1 PCA qualitative

Statistics - A) Joe’s first job as a statistician was to analyse the countries that won medals in the 2012 London Olympic Games. He started his task by collecting data from 84 countries that had won at least one medal. The data’s features (dimensions) for each country (sample/observation) consisted of the number of medals won, the population size (in hundred thousands) and the GDP (in millions of US dollars) all taken in 2012. To start his analysis Joe ran PCA on the data.

1. The first eigenvector explained 99.9% of the data’s variance. Joe was very happy! He was already planning on winning rivers of money by betting on the medals distribution of the 2016 Olympic Games. Assuming that Joe is a competent analyst why did he think he could predict the medals distribution of the next olympic games?

2. Joe’s boss was not as happy. He ran the PCA by himself and he found that the first eigenvector explained only 74.0% of the data’s variance. They compared what they both had done but the only difference they found was that Joe’s boss used the GDP values in billions of dollars while Joe’s GDP values were in millions of dollars. Why did they obtain different results? How should Joe and his Boss have treated the data before the PCA? What was probably the first eigenvector found by Joe?

3. Joe has finally learned the correct way to handle the data before running PCA. He was now trying to run the analysis with extra features (dimensions). He added the countries call prefix code (e.g. 41 for Switzerland) to the last column of the data. Table 1 shows the results that he obtained. How would you interpret these results?

<table>
<thead>
<tr>
<th></th>
<th>Medals</th>
<th>Population</th>
<th>GDP</th>
<th>Codes</th>
<th>Variance Explained</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Eigenvector</td>
<td>-0.619</td>
<td>-0.478</td>
<td>-0.621</td>
<td>-0.051</td>
<td>56%</td>
</tr>
<tr>
<td>2nd Eigenvector</td>
<td>0.132</td>
<td>-0.263</td>
<td>0.149</td>
<td>-0.944</td>
<td>26%</td>
</tr>
</tbody>
</table>

Table 1: Joe’s results.

Note: These results are based on the analysis of real data collected from http://www.medalspercapita.com/#medals-per-capita:2012 However, the countries call prefix code is fake, it was generated from random number generator with uniform distribution.
Solution 1) Being a good analyst Joe would have checked if the data had similar ranges and spreads on all axes. Assuming he had done that, having an eigenvector explaining 99.9% of the data’s variance means that all factors in the analysis were linearly correlated. This means that, if we know the value of one variable, we can perfectly predict the value of all the other dimensions. Because of this Joe was planning on predicting the number of medals each country would get on the 2016 Olympics based on their population and/or GDP in that year.

Solution 2) It seems that Joe was not a good analyst after all. His first eigenvector was able to explain 99.9% of the data’s variance just because the spread of the GDP variable was too high. When his boss ran the same analysis but with a smaller scale for the GDP variable he found an eigenvector that could only explain 74.0% of the data’s variance. If we have a variable that has a much larger spread than the other variables, then the variance of the data along this dimension will be much larger than in the other dimensions. Therefore, the first principal component will be pointing in a direction close to that variable axis, in this case the direction [0,0,1]. To avoid these wrong conclusions, it is good practice to rescale the data to avoid order of magnitude differences across dimensions. This can be easily done by dividing each dimension by the standard deviation of the values of this dimension, normalizing their standard deviations, or simply by doing the eigenvalue decomposition on the correlation matrix as opposed to the covariance matrix.

To get a geometric understanding of this scaling problem imagine that Joe samples data from two normal distributions with equal mean ($\mu_0 = \mu_1 = 0$) but with different scales ($\sigma_0 = 20, \sigma_1 = 1$). So that the two variables are $X \sim \mathcal{N}(0, 20)$ and $Y \sim \mathcal{N}(0, 1)$ (see fig [1]).
The PCA on the unnormalized data will return a principal component that points in the direction of the variable with the biggest scale and it will be able to explain most of the data’s variance. Notice that the two variables are uncorrelated, the covariance matrix is a 2 by 2 diagonal matrix with one value much bigger than the other. If we normalize the data (divide it by its standard deviation) before the PCA, we will not be able to predict the direction of the principal component (there is no preferential direction on uncorrelated data) and that vector will be only able to explain about 50% of the data’s variance.

Note, however, that if you compare data that live in the same dimensions, e.g. when comparing distribution of datapoints in Cartesian space, you should not normalize your data, as the variance along each dimension bear meaningful information which you would loose through a re-scaling.

