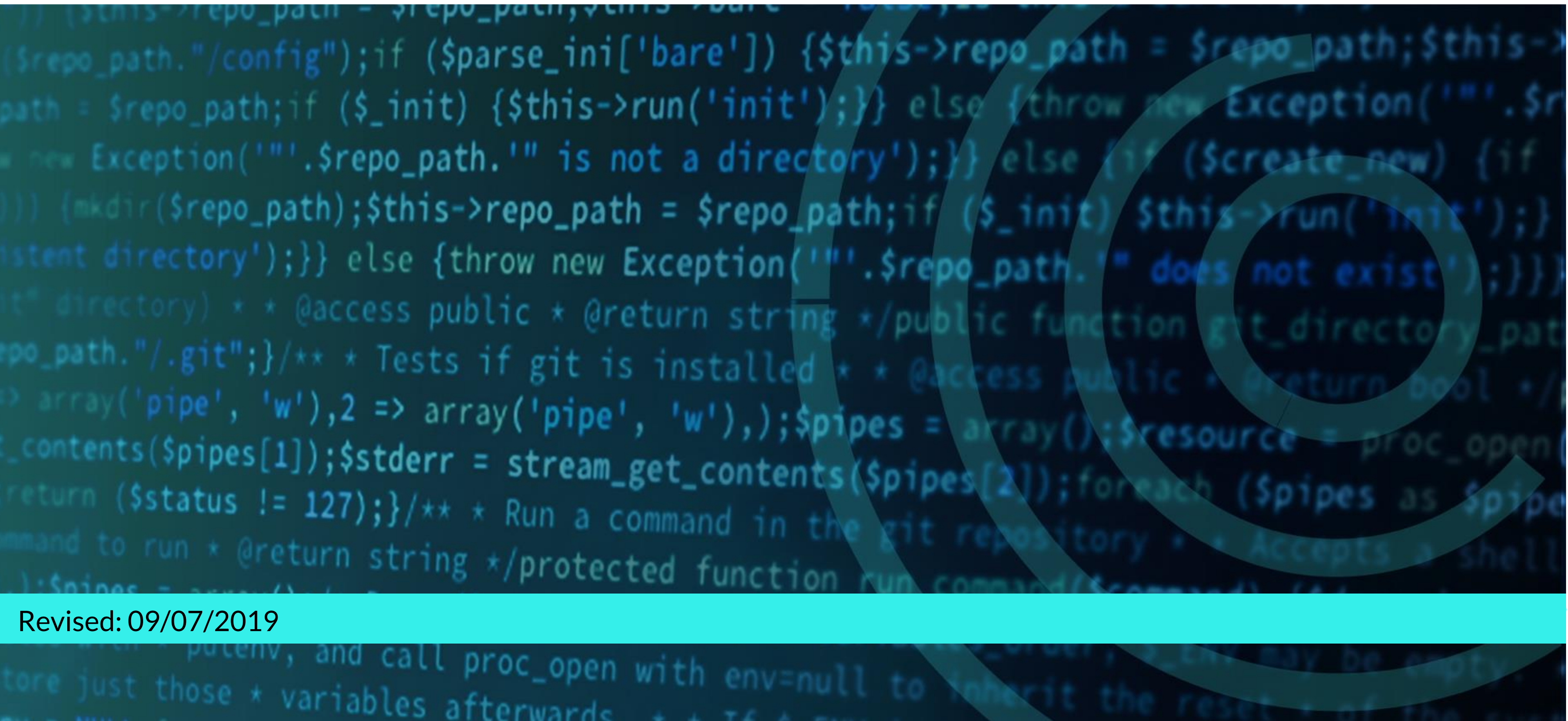
WORLDQUANT UNIVERSITY

Compiled Content Module 2

MScFE 650

Machine Learning in Finance



Revised: 09/07/2019

Table of Contents

Module 1: Dimensionality Reduction

This module will introduce students to two different dimensionality reduction techniques – principal component analysis (PCA) and linear discriminant analysis (LDA). The module starts with the consideration of PCA and its suitability to use-cases, and later shifts its focus to LDA.

Unit 1: Principal Component Analysis (Part 1)

In this set of notes, we expand on some of the ideas introduced in the first lecture video. To briefly recap, in this module, we are studying two basic dimensionality reduction techniques, namely principal component analysis (PCA) and linear discriminant analysis (LDA). In each case the goal is to project the data onto a lower-dimensional space while retaining some of the essential characteristics of the data. Each of the two methods, focus on very different, but specific, characteristics of the data. Let us consider the two approaches separately.

In very general terms, PCA can be characterized as identifying a lower dimensional subspace of a given linear vector space. In order to explain the general idea, we consider a practical application.

In facial recognition, we are interested in facial images, and only facial images. Any image is represented as an there every entry in the matrix (every pixel) represents a shade of gray (the dimension increases three-fold for color images).

It should be clear that an $m \times n$ gray-scale image is a specific point in an $m \times n$-dimensional vector space. This can be very high even for low resolution images.

Since we are interested in only facial images it might just be possible that the facial images occupy lower dimensional subspace of the full $m \times n$-dimensional images space.

In summary, PCA finds lower dimensional subspaces that describe the essential properties of our data. This allows one to project the data onto these lower dimensional subspaces, sometimes leading to significant dimensionality reductions.

