecs

x_lowerdim = Evec_retained'*x; % lower dimensional representation
x_reconstruction = Evec_retained*x_lowerdim + repmat(m,1,p); % reconstruction of x

subplot(1,2,1); imagesc(xdata); subplot(1,2,2);
imagesc(x_reconstruction);
```

Interpreting the Eigenvectors

Do the eigenvectors themselves explicitly have any meaning? No! They only act together to define the linear subspace onto which we project the data – in themselves they have no meaning. We can see this since, in principle, any basis which spans the same subspace as the eigenvectors of the covariance matrix is equally valid as a representation of the data. For example, any rotation of the basis vectors within the subspace spanned by the first M eigenvectors would also have the same reconstruction error. The only case when the subspace is uniquely defined is when we only use one basis vector – that is, the principal component of the correlation matrix alone.

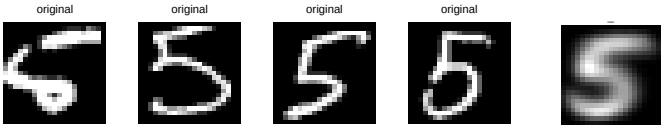

Figure 3: (left) Four of the 892 images. (right) The mean of the 892 images

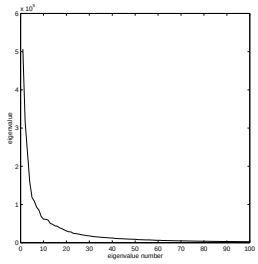

Figure 4: The 100 largest eigevalues

The "intrinsic" dimension of data

How many dimensions should the linear subspace have? As derived (at the end of the chapter), the reconstruction error is dominated by the largest eigenvalues of the covariance matrix. If we plot the eigenvalue spectrum (the set of eigenvalues ordered by decreasing value), we might hope to see a few large values and many small values. Indeed, if the data did lie very close to say a M dimensional linear manifold (hyperplane), we would expect to see M large eigenvalues, and the rest to be very small. This would give an indication of the number of degrees of freedom in the data, or the intrinsic dimensionality. The directions corresponding to the small eigenvalues are then interpreted as "noise".

Warning!

It might well be that a small reconstruction error can be made by using a small number of dimensions. However, it could be that precisely the information required to perform a classification task lies in the "noise" dimensions thrown away by the above procedure (though this will hopefully be rather rare). The purpose of linear discriminants is to try to deal with this problem.

Non-linear Dimension Reduction

Whilst it is straightforward to perform the above linear dimension reduction, bear in mind that we are presupposing that the data lies close to a hyperplane. Is this really realistic? More generally, we would expect data to lie on low dimensional curved manifolds. Also, data is often clustered – examples of handwritten '4's look similar to each other and form a cluster, separate from the '8's cluster. Nevertheless, since linear dimension reduction is so straightforward, this is one of the most powerful and ubiquitous techniques used in dimensionality reduction.

2.1 Example : Reducing the dimension of digits

We have 892 examples of handwritten 5's. Each is a 21*23 pixel image – that is, each data point is a 483 dimensional vector. We plot 4 of these images in Figure 3. The mean of the data is also plotted and is, in a sense, an archetypal 5. The covariance matrix has eigenvalue spectrum as plotted in Figure 4, where we plot only the 100 largest eigenvalues. The reconstructions using different numbers of eigenvectors (10, 50 and 100) are plotted in Figure 5. Note how using only a small number of eigenvectors, the reconstruction more closely resembles the mean image.

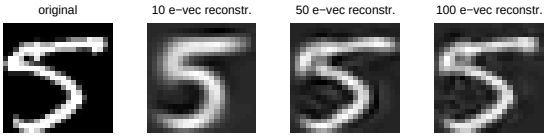

Figure 5: The reconstruction using different linear subspace dimensions

2.2 Mega Dimensional Data

You might be wondering how it is possible to perform PCA on extremely high dimensional data. For example, if we have 500 images each of $1000 \times 1000 = 10^6$ pixels, the covariance matrix will be $10^6 \times 10^6$ dimensional – well beyond the storage capacities of many computers.