Re-scaling should be done only when comparing data which have different unity and differ by one or more order of magnitude, e.g. when comparing measurements captured with force sensors (measured in Newton) and measurement from position sensors (measured in mm).

Solution 3) The fact that the two first eigenvectors are able to explain 82% of the data’s variance variables (Table 1) is an indication that there
is a correlation among the original dimensions. To know which variables are correlated, one must look at the coefficients of the eigenvectors on the original dimensions. Taking a look at these, we see that the first principal component (eigenvector) has large negative coefficients in all dimensions but the prefix codes. This reveals a correlation across the first three original dimensions, so across Medals, Population and GDP. One can see this by inspecting figure 2 in which we see a strong correlation between GDP and Population. This is the same for the Medals (not shown here).

There is, indeed, no reason for the countries call prefix code to be correlated with any of the other variables. Therefore, the information given by this variable is independent of the other dimensions and, unsurprisingly, we see that the second eigenvector point along the Codes axis (see figure 3), and that the first eigenvector has no contribution from Codes. One can confirm this by looking at figure 3 which shows the GDP-Codes slice of the data. This fact holds for Codes vs the other remaining dimensions (not shown here).

For curiosity the normalized linear correlations between the variables are the following:

<table>
<thead>
<tr>
<th></th>
<th>Medals</th>
<th>Population</th>
<th>GDP</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medals</td>
<td>1.00</td>
<td>0.47</td>
<td>0.88</td>
<td>0.00</td>
</tr>
<tr>
<td>Population</td>
<td>0.47</td>
<td>1.00</td>
<td>0.48</td>
<td>0.15</td>
</tr>
<tr>
<td>GDP</td>
<td>0.88</td>
<td>0.48</td>
<td>1.00</td>
<td>-0.02</td>
</tr>
<tr>
<td>Codes</td>
<td>0.00</td>
<td>0.15</td>
<td>-0.02</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Figure 2: The first eigenvector (in red) has similar contributions from both the GDP and the population variables. The second eigenvector (in green) is very small when projected to this space because it’s almost orthogonal to it.

Figure 3: There is no correlation between GDP and Codes. As we can see the second eigenvector (green) is nearly completely aligned with the Codes dimension.
Figure 4: We can see that the Codes distribution is uniform whilst the GDP and Population distribution is not.

**Dimension Reduction - B**

1. Jenny has a dataset of 10'000 images of faces. These are all black and white 100 × 200 px images where each pixel can have a value from 0 to 255 (1 byte) to represent the intensity. Because Jenny cares about reducing the dimension of each image but not to find common features across the images, she stores the image in a matrix $X$ where each column is a different image and each row corresponds to each different pixel location. Note: The images are now represented as column vectors with 20'000 (width × height) dimensions.

$$X = \begin{bmatrix}
    x_{1,1} & x_{1,2} & \cdots & x_{1,10'000} \\
    x_{2,1} & x_{2,2} & \cdots & x_{2,10'000} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{20'000,1} & x_{20'000,2} & \cdots & x_{20'000,10'000}
\end{bmatrix}$$

The dataset occupy 200MBs $(1 \text{ byte/pixel}) \times 20'000 \times 10'000 \text{ (images)} = 200 \text{ (MBs)}$ in memory. Jenny needs to compress them because she wants to make the dataset available on her internet server, and, if multiple users download it, she ends up paying a big internet bill. To find the optimal compression, she decides to run PCA on those images. She finds that the first 100 principal components explain 90% of the data’s variance. She also observes that if she projects the images onto the first 100 eigenvectors and discards the other projections, her images loose resolution but remain readable. This project hence might pay off thanks to the generated decrease in
memory space. Has Jenny really gained in memory? And if so, how much has she gained? How much memory does the dataset occupy now?

(Hint: Let’s say each float number takes 8 bytes (64 bits) of space)

**Solution 1)**

There are three solutions to this problem depending on what Jenny aims to do.

**Global Data Compression:**

Jenny considers this as a global dimensional reduction task and opts to use PCA to reduce the dimensionality of all image data set at once.

For doing this, she first subtracts the mean from her data set $X = X - E\{X\}$ such that it is centred and then proceeds to compute the sample covariance matrix $C = E\{XX^T\}$, with $C \in \mathbb{R}^{20'000 \times 20'000}$.