LDA is particularly useful in classification problems. We'll discuss different techniques for classification later, here we are interested in preparing the data in order to build more efficient classifiers. In particular, we want to reduce the dimensionality of the data in such a way that we have maximum class separation in the lower dimensional space.

Principal components

Given N observations $x_n \in R^d$, $n = 1, \dots, N$ of dimension d, we want to find the directions of maximum variation in the data. First, we project the data onto a one-dimensional subspace defined by u_1, where $u_1^T u_1 = 1$. Thus x_n is projected onto the vector $(u_1^T x_n)\, u_1$, and in this coordinate system, the projected value is just given by $u_1^T x_n$. The idea is to choose u_1 in such a way that the variance of the projected values is as large as possible. If the sample mean is given by

$$\overline{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \,,$$

then the mean of the projected data us $u_1^T\overline{x}$ and the variance of the projected data is

$$\frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T\overline{x})^2 \;= \frac{1}{N}\sum_{n=1}^{N} u_1^T(x_n - \overline{x})(x_n - \overline{x})^T u_1$$

$$= u_1^T S u_1 \,,$$

where S is the sample covariance,

$$\mathrm{S} = \frac{1}{N}\sum_{n=1}^{N}(\, x_n - \overline{x})(x_n - \overline{x})^T.$$

Since we are only interested in the direction of u_1, we maximize $u_1^T S u_1$ subject to the constraint $u_1^T u_1 = 1$. Introducing a Lagrange multiplier[1] λ, we therefore maximize

$$u_1^T S u_1 + \lambda(1 - u_1^T u_1),$$

subject to $u_1^T u_1 = 1$. Differentiation with respect to u_1 leads to the eigenvalue problem

$$S u_1 = \lambda u_1,$$

and using the constraint gives

$$\lambda = u_1^T S u_1.$$

The projected variance therefore equals the eigenvalue λ of the sample covariance S, and attains its maximum value if we choose the largest eigenvalue $\lambda_{1,}$ with u_1 its corresponding eigenvector. This is our first principal vector. Also, note that the standard deviation — the mean deviation — is given by $\sqrt{\lambda}$.

[1] If you are unfamiliar with Lagrange multipliers, we will be discussing them in more detail in Module 3.

The rest of the principal vectors are obtained by each time projecting onto a vector orthogonal all the previous principal directions. This means that the principal directions are the eigenvectors of the sample covariance matrix, and the eigenvalues are the variances in those directions.

Dimensionality reduction

In the previous section, we identified the principal direction, the eigenvectors of the sample covariance matrix, as well as the variance in these directions, the corresponding eigenvalues, written as:

$$SQ = Q\Lambda \tag{1.1}$$

where Q is the eigenvector matrix of the covariance matrix S and Λ a diagonal matrix with the eigenvalues along its diagonal. For convenience, we arrange that the eigenvalues are ordered in non-increasing order of magnitude, i.e. $\lambda_1 \geq \lambda_2 \ldots \geq$. Also recall that the eigenvector matrix is *orthogonal*, i.e. $Q^T Q = I = QQ^T$. With these formalities taken care of, consider the transformation,

$$y_n = Q^T(x_n - \overline{x}), \;\; n = 1, \ldots, N. \tag{1.2}$$

First, by subtracting $\boldsymbol{\overline{x}}$,we have shifted the origin of the coordinate axes to the mean of the data, and then we have used the orthogonal matrix Q to rotate the coordinate axes around the mean so that the coordinate axes coincide with the principal directions. Thus, the sample covariance of the transformed data is given by

$$\begin{aligned} \mathrm{S_y} &= \frac{1}{N}\sum_{n=1}^{N} \mathrm{y_n} y_n^T \\ &= \frac{1}{N}\sum_{n=1}^{N} \mathrm{Q^T(x_n - \overline{x})(x_n - \overline{x})^T} \, Q \\ &= Q^T \, SQ \\ &= \Lambda. \end{aligned}$$

Since the transformed covariance matrix S_y is diagonal — with the eigenvalues of S along its diagonal — it also means that the different components of the transformed data vectors y are uncorrelated.

At this point, $y \in \mathrm{R}^d$ (and we have not yet achieved any dimensionality reduction). In fact, we have not changed the data at all — we have simply shifted and rotated the coordinate axes.

Let us now form a new matrix Q_ν consisting of the first ν columns of the eigenvector matrix Q, i.e. the columns of Q_ν consist of the n eigenvectors of S belonging to the ν −largest eigenvalues. Note that,

$$Q_\nu^T S Q_\nu = \Lambda_\nu, \tag{1.3}$$

where Λ_ν is an $\nu \times \nu$ – diagonal matrix consisting of the ν – largest eigenvalues of S. It is now straightforward to transform our original data as

$$y_n = Q_\nu^T (x_n - \overline{x}), \;\; n = 1, \dots, N, \tag{1.4}$$

where $y \in R^\nu$, i.e. the data dimensionality is reduced from d to ν. Maybe it will make things clearer if we write down the components y_n of explicitly. First note that the rows of Q^Tconsist of the principal directions,

$$Q^T = \begin{bmatrix} q_1^T \\ \vdots \\ q_v^T \end{bmatrix}.$$

The projection (1.4) can therefore be written as:

$$Y_n = \begin{bmatrix} q_1^T \,(x_n - \overline{x}) \\ \vdots \\ q_v^T \,(x_n - \overline{x}) \end{bmatrix},$$

i.e. the j^{th} component of y_n is given by,

$$y_{jn} = q_j^T (x_n - \overline{x}), \;\; j = 1, \dots, \nu.$$

This is simply the projection of $x_n - \overline{x}$ onto the j^{th} axis q_j.