One approach around this difficulty is to perform the calculations in a lower dimensional space. Note that there can only be at most P non-zero eigenvalues.

Using $\mathbf{X}$ to denote the (zero mean) data and $\mathbf{E}$ the matrix of eigenvectors – this is non-square since there will be fewer eigenvalues than dimensions. We write the eigenvalues as a diagonal matrix $\mathbf{\Lambda}$. The eigenvalue requirement is

$$\mathbf{X}\mathbf{X}^T\mathbf{E} = \mathbf{E}\mathbf{\Lambda} \tag{2.4}$$

$$\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{E} = \mathbf{X}^T\mathbf{E}\mathbf{\Lambda} \tag{2.5}$$

$$\mathbf{X}^T\mathbf{X}\tilde{\mathbf{E}} = \tilde{\mathbf{E}}\mathbf{\Lambda} \tag{2.6}$$

where we defined $\tilde{E} = \mathbf{X}^T\mathbf{E}$. The last line above represents the eigenvector equation for $\mathbf{X}^T\mathbf{X}$. This is a matrix of dimensions $P \times P$ – in the above example, a 500×500 matrix as opposed to a $10^6 \times 10^6$ matrix previously. We then can calculate the eigenvectors $\tilde{\mathbf{E}}$ and eigenvalues $\mathbf{\Lambda}$ of this matrix more easily. Once found, we then use

$$\mathbf{E} = \mathbf{X}\tilde{\mathbf{E}}\mathbf{\Lambda}^{-1} \tag{2.7}$$

2.3 PCA is good because it is a poor compressor!

A moments thought throws up the following condundrum: It seems that we wish to compress high dimensional data to a lower dimensional representation. However, clearly, the optimal compressed representation retains no structure since, if it did, further compression would still be possible. The goal of feature extraction is not consistent with optimal compression, since we wish to remove some redundancy, yet retain enough structure in the lower dimensional representation such that any further use of the data – making a machine which can generalise from the lower dimensional representations for example – has a chance. Hence, perhaps somewhat perversely, PCA is a reasonable feature extraction method because it is such a poor compressor!

2.4 Regression and PCA

We discussed using PCA to reduce the dimensionality of data, based on the idea that data may lie close to a low dimensional hyperplane. Since a line is a low dimensional hyperplane, one may wonder what the difference is between using PCA to fit a line and a regression approach. The answer is that the objective functions are different. Regression finds a line that minimizes the vertical distance between a datapoint and the line; PCA finds a line that minimizes the distance between a datapoint and the line – see fig(6).

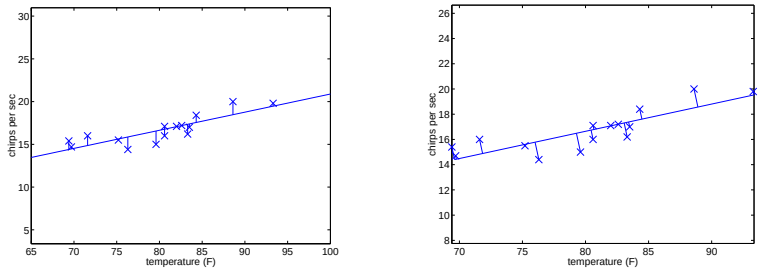

Figure 6: Left: Straight line regression fit to the cricket data. Right: PCA fit to the data. In regression we minimize the residuals – the fit represents the shortest vertical distances. In PCA the fit minimizes the orthogonal projections to the line.

3 Just for Interest ... Deriving the Optimal Linear Reconstruction

3.1 Optimal reconstruction weights

We wish to represent a data point $\mathbf{x}$ as a linear combination of a (small) number of vectors. If the vectors $\mathbf{c}, \mathbf{b}^i, i \in 1, \ldots M$ are already given, how should we choose the coefficients w_i?