Jenny does an eigen decomposition of $C = V\Lambda V^T$ where $\Lambda$ is a diagonal matrix in which the entries are the eigenvalues and $V$ is a matrix whose columns are the eigenvectors $V = [e^1, ..., e^{20'000}]$, $e^1 \in \mathbb{R}^{10'000 \times 1}$. Jenny’s hope is that she can find a linear projection which can reduce the dimensionality of her data, $Y = AX$ where $A = V^T$.

By inspecting the eigenvalues she finds that a 100 of them explains 90% of her data sets variance. She decides to discard the remaining eigenvectors obtaining the projection matrix $A_p$ for which $p = 100$, ($A_p \in \mathbb{R}^{100 \times 20'000}$).

This implies that there exists a 100 dimensional subspace (given the 20'000 original space) in which the projected data retains 90% of the information. As Jenny wants to simply compress the information and make only the compressed data available through the net, she will project all the images to the 100 dimensional subspace and discard the eigenvectors.

$$Y = A_pX \quad (1)$$

$$Y \in \mathbb{R}^{100 \times 10'000} \quad (2)$$

$$A_p \in \mathbb{R}^{100 \times 20'000} \quad (3)$$

$$X \in \mathbb{R}^{20'000 \times 10'000} \quad (4)$$

After this PCA step the projection matrix $A_p$ and the original data $X$ are no longer needed, leaving Jenny with $Y$. The total amount of memory saved would be $(20'000 \times 10'000) - (8 \times 100 \times 10'000) = 192Mb$. This gives a space saving of $96\% = 192/200$.

**Local Data Compression**

In the first solution, Jenny had assumed that all data shared some common properties and hence that computing an eigenvalue decomposition on
the entire dataset would be beneficial to extract these common properties and remove the distracting "noise" information. This way of proceedings is good only if the dataset is quite homogeneous, e.g. dataset of faces of people, where the face is nicely centered. If the dataset is composed of an heterogeneous dataset with all sorts of images, then, likely there is not much common across the images and the PCA will not yield a very good compression.

In this case, Jenny should rather treat each image separately and apply PCA on each image (as shown in the example of the class’s slides). To compute PCA on each image separately, Jenny must computes 100 covariance matrices $C_i \in \mathbb{R}^{20'000 \times 20'000}$ for $i = 1 : 100$. She would first subtracts the mean from each image $\hat{x}_i = x_i - \mathbb{E}\{x_i\}$ such that it is centred and proceeds to compute the sample covariance matrix $C_i = \mathbb{E}\{x_i x_i^T\}$. Then, for each image, she would select the subset of eigenvectors that yields the smallest reconstruction error. The number of eigenvectors may vary across each image. She would then project each image on its selected set of eigenvectors. The advantage of this procedure is that it yields the optimal tradeoff between dimensionality reduction and image resolution. Often, this process leads to a more efficient data compression of the order of 90%. In other words, each projection would require much less than the 100 eigenvectors Jenny found in the global data compression approach.

However, the drawback is that a) the images end up having a different resolution; b) the process is costly (computing 100 times PCA in place of once).

**Remote Data Un-compression**

Finally, let us consider the case where Jenny is interested in not losing any resolution in her images and allow image processing enthusiasts to recover the images in close to full resolution.

As she also wants to minimise the amount of bandwidth she uses when sending her dataset over the internet, she gathers that it is more efficient to transmit $A_p$, $Y$ and $\mathbb{E}\{X\}$ rather than simply sending the raw data set $X$ (here, she uses the compression obtained in the global compression approach). When the enthusiast receives $Y$, $A_p$ and $\mathbb{E}\{X\}$ he can then reconstruct a lossy version of the original data set of images $\hat{X}$ through mean-square projection.

$$A_p^{-1}Y + \mathbb{E}\{X\} = \hat{X}$$

(5)

The reason we have to add the mean back to the re-projected data is because we originally centred it. If the data was already centred there would be no need to store or add the mean since it would be zero.

Note also that the reason that this is a lossy compression $\hat{X} \neq X$ is because the explained variance with $p = 100$ is not 100%. If this were the
case then there would be lossless compression and $\hat{X} = X$.