How much do we lose by reducing the dimensionality in this way? It turns out that the magnitude of the first neglected eigenvalue, i.e. $\lambda_{\nu+1}$, is a natural measure of the amount of information we lose. One can therefore define the relative amount of information that we lose as

$$\frac{\lambda_{\nu+1}}{\sum_j \lambda_j}.$$

Of course, if $\lambda_{\nu+1}$ = 0 the projection is exact.

Recovering the data from its projections

If any information is lost during the projection onto the principal components, it is not possible to reconstruct the original exactly — the lost information cannot be recovered. However, an approximate reconstruction is always possible, and it is exact if there is no information loss during the projection. Keeping in mind that the entries in the vector y_n are the coordinates along the principal directions, the reconstruction is given by,

$$x_n = \overline{x} + Q_\nu \, y_n. \tag{1.5}$$

Alternatively, we can 'solve' for x_n from (1.4), given y_n (and $\overline{x}$). This is an underdetermined system and the generlized solution is given by (1.5).

The whitening transformation

We need not stop with the dimensionality reduction given by (1.3); indeed, it is often convenient to transform the data so that it is spherically distributed. Since this amounts to the identity covariance matrix, the corresponding transformation follows from (1.3):

$$\Lambda_\nu^{-1/2} Q_\nu^T S Q_\nu \Lambda_\nu^{-1/2} = I_\nu, \tag{1.6}$$

where I_ν is the $\nu \times \nu$ identity matrix. Thus, the whitening transformation of the data is given by

$$y = \left(Q_\nu \Lambda_\nu^{-1/2}\right)^T (x - \overline{x}). \tag{1.7}$$

Geometrically, what we do is to rotate the data around the mean so that the coordinate axes are aligned with the principal directions. Then we scale each axis so that the variances along all the coordinate axes are unity.

It might be useful to point out exactly what we have done. First, we use the covariance matrix S to find the transformation (1.6) that transforms it to the identity matrix. Having identified this transformation, it can then be used to transform the data (1.7) so that the covariance matrix of the transformed data is the identity.

Using the Singular Value Decomposition

Although the procedure described above is acceptable, a more elegant alternative is available using the Singular Value Decomposition (SVD). Again, we assume that we have N, d-dimensional observations, $x_j, \;\; j = 1, \dots, N$. We also assume that we have already removed the mean from the data so that $\sum_{j=1}^{N} x_j = 0$. We now collect this data into a single $d \times N$ matrix (array) D with the columns of D consisting of the individual observations

$$D = [x_1 \cdots x_N]$$

It is important that $\mathrm{N} > d$. Can you think why? The SVD of D is given by,

$$D = U\Sigma V^T.$$

Since D is a $d \times N$ matrix, U and V are $d \times d$ and $N \times N$ *orthogonal matrices*, respectively. Σ is a $d \times N$ *diagonal* matrix with the non-zero *singular values* arranged from large to small, on its diagonal:

$$\Sigma = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots \cdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & & \sigma_d & 0 & \cdots & 0 \end{bmatrix},$$

Where $\sigma_j \geq \sigma_{j+1} \geq 0$. Thus, the singular values are *ordered,* from large to small, and all SVD software automatically orders the singular values in this way. In fact, this ordering of the singular values is one of the defining properties of the SVD.Taking into account all the zeros in Σ, the SVD can be written in reduced form,

$$D = U\Sigma_+ V_+^T, \tag{1.8}$$

where U is unchanged (assuming $N > d$), Σ_+ is now the $d \times d$ diagonal matrix obtained from Σ as,

$$\Sigma_+ = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots \cdots & \ddots & \vdots \\ 0 & & \sigma_d \end{bmatrix},$$

and V_+ is an $N \times d$ matrix, consisting of the first d columns of V. This is interesting: the dimension of V_+^T appearing in the reduced SVD is $d \times N$, exactly the same as the data D and it is indeed possible to think of V_+^T as a transformation of the original data D. Exactly what this transformation is, and how it should be interpreted, is discussed next. It will become clear that the three factors in the SVD provide us with important information about the data. Let us now investigate exactly what information was extracted from the data using the SVD. Accordingly, form the products, DD^T and D^TD. Keeping in mind that $UU^T = I = U^TU$, and $VV^T = I = V^TV$, it follows that

$$DD^T = U\Sigma\Sigma^T U^T \textit{ and } D^TD = V\Sigma^T\Sigma V^T$$

so that

$$(DD^T)U = U(\Sigma\Sigma^T) \text{ and } (D^TD)V = V(\Sigma^T\Sigma).$$

Both $\Sigma^T\Sigma$ and $\Sigma\Sigma^T$ are square, diagonal matrices with the squares of the singular values on their diagonals. We now draw the important conclusion that U is the eigenvector matrix of the symmetric matrix DD^T, with eigenvalues the squares of the singular values. V is the eigenvector matrix of the symmetric matrix D^TD, and again its eigenvalues are the squares of the singular values. In order to understand the meaning of all of this, note that DD^T is almost the data covariance matrix that appeared in (1.1). All that is missing is a factor $\frac{1}{N}$ or $\frac{1}{N-1}$, depending on whether we use a biased or unbiased estimate. Specifically, the biased covariance matrix is given by $\frac{1}{N}DD^T$. If we compare with (1.1), the only difference is in the scaling of the eigenvalues. This difference in scaling is quite a nuisance since it makes the interpretation of the singular values not quite obvious. Fortunately, the eigenvectors, U in (1.8) and Q in (1.1), are exactly the same.

