Assigment 1. Advanced machine learning.

Author:
Rodrigo Castellano Ontiveros
roco@kth.se

Question 1.1.1: Explain why this data-centering step is required while performing PCA. What could be an undesirable effect if we perform PCA on non-centered data?

So far, we have performed PCA minimizing the reconstruction error. To do so, we have applied SVD. On the other hand, we could also perform PCA by maximizing the preserved variance and decorrelation, doing eigenvector decomposition (EVD). When doing SVD we work on the data, whereas with EVD we work with the covariance matrix. We have to center the data to calculate the covariance matrix [1].

In centered PCA, the principal components are uncorrelated linear combinations of the variables, which are colum-centered. In this case, the principal components maximize variance, whereas uncentered principal components are linear combination of variables that are uncentered, and maximize non-central second moments, subject to having their crossed-non central second moments equal to zero. Then ,the eigenvalues of the uncentered principal components indicate the their non-central second moments instead of the variance Non centering might imply that the components are not decorrelated anymore and the mean will be reflected in the first component, instead of the variance [1].

Question 1.1.2: Does the previous argument imply that a single SVD operation is sufficient to perform PCA both on the rows and the columns of a data matrix?

The previous argument implies that a single SVD operation is not sufficient to perform PCA both on the rows and the columns of a data matrix. This happens because the centering of a matrix based on its rows is different than the centering of a matrix based on its colums. Therefore, although SVD for a matrix gives also the SVD for its transpose, when we perform PCA, the matrices that we do SVD on, are not the transpose of each other.

Question 1.1.3: Explain why the use of the pseudo-inverse is a good choice to obtain the inverse mapping of the linear map

In PCA we use the coding and decoding functions. For the decoding, the linear operator is W. To perform the inverse mapping the best way to do it is by using W^+, given that $W \in \mathbb{R}^{dxk}$ doesn't have an inverse matrix ($k < d$), and the pseudoinverse W^+ is unique [2]. There is the advantage that when performing PCA, the pseudoinverse of W is computed by

$$W^+ = \left(W^T W\right)^{-1} W^T = W^T \tag{1}$$

where $W^TW = I_k$ because W has orthonormal colums. This is very convenient to compute.

Question 1.1.4: Derive PCA using the criterion of variance maximization and show that one gets the same result as with the criterion of minimizing the reconstruction error. Show this result for projecting the data into k dimensions, not just 1 dimension.

As shown in [2], PCA can be performed using two approaches: minimal reconstruction error and maximal preserved variance and decorrelation. Both of them lead to the same result, provided the data is centered.

i) Minimal reconstruction error. In this approach the reconstruction mean square error E_{codec} is minimized.

Consider the dataset (already centered) $\mathcal{Y} = \{y_1, \ldots, y_n\}$, represented as matrix $Y = (y_1, \ldots, y_n) \in \mathbb{R}^{d \times n}$, where a data point is represented by a vector $y = (y_1, \ldots, y_d)^T \in \mathbb{R}^d$. Also, consider a linear transformation $W \in \mathbb{R}^{dxk}$ with orthonormal colums, where d is the number of features, n is the number of datapoints, and k is the number of latent variables.

$$\begin{aligned}
E_{\text{codec}} &= E_y \left\{ \left\| y - WW^Ty \right\|_2^2 \right\} \\
&= E_y \left\{ \left(y - WW^Ty \right)^T \left(y - WW^Ty \right) \right\} \\
&= E_y \left\{ y^Ty - 2y^TWW^Ty + y^TWW^TWW^Ty \right\} \\
&= E_y \left\{ y^Ty - 2y^TWW^Ty + y^TWW^Ty \right\} \\
&= E_y \left\{ y^Ty - y^TWW^Ty \right\} \\
&= E_y \left\{ y^Ty \right\} - E_y \left\{ y^TWW^Ty \right\}
\end{aligned} \tag{2}$$

Therefore, finding the minimum of E_{codec} is equivalent to finding the maximum of $E_y\{y^TWW^Ty\}$, since $E_y\{y^Ty\}$ is a constant term, given y. We can make use of the sample mean to approximate the previous expression:

$$\begin{aligned}
E_y \left\{ y^TWW^Ty \right\} &\approx \frac{1}{n} \sum_{\text{i}=1}^{n} (y(i))^T WW^T (y(i)) \\
&\approx \frac{1}{n} \operatorname{tr} \left(Y^TWW^TY \right)
\end{aligned} \tag{3}$$

Then, SVD can be calculated for Y, such that $Y = V\Sigma U^T$, ordering the singular values from greater to smaller. Substituting this in expresion (3), the problem now is to find

$$\arg\max_W \operatorname{tr} \left(U\Sigma^TV^TWW^TV\Sigma U \right) = VI_{d \times k} \tag{4}$$

The minimum is achieved by setting the k colums of W colinear with the colums of V, i.e.,

$$W = I_{kxd}V^T \tag{5}$$

leading to the expression to find the projection

$$x = I_{k \times d} V^T y \tag{6}$$

ii) Maximal preserved variance and decorrelation. In this case, we assume that the latent variables in x are uncorrelated, and x is centered. We can calculate the covariance matrix as $C_{xx} = E\{xx^T\}$, which is diagonal if x is centered. Taking into account that C_{yy} is known (otherwise it can be calculated with the sample variance $C_{yy} = \frac{1}{n} yy^T$) and PCA in the model works, then

$$\begin{aligned} C_{yy} &= E\left\{yy^T\right\} \\ &= E\left\{Wxx^T W^T\right\} \\ &= WE\left\{xx^T\right\} W^T \\ &= WC_{xx}W^T \end{aligned} \tag{7}$$

Therefore, by multiplying on both sides by W^T and W, the variance of the projection is

$$C_{xx} = W^T C_{yy} W \tag{8}$$

By applying EVD to C_{yy}, we get $C_{yy} = V \Lambda V^T$ [2].

The goal is to maximize the variance of the projecyed data. Some results have to be shown first. The results are taken from New York University (NYU) lectures [3].

The first property to check is that a quadratic function has a maximum in the unit sphere. When dealing with a covariance matrix, this would mean the direction of maximum variance.

Lemma 0.1. *For any symmetric matrix $A \in \mathbb{R}^{d \times d}$, there exists a vector $u_1 \in \mathbb{R}^d$ such that*

$$u_1 = \arg \max_{\|x\|_2 = 1} x^T A x$$

We can apply the extreme value theorem, which says that a continuous function