Jenny would now have to store the variables $A_p$, $Y$ and $E\{X\}$ which now are represented by floating numbers instead of integers. So each element of the variables will take 8 bytes of space instead of 1 byte.

$$
\begin{align*}
A_p: & \quad (8 \times 100 \times 20'000) = 16 \text{ MBs} \\
Y: & \quad (8 \times 100 \times 10'000) = 8 \text{ MBs} \\
E\{X\}: & \quad (8 \times 1 \times 20'000) = 0.16 \text{ MBs}
\end{align*}
$$

Table 2: Memory requirement per transmitted variable.

The total amount of memory the data set occupies now is 24.16 MBs as oppose to the original 200 MBs giving Jenny 88% compression factor.
2 PCA quantitative (to be done at home)

A) Let us go through through the important steps of PCA:

1. Compute the covariance matrix, $C$, of the dataset composed of the four points shown in the left figure ($(\alpha \alpha), (\alpha - \alpha), (-\beta \beta), (\beta - \beta)$).

**Solution** The empirical covariance matrix is given by

$$ C = \frac{1}{M} \sum_{i=1}^{M} (x^i - \mu)(x^i - \mu)^T $$

$$ = \frac{1}{M} X'X'^T. $$

with $X' = X - \mu$.

Using the values of the given data points we get (as $\mu = 0$)

$$ C = \frac{1}{4} \begin{bmatrix} \alpha & -\alpha & -\beta & \beta \\ \alpha & -\alpha & \beta & -\beta \\ -\alpha & -\beta & \beta & -\beta \\ -\beta & \beta & -\beta \end{bmatrix} \begin{bmatrix} \alpha & \alpha \\ -\alpha & -\alpha \\ -\beta & -\beta \\ \beta & \beta \end{bmatrix} $$

$$ = \frac{1}{2} \begin{bmatrix} \alpha^2 + \beta^2 & \alpha^2 - \beta^2 \\ \alpha^2 - \beta^2 & \alpha^2 + \beta^2 \end{bmatrix} $$

2. Compute the eigenvectors and eigenvalues of $C$. Discuss how the eigenvectors and eigenvalues are affected by $\alpha$ and $\beta$. 
Solution  The eigenvalues $\lambda^i$ and eigenvectors $e^i$ are defined by the following property:

\[ Ce^i = \lambda^i e^i \]

\[ \Rightarrow (C - \lambda^i I) e^i = 0 \]

For a non-zero vector, $e^i$, to satisfy this equation the determinant $| (C - \lambda^i I) | = 0$, meaning that it cannot have an inverse. If it did have an inverse then $v = 0$ which is the degenerate case.

\[
\begin{vmatrix}
\frac{1}{2} \alpha^2 + \frac{1}{2} \beta^2 - \lambda^i & \frac{1}{2} \alpha^2 - \frac{1}{2} \beta^2 \\
\frac{1}{2} \alpha^2 - \frac{1}{2} \beta^2 & \frac{1}{2} \alpha^2 + \frac{1}{2} \beta^2 - \lambda^i \\
\end{vmatrix} = 0
\]

\[ \Rightarrow \left( \frac{1}{2} \alpha^2 + \frac{1}{2} \beta^2 - \lambda^i \right)^2 - \left( \frac{1}{2} \alpha^2 - \frac{1}{2} \beta^2 \right)^2 = 0 \]

\[ \Rightarrow \lambda^2 + \lambda^i (\alpha^2 + \beta^2) + (\alpha^2 \beta^2) = 0 \]

\[ \Rightarrow \lambda^i = \frac{(\alpha^2 + \beta^2)^2 \pm (\alpha^2 - \beta^2)}{2} \]

Now the eigenvectors are simply the solutions to $(C - \lambda^i I) e^i = 0$. For $\lambda^1$

\[
\frac{1}{2} \begin{bmatrix}
-\alpha^2 + \beta^2 & \alpha^2 - \beta^2 \\
\alpha^2 - \beta^2 & -\alpha^2 + \beta^2 \\
\end{bmatrix}
\begin{bmatrix}
e^i_1 \\
e^i_2 \\
\end{bmatrix} = \begin{bmatrix}0 \\
0 \\
\end{bmatrix}
\]

\[ \Rightarrow e^i_1 = e^i_2 \]

Additionally, the eigenvectors must be normalized (i.e $||e^i|| = 1$)

\[ \sqrt{e^i_1^2 + e^i_2^2} = 1 \]

\[ \Rightarrow e^i = \begin{bmatrix}
\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} \\
\end{bmatrix}^T \]