The first important thing we learn from SVD is that the principal directions of the data are given by U in (1.8).

We demonstrated that the eigenvalues of the covariance matrix (1.1) are the variances along the principal directions. These eigenvalues differ by the scaling factor $\frac{1}{N}$ from the squares of the singular values.

The second important fact we learn from the SVD is that $\frac{1}{\sqrt{N}}\Sigma_+$ are the standard deviations along the principal directions.

But we are not done yet with the magic of SVD. Let us return to the reduced SVD given by (1.8), but rewritten as

$$U^TD = I\Sigma_+V_+^T.$$

Let us concentrate on the left-hand side, U^TD. Since we multiply the data from the left, it means that we have now transformed the original data D to a new set of data D'.

So, what exactly is this new data D'? Let's calculate its SVD. But wait a minute, we already have it:

$$D' = I\Sigma_+ V_+^T.$$

The principal directions of the transformed data are just the columns of I, the identity matrix. Therefore, the principal directions of the transformed data $D'$are aligned with the coordinate axes. The new data

$$D' = \Sigma_+ V_+^T$$

is therefore just the old data rotated so that its principal directions are aligned with the coordinate axes. And the standard deviations along the principal directions are again given by $\frac{1}{\sqrt{N}}\Sigma_+$.We get this almost for free from the SVD.

Let us continue and transform D' by dividing with the largest singular value σ_1,

$$\frac{1}{\sigma_1} D' = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\sigma_d}{\sigma_1} \end{bmatrix} V_+^T.$$

This is just a scaled version of D' so that the largest singular value equals one. If we need the largest standard deviation to be one, we still need to *multiply* the right-hand side by$\sqrt{N}$, since we actually need to divide by $\frac{\sigma_1}{\sqrt{N}}$.

This is still not the end of the magic. Let us assume that all the singular values are non-zero so that we can multiply with the inverse of Σ_+ to get

$$\Sigma_+^{-1} D' = V_+^T.$$

Since we can also rewrite this as

$$\Sigma_+^{-1} U^T D = V_+^T,$$

it is clearly another transformation of the original data. In order to see what it means, let us write down its SVD explicitly.

$$\Sigma_+^{-1} U^T D = IIV_+^T.$$

For this data, both the U and Σ matrices are the identities. This means two things:

1 The principal directions are aligned with the coordinate axes.

2. All the singular values are equal to one. This means that the data has been reshaped to be spherical. And if we need the standard deviations to be unity, we need to multiply the right-hand side with $\sqrt{N}$.

This is worth repeating: the factor V_+^T returned by the SVD is just the whitened data. Even now we are not quite done yet. Consider the case where only $r \leq d$ of the singular values are non-zero. Since the singular values measure the spread in specific directions (given by the principal directions), a zero singular value tells us that the spread in that direction is zero. This means that the data is not truly d-dimensional, but rather r-dimensional. r is therefore called the rank of the data, in which case the reduced SVD becomes

$$D = U_r \Sigma_r V_r^T,$$

where U_r is an $d \times r$ matrix consisting of the first r columns of U, V_r, is an $r \times N$ dimensional matrix consisting of the first r columns of V, and Σ_r is the $r \times r$ diagonal matrix consisting of the *nonzero* singular values. You now need to answer the following important questions:

1. What does it mean if we transform the original data as $D' = U_r^T D$?
2. What does it mean if we transform the original data as $\frac{1}{\sigma_1} D' = \frac{1}{\sigma_1} U_r^T D$?
3. What does it mean if we transform the original data as $\Sigma_r^{-1} D' = \Sigma_r^{-1} U_r^T D$?

Note that there is nothing to prevent us from keeping fewer singular values than the rank. Say we keep $\mathrm{v} \leq r$ singular values and form a new dataset,

$$\mathcal{D}_\mathrm{v} = U_\mathrm{v} \Sigma_\mathrm{v} V_\mathrm{v}^T,$$

where U_v is a $d \times \mathrm{v}$ matrix consisting of the first v columns of U, V_v is a $\mathrm{v} \times N$ dimensional matrix consisting of the first ν columns of V, and Σ_v is the $\mathrm{v} \times \mathrm{v}$ diagonal matrix consisting of the ν singular values. In this case $\mathcal{D}_\mathrm{v}$ is an approximation of the original matrix, and the quality of the approximation depends only on $\sigma_{\mathrm{v}+1}$, the first discarded singular value.