If we want the square difference (in each component) between the approximation and $\mathbf{x}$ to be minimal, this gives an error measure

$$E(\mathbf{w}) = \left(\mathbf{x} - \mathbf{c} - \sum_i w_i \mathbf{b}^i \right)^2 \tag{3.1}$$

Defining $\mathbf{d} \equiv \mathbf{x} - \mathbf{c}$ (3.2)

$$E(\mathbf{w}) = \left(\mathbf{d} - \sum_i w_i \mathbf{b}^i \right)^2 \tag{3.3}$$

$$= \mathbf{d}^T \mathbf{d} - 2 \sum_i w_i \mathbf{d}^T \mathbf{b}^i + \sum_{i,j} w_i w_j (\mathbf{b}^i)^T \mathbf{b}^j \tag{3.4}$$

Since the vectors $\mathbf{b}^i$ are orthonormal, and $\mathbf{d}$ is a constant vector, we can form an equivalent error

$$\tilde{E}(\mathbf{w}) = \sum_i \left\{ -2 w_i \mathbf{d}^T \mathbf{b}^i + (w_i)^2 \right\} \tag{3.5}$$

differentiating with respect to w_j gives immediately that the optimal weight coefficients are given by the projection of the vector $\mathbf{d}$ onto the vectors $\mathbf{b}^i$.

$$w_j = \mathbf{d}^T \mathbf{b}^j = (\mathbf{x} - \mathbf{c})^T \mathbf{b}^j = (\mathbf{b}^j)^T (\mathbf{x} - \mathbf{c}) \tag{3.6}$$

3.2 What is the optimal "basis" set?

Using equation (3.6) above, we have that the optimal reconstruction of a vector $\mathbf{x}^\mu$ is

$$\mathbf{x}^\mu \approx \mathbf{c} + \sum_j \mathbf{b}^j (\mathbf{b}^j)^T (\mathbf{x}^\mu - \mathbf{c}) \tag{3.7}$$

For convenience, define $\mathbf{M} = \sum_j \mathbf{b}^j (\mathbf{b}^j)^T$. If we have a set of vectors, $\mathbf{x}^\mu, \mu \in 1, \ldots P$, then the reconstruction error over the whole data set is (defining $\mathcal{B} = \left\{ \mathbf{b}^i, i \in 1, \ldots M \right\}$)

$$E(\mathbf{c}, \mathcal{B}) = \sum_\mu \left(\mathbf{x}^\mu - \mathbf{c} - (\mathbf{x}^\mu - \mathbf{c})^T \sum_j \mathbf{b}^j \mathbf{b}^j \right)^2 \tag{3.8}$$

Defining $\mathbf{A} = \mathbf{I} - \mathbf{M}$, this is

$$E(\mathbf{c}, \mathcal{B}) = \sum_{\mu=1}^{P} \left(\mathbf{A}\left(\mathbf{x}^\mu - \mathbf{c}\right)\right)^2 = P\mathbf{c}^T\mathbf{A}^T\mathbf{A}\mathbf{c} - 2\mathbf{c}^T\mathbf{A}^T\mathbf{A}\sum_\mu \mathbf{x}^\mu + const. \tag{3.9}$$

Differentiating with respect to c_i to find the optimum vector $\mathbf{c}$ gives

$$\mathbf{c} = \frac{1}{P}\sum_{\mu=1}^{P} \mathbf{x}^\mu \tag{3.10}$$

That is, the optimum fixed "origin" for the data is the mean of the data. If we therefore zero mean the data (by subtracting the mean from each data point), then $\mathbf{c} = \mathbf{0}$. Consider we've done this, since this makes the following analysis simpler. We still need to find the optimal vectors $\mathcal{B}$.