For $\lambda^2$

\[
\frac{1}{2} \begin{bmatrix}
\alpha^2 - \beta^2 & \alpha^2 - \beta^2 \\
\alpha^2 - \beta^2 & \alpha^2 - \beta^2 \\
\end{bmatrix}
\begin{bmatrix}
e^i_1 \\
e^i_2 \\
\end{bmatrix} = \begin{bmatrix}0 \\
0 \\
\end{bmatrix}
\]

\[ \Rightarrow e^i_1 = -e^i_2 \]

\[ \Rightarrow e^i = \begin{bmatrix}
\frac{1}{\sqrt{2}} \\
-\frac{1}{\sqrt{2}} \\
\end{bmatrix}^T \]

The eigenvectors are unaffected by $\alpha$ and $\beta$, the direction of the data’s spread remains the same regardless of their value. The size of the eigenvalues is proportional $\alpha^2$ and $\beta^2$. This is due to the increase of the spread of the data.
3. Compute the projection of the dataset on the principal directions. Then reduce the dimensionality of the dataset by discarding the less relevant dimension in the projected space (assuming $\alpha > \beta$).

**Solution**  Each data point is projected by multiplying its vector with the projection matrix $A$, which is built by piling up the eigenvectors:

$$ y^i = Ax^i = \begin{bmatrix} e_1^T \\ e_2^T \end{bmatrix} x^i $$

For the first point $(\alpha, \alpha)$:

$$ y^1 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} = \begin{bmatrix} \sqrt{2} \alpha \\ 0 \end{bmatrix} $$

The same procedure applied to the other points yields:

$$ y^2 = \begin{bmatrix} -\sqrt{2} \alpha \\ 0 \end{bmatrix} \quad y^3 = \begin{bmatrix} 0 \\ -\sqrt{2} \beta \end{bmatrix} \quad y^4 = \begin{bmatrix} 0 \\ \sqrt{2} \beta \end{bmatrix} $$

The less relevant dimension is the dimension given by the eigenvector with the smallest eigenvalue, which in our case is $e_2$:

This yields:

$$ y^1 = [\sqrt{2} \alpha] \quad y^2 = [-\sqrt{2} \alpha] \quad y^3 = [0] \quad y^4 = [0] $$

4. From the reduced dataset, reconstruct an approximation of the dataset in the original referential. Compute the average reconstruction error

$$ E = \frac{1}{M} \sum_{i=1}^{M} ||x^i - \hat{x}^i||^2 $$

where $x^i$ are the original data points and $\hat{x}^i$ their approximation, and discuss its relation to the eigenvalues of $C$.

**Solution**  We use the transpose of the reduced projection matrix to project the reduced dataset back to its original space

$$ \hat{x}^i = A_{red}^T y^i = \begin{bmatrix} e_1^T \end{bmatrix}^T y^i $$
For the first data point this is:

\[ \hat{x}^1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{\sqrt{2}}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{2} \\ \alpha \end{bmatrix} = \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} \]

The other reconstructed points are:

\[ \hat{x}^2 = \begin{bmatrix} -\alpha \\ -\alpha \end{bmatrix} \quad \hat{x}^3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \hat{x}^4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

The average reconstruction error is \( \frac{1}{M} \sum_{i=1}^{M} ||x^i - \hat{x}^i||^2 = \beta^2 \), which is equal to the eigenvalue of the discarded projection dimension.

5. For the data from the right figure, what structure (isotropic, diagonal, full, symmetric) does the covariance matrix have?

**Solution**  The covariance matrix is diagonal, it is aligned with the axis and as all covariance matrices are, it is symmetric. Its form would be diagonal with unequal values at the diagonal.

**B)** Show that when an N-dim set of data points \( X \) is projected onto the eigenvectors \( V = [e^1, \cdots, e^N] \) of its covariance matrix, \( C = XX^T \), the covariance matrix \( YY^T \) of the projected data \( Y \) is diagonal and hence that, in the space of the eigenvector decomposition, the distribution of \( X \) is uncorrelated.

**Solution**

\[
YY^T = V^T X \left( V^T X \right)^T \\
= V^T X X^T V \\
= V^T \Lambda V^T V \\
= \Lambda \Lambda \\
= \Lambda 
\]

\( \Lambda \) is a diagonal matrix composed of the eigenvalue of \( XX^T \).