If $\mathrm{v} = r$, then $\sigma_{\mathrm{v}+1} = 0$. More precisely, the difference between D and $\mathcal{D}_\mathrm{v}$ is given in the Frobenius norm by,

$$\|D - \mathcal{D}_\mathrm{v}\|_F = \sigma_{\mathrm{v}+1}.$$

The fraction of the original data explained by the ν principal components is given by,

$$\frac{\sigma_\nu}{\sum_{n=1}^{r} \sigma_n}.$$

Unit 2: Principal Component Analysis (Part 2)

Again, in this set of notes we expand on the ideas already introduced. In practice, it is usually a good idea to scale the data in order to ensure that one does not run into numerical instabilities. For example, the data is often scaled so that the variance along the first principal axis is 1. This is achieved by transforming the data according to

$$y_n = \frac{1}{\sqrt{\lambda_1}} Q_r^T (x_n - \overline{x}) \quad n = 1, \dots, N$$

where λ_1 is the largest eigenvalue of the sample covariance matrix. It is in general not a good idea to calculate the eigenvalues of the sample covariance matrix numerically, since this procedure is unstable. More precisely, by calculating the covariance matrix, we lose roughly half the available significant digits, and this can cause problems. An alternative, and more stable way, is to form the data matrix:

$$X = \frac{1}{\sqrt{N}} [\, x_1 - \overline{x} \cdots x_N - \overline{x} \,]$$

and calculate its SVD, $X = U\Sigma V^T$. Let us remind the reader that the singular values of X are the square roots of the eigenvalues of the sample covariance matrix S, i.e. $\sigma_n^2 = \lambda_n$, and the columns of U are the eigenvectors of S. Thus, the singular values σ_n measure the standard deviation along the principal directions; and the eigenvalues λ_n, the variances.

Let us now explain in more detail how dimensionality reduction is achieved. If we choose $r = \operatorname{rank}(X)$, i.e. r equals the number of non-zero singular values, then the first r columns of U form an orthogonal basis for the column space of X.

Since the singular values are the standard deviations along these principal directions, they measure to spread of the data in that direction. A zero singular value implies the absence of any data in that direction. Projecting our data onto the first r columns of U therefore entails no loss of information. It squeezes out the “empty” dimensions.

In practice, a system often benefits from discarding the directions with little data representation, i.e. the directions associated with small singular values. A simple example may be helpful. Suppose we expect that our 2D data has to fall on a straight line. If we calculate the principal components of this data, we certainly expect that the first principal direction should point in the direction of the line. Can you see why it is important to remove the mean? In an ideal world, the second singular value should be zero. But, of course, there are small measurement and possibly other errors.

This means that the data does not fall exactly on the line, with the result that the second singular value measures the extent of the noise — the deviation from the straight line.

The fact that the second singular value is small is probably an indication that we are dealing with noise, in which case it is perfectly reasonable to project the data back onto the space (line) defined by the first principal component. Note that this amounts to calculating the linear least-squares approximation.

Let us now consider the higher dimensional case. The singular values tell us which of the directions can be safely discarded. Let us say for our application that only the first r singular values are significant. We then project the data onto the r dimensional subspace defined by the orthogonal basis consisting of the first r columns of U. Let us call this matrix U_r. Given a data value x, its projection onto U_r is given by z where $x - \overline{x} = U_{rz}$ in a least-squares sense. Since U_r has orthogonal columns, the least-squares solution gives us the projection,

$$z = U_r^T(x - \overline{x}).$$

It sometimes happens that we have so much data that it is just not possible to form the data matrix explicitly as required by the SVD approach. This is often the case in speech processing, for example. In that case, one has no choice but to form the covariance matrix and calculate its eigenvalues and eigenvectors.

This can be done recursively. Suppose we have a set of $n-1$ data values, $x_j,\ \ j = 1, \dots, n-1$, with mean μ_{n-1} and sample covariance S_{n-1}. We now add another value x_n to the set. We want to construct the new mean and covariance that include the new data value, from the known values μ_{n-1} and S_{n-1}. It is left as an exercise to derive the following formulas:

$$\begin{aligned} M_n &= \mu_n + \frac{1}{n}(x_n - \mu_{n-1}) \\ S_n &= \frac{n-1}{n}S_{n-1} + \frac{n-1}{n}(\mu_n - \mu_{n-1})(\mu_n - \mu_{n-1})^T + \frac{1}{n}(x_n - \mu_n)\frac{1}{n}(x_n - \mu_n)^T \end{aligned}$$

It is interesting to look briefly at the structure of these equations. If it so happens that the new data value equals the previous mean $x_n = \mu_{n-1}$, then one might be tempted to think that this does not add any information about the dataset; the mean, for example, remains unchanged and the final two terms on the right-hand side of the expression for the updates sample covariance become zero. Therefore, the new sample covariance is slightly smaller,

$$S_n = \frac{n-1}{n}S_{n-1}$$

than the previous one. This means that the data is slightly better clustered. One can also see that for large n, the updates become negligible, unless the new data value differs substantially from the previous mean.

Unit 3: Linear Discriminant Analysis (Part 1)

LDA is closely associated with the classification problem where each data value, or observation, is assigned to one of say, k classes. For example, the observation, can be one of the ten digits, $0, 1, \ldots, 9$, and we need to decide which one. We'll return to this problem in more detail in a later module, but it is not too early to point out that, given an observation x, we are interested in its class assignment probabilities. In the case of k classes we, therefore, calculate the k class assignment probabilities:

$$[P(\mathcal{C}_j|x), \ \ j = 1, \ldots, k],$$

where $\mathcal{C}_j$, $\mathrm{j} = 1, \ldots, \mathrm{k}$, indicate the k different classes. The simplest way of constructing the model $P(\mathcal{C}_j|x)$ is to learn it from labeled data $\mathcal{D}$ consisting of N observations, where each observation consists of a data value x_n and a class label t_n.The actual construction of the different models is a topic to be discussed later. In this section, we are interested in manipulating the data so that the classification task becomes easier. In particular, we want to reduce the dimension of the data values x_n, $n = 1, \ldots, N$, in such a way that maximum class separation is obtained in the lower-dimensional space.

Two-class LDA

Given labeled data where each observation $x_n \in R^d$ is provided with a corresponding class label t_n, we first consider the case where we have only two classes $\mathcal{C}_1$ and $\mathcal{C}_2$. Given a d-dimensional observation x_n, it is projected down to one dimension using,

$$y_n = w^T x_n, \ \ n = 1, \ldots, N.$$

w has to be chosen in such a way as to preserve maximum separation between the classes in the projected one-dimensional space. One possibility is to maximize the separation between the projected class means. This is indicated in the figure below.

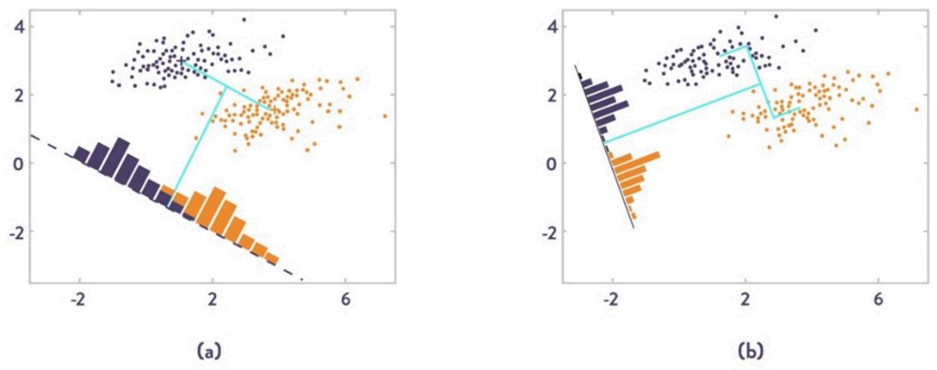


Figure 1: Projection onto the line joining the class means (Bishop, 2006).

Calculating the means of the two classes,

$$m_1 = \frac{1}{N_1} \sum_{x_n \in \mathcal{C}_1} x_n \, , \quad m_2 = \frac{1}{N_2} \sum_{x_n \in \mathcal{C}_2} x_n ,$$

we want to maximally separate the projected class means, i.e. we want to choose w so as to maximize

$$m_2 - m_1 = w^T(m_2 - m_1),$$

where $m_k = w^T \, m_k$, with $w^T w = 1$. Thus we want to maximize $w^T(m_2 - m_1)$ subject to the constraint $w^T w = 1$. Introducing a Lagrange multiplier λ, we therefore maximize

$$L(w) = w^T(m_2 - m_1) + \lambda(w^T w - 1),$$

subject to $w^T w = 1$. From $\nabla_w L = 0$ follows

$$m_2 - m_1 + 2\lambda w = 0.$$

Thus

$$w \propto m_2 - m_1,$$

so that w points in the direction of the vector connecting the two class means.

This implies that the decision boundary is orthogonal to w, see figure (a) above. From this it should be clear that there is considerable class overlap in the projected space. It should be equally clear that we can do better, as illustrated in figure (b).

The problem is to find a mathematical formulation of the desired direction on which to project. It was not a bad idea to maximize the distance between the projected class means; we just need to modify it a little. In addition, we therefore also minimize the total projected within-class scatter. Said differently, we want the projected classes to be as tightly scattered as possible. For this, we need an expression for the within-class scatter for each of the two classes:

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2,$$

$where\ y_n = w^T x_n$. The total within-class scatter is then given by $s_1^2 + s_2^2$. The Fisher criterion is to maximize $J(w)$ where

$$\begin{aligned} J(w) &= \frac{(m_2 - m_1)^2}{s\frac{2}{1} + s\frac{2}{2}} \\ &= \frac{w^T S_B w}{w^T S_W w} \end{aligned}$$

where

$$S_B = (m_2 - m_1)(m_2 - m_1)^T$$

and

$$S_W = \sum_{n \in \mathcal{C}_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in \mathcal{C}_2} (x_n - m_2)(x_n - m_2)^T.$$

Note that we don't need a Lagrange multiplier in this case since a scaling of w does not change the values of $J\,(w) =\ 0$,. We can therefore directly calculate the gradient and set is to zero, $\nabla J(w) = 0$. Doing the algebra, shows that $J(w)$ is maximized when

$$(w^T S_B w) S_W w = (w^T S_W w) S_B w.$$

We are only interested in the direction of $\boldsymbol{w}$, and not its magnitude, so we can simply drop the scalar factors so that $S_W w = S_B w$. This is preferable since $S_B w = (m_2 - m_1)(m_2 - m_1)^T w \propto m_2 - m_1$ we get that

$$w \propto S_W^{-1}(m_2 - m_1).$$

You may have noted that the within-class scatter for each class is not normalized by the number of points in that class. This mean that less weight is attached to classes with fewer members.

Unit 4: Linear Discriminant Analysis (Part 2)

LDA for multiple classes

Given a training set $x_n \in R^d,\ \ n = 1, \dots, N$ consisting of N observations, each of dimension d, and each one with class labels t_n indicating one of k different classes, the idea is to project the data onto a d'-dimensional subspace, through,

$$y_n = W^T x_n,\ \ n = 1, \dots, N, \tag{1.9}$$

where W is an $d \times d'$ matrix, written in terms of its columns as,

$$W = [\boldsymbol{w}_1 \cdots w_{d'}].$$

Our task is to find the W that ensures maximum class separation in the projected space. Formally, one finds a measure $J(W)$ of the class separation. Once $J(W)$ is known, it becomes a matter of finding the optimal W^*,

$$W^* = \arg\max_W J(W).$$

Instead of constructing $J(W)$ explicitly, however, we appeal to geometric intuition. Similarly, as in the two-class problem, we first define the total within — and between-class scatters. The total within-class scatter is defined by

$$S_w = \sum_{n=1}^{k} P_n \Sigma_n \tag{1.10}$$

where Σ_n is the sample class covariance for class n $P_n = \frac{N_n}{N}$, with $N_n$the number of data values belonging to class n.

If m is the mean of the total dataset,

$$m = \frac{1}{N} \sum_{n=1}^{N} x_n = \sum_{n=1}^{k} P_n m_n \tag{1.11}$$

where m_n is the mean of class n, the between-class scatter is defined as:

$$S_b = \sum_{n=1}^{k} P_n(m_n - m)(m_n - m)^T. \qquad (1.12)$$

As in the case of the two-class problem, the idea is to minimize the within-class scatter while simultaneously maximizing the between-class scatter. Geometrically, this is achieved by whitening the within-class scatter, i.e. by finding a transformation W such that $W^T S_w W = I$, while simultaneously diagonalizing the between-class scatter, $W^T S_b W = D_b$, where D_b is a diagonal matrix. (Note: The notation is no accident — as demonstrated below, it turns out that the transformation W that simultaneously diagonalizes both S_w and S_b is exactly what is needed in 1.9).

This proceeds in two steps. First, the within-class scatter is minimized by whitening S_w, i.e. S_w is transformed to the identity matrix. Thus, the within-class scatter is minimized by transforming the data in such a way that the variances along all the coordinate axes become unity. Then we calculate the between-class scatter S_b of the transformed data by applying the same transformation to S_b.

This has the effect that the between-class scatter is exaggerated in the direction of minimum within-class scatter. The final step is to rotate the coordinate axes of the transformed data so that they align with the principle directions defined by the transformed between-class scatter. Since the total within-class scatter of the transformed data is already spherical, it is not changed by a rotation of the axes.

This final rotation, however, identifies the directions of maximum variance of the between-class scatter of the transformed data. These are, therefore, the desired axes onto which the data is projected.

Exercise 1

You can attempt the following exercise to practice what you have just learnt. Please note that this exercise does not count towards your grade and you do not need to submit it.

Is it in general, possible to simultaneously diagonalize more than two covariance matrices?

Let us now make an algorithm out of this procedure:

1 Whiten the within-class scatter S_w, by using the spectral decomposition[2] of S_w,

$$S_w = Q\Lambda Q^T,$$

where Q is the eigenvector matrix of S_w and Λ the corresponding diagonal eigenvalue matrix. We also want to remove the 'empty' dimensions, i.e. the directions in which the variances are zero. For this purpose, we conveniently arrange things[3] so that the eigenvalues are ordered from large to small, $\lambda_n \geq \lambda_{n+1}$. Assuming that the rank of S_w is r (defined as the number of non-zero eigenvalues), define Q_r as the $d \times r$ matrix consisting of the first r columns Q, and Λ_r as the diagonal $r \times r$ matrix with the first r (non-zero) eigenvalues on its diagonal. One can then write

$$S_w = Q_r \Lambda_r Q_r^T.$$

Thus, if S_w is singular, it is whitened by

$$\Lambda_r^{-\frac{1}{2}} Q_r^T S_w Q_r \Lambda_r^{-\frac{1}{2}} = I_r .$$

Remark: It might be informative to apply the transformation directly to the data,

$$z_j = \Lambda_r^{-\frac{1}{2}} Q_r^T \mathrm{x}_j, j = 1, \dots, N. \tag{1.13}$$

The within-class scatter of the transformed data z_j is given by

$$S_w^z = \sum_{n=1}^{k} P_n \Sigma_n^z,$$

with the transformed class scatter,

$$\Sigma_n^z = \frac{1}{N_n} \sum_{j=1}^{N_n} (z_j - \overline{z}_n)(z_j - \overline{z}_n)^T, n = 1, \dots, k, \tag{1.14}$$

[2] If S_w is non-singular, it is computationally more efficient to whiten S_w by factorizing $S_w = LL^T$ by using, for example, the Cholesky factorization, i.e. $L^{-1} S_w L^{-T} = I_d$, where I_d is the $d \times d$ identity matrix.

[3] Although computationally not the most efficient, you may want to consider using the SVD of yourfavorite software library. In this case the SVD is the same as the spectral decomposition. The main advantage is that you don't need to order the eigenvalues; the SVD does it automatically.

and the transformed class mean,

$$\begin{aligned} \overline{z}_n &= \frac{1}{N_n}\sum_{j=1}^{N_n} z_j \\ &= \Lambda_r^{-\frac{1}{2}} Q\frac{T}{r}\left(\frac{1}{N_n}\sum_{j=1}^{N_n} x_j\right) \\ &= \Lambda_r^{-\frac{1}{2}} Q\frac{T}{r}\overline{x}_n \end{aligned}$$

Substituting this expression for $\overline{z}_n$, the expression for z_j given by (1.13) into the expression for (1.14) for the transformed class covariance, it follows that

$$\Sigma_n^z \quad = \Lambda_r^{-\frac{1}{2}} Q_r^T \Sigma_n Q_r \Lambda_r^{-\frac{1}{2}}.$$

Thus,

$$\begin{aligned} S_w^z &= \sum_{n=1}^{k} P_n \, \Sigma_n^z \\ &= \Lambda_r^{-\frac{1}{2}} Q_r^T \left(\sum_{n=1}^{k} P_n \, \Sigma_n^z\right) S_w Q_r \Lambda_r^{-\frac{1}{2}} \\ &= \Lambda_r^{-\frac{1}{2}} Q_r^T \, S_w Q_r \Lambda_r^{-\frac{1}{2}} \\ &= I_r \, . \end{aligned}$$

This shows that the transformation (1.13) derived from the total within-class scatter, transforms the data in such a way that the total within-class scatter of the transformed data is whitened.

2 We now calculate the between-class scatter S_b^z of the transformed data, using the transformation (1.13). Of course, there is no need to calculate the transformed data explicitly since we can directly calculate the transformed between-class scatter,

$$S_b^z = \Lambda_r^{-\frac{1}{2}} Q_r^T S_b Q_r \Lambda_r^{-\frac{1}{2}} .$$

Note that S_b^z is an $r \times r$ matrix with $r = d$ if S_w is non-singular.

3 Since S_b^z is also a scatter matrix, we 'rotate' it so that the principal directions coincide with the coordinate axes. Accordingly, we diagonalize S_b^z using its orthogonal eigenvector matrix, $U_b, S_b^z = U_b D_b U_b^T$ where $U_b U_b^T = I_r = U_b^T U_b$. This means that $U_b^T S_b^z U_b = D_b$,where D_b is the diagonal eigenvalue matrix.

4 Substituting the expression for S_b^z, it follows that

$$U_b^T \Lambda_r^{-\frac{1}{2}} Q_r^T S_b Q_r \Lambda_r^{-\frac{1}{2}} U_b = D_b.$$

It is straightforward to verify that, rotating the transformed data by U_b, leaves the transformed within-class covariance S_w invariant. We get:

$$U_b^T \Lambda_r^{-\frac{1}{2}} Q_r^T S_w Q_r \Lambda_r^{-\frac{1}{2}} U_b = U_b^T I_r U_b = I_r .$$

5 For $W = Q_r \Lambda_r^{-\frac{1}{2}} U_b$ we therefore find that $W^T S_w W = I_r$ and $W^T S_b W = D_b$ as required.

The most important observation is that W is exactly the transformation matrix needed for dimensionality reduction, as alluded to above. We have therefore arrived at an expression for W in (1.9),

$$W = Q_r \Lambda_r^{-\frac{1}{2}} U_b, \tag{1.15}$$

where we use the expression appropriate for squeezing out the 'empty' dimensions corresponding to zero singular values.

Of course, if S_w is not singular, then Q_r and Λ_r are simply the full eigenvector – and eigenvalue matrices, respectively, of S_w.
In summary, the projection is given by $y_n W^T x_n$ where $x_n \in R^d$, where

$$W^T = U_b^T \Lambda_r^{-\frac{1}{2}} Q_r^T.$$

Since Q_r is a $d \times r$ matrix where r is the rank of S_w, and U_b as well as Λ_r are $r \times r$ matrices, W^T is an $r \times d$ matrix. Thus, apart from squeezing out the 'empty' dimensions of S_w, we have not yet achieved any dimensionality reduction.

For that we need to investigate the eigenvalues associated with U_b. Since we want to project down to d' dimensions, we select the eigenvectors in U_b belonging to the d' largest eigenvalues to form an $r \times d'$ matrix $U_{d'}$. The transformation matrix therefore becomes

$$W^T = U_{d'}^T \Lambda_r^{-\frac{1}{2}} Q_r^T. \tag{1.16}$$

Collaborative Review Task

The collaborative review task is designed to test your ability to apply and analyze the knowledge you have learned in the module.

Instructions

- Work through the notebook, answer all questions/problems.
- You are allowed to consult the internet, as well as your peers on the forum.
- Your answers and solutions to the problems should be added to this notebook.
- Submit your final work as an html file.
- Note that Python (version 3.6.4) has been used to calculate the solutions.

Part 1: principal component analysis

Download this notebook to complete the PCA assignment.

Part 2: linear discriminant analysis

Download this notebook to complete the LDA assignment.