A standard textbook approach to finding the optimal basis uses Lagrange multipliers. Here we adopt a different approach, using more elementary methods. It is clear that the eigenvectors of the correlation matrix $\mathbf{S} = \frac{1}{P-1}\sum_{\mu=1}^{P} \mathbf{x}^\mu(\mathbf{x}^\mu)^T$ are spanned by the data $\mathbf{x}^\mu$. Conversely, the data are spanned by the eigenvectors of the correlation matrix. That is, each data point can be represented as a linear combination of the eigenvectors of $\mathbf{S}$,

$$\mathbf{x}^\mu = \sum_{k=1}^{N} \gamma_k^\mu \mathbf{e}^k \tag{3.11}$$

where $\mathbf{e}^k, k \in 1, \ldots N$ are unit length eigenvectors of $\mathbf{S}$. (That is, $\mathbf{S}\mathbf{e}^k = \lambda^k\mathbf{e}^k$, $(\mathbf{e}^k)^T\mathbf{e}^l = 0$ if $k \neq l$ and 1 otherwise). Since the eigenvectors are orthonormal, taking the scalar product of the above equation with $\mathbf{e}^j$ gives $\gamma_j^\mu = (\mathbf{e}^j)^T\mathbf{x}^\mu$, so that any datapoint can be written

$$\mathbf{x}^\mu = \sum_k (\mathbf{e}^k)^T\mathbf{x}^\mu\mathbf{e}^k \tag{3.12}$$

(indeed, this is true for any basis representation of a vector). Using the basis vectors $\mathbf{b}^i$, the residual between $\mathbf{x}^\mu$ and its reconstruction using these basis vectors is

$$\mathbf{r}^\mu = \mathbf{x}^\mu - \sum_j (\mathbf{b}^j)^T\mathbf{x}^\mu\mathbf{b}^j \tag{3.13}$$

Using the above expansion of $\mathbf{x}^\mu$ in terms of the eigenvectors of $\mathbf{S}$ gives

$$\mathbf{r}^\mu = \sum_k \left((\mathbf{x}^\mu)^T\mathbf{e}^k - \sum_j \mathbf{b}^j(\mathbf{b}^j)^T\mathbf{e}^k(\mathbf{x}^\mu)^T \right)\mathbf{e}^k \tag{3.14}$$

$$= \sum_k (\mathbf{x}^\mu)^T\mathbf{e}^k \underbrace{\left(\mathbf{e}^k - \sum_j \mathbf{b}^j(\mathbf{b}^j)^T\mathbf{e}^k\right)}_{\mathbf{f}^k} \tag{3.15}$$

The total reconstruction error for all the data points is then

$$\sum_\mu (\mathbf{r}^\mu)^T\mathbf{r}^\mu = \sum_{\mu,k,j} (\mathbf{e}^k)^T\mathbf{x}^\mu(\mathbf{f}^k)^T(\mathbf{x}^\mu)^T\mathbf{e}^j\mathbf{f}^j \tag{3.16}$$

$$= (P-1)\sum_{j,k} (\mathbf{e}^k)^T\mathbf{S}\mathbf{e}^j(\mathbf{f}^k)^T\mathbf{f}^j = \sum_j \lambda_j(\mathbf{f}^j)^T\mathbf{f}^j \tag{3.17}$$

Thus, if we can make the vectors $\mathbf{f}^j$ zero for the largest eigenvalues λ_j (remember that the eigenvalues of a positive definite matrix are positive), we will make the smallest reconstruction error. If we make $\mathbf{b}^j = \mathbf{e}^j$ for

the largest eigenvalues λ_j, then $\mathbf{f}^j = \mathbf{0}$, and the contribution to the reconstruction error from eigenvalue λ_j will be zero.

Residual Error

If we choose to use only M of the eigenvectors, this is equivalent to a basis expansion in which the first M vectors $\mathbf{b}^j, j = 1, \ldots, M$ are set to the M 'largest' eigenvectors and the rest $\mathbf{b}^j, j = M+1, \ldots, N$ are set to zero. In this case $\mathbf{f}^j = \mathbf{e}^j, j = M+1, \ldots, N$ and the residual error is $(P-1)\sum_{j=M+1}^{N} \lambda_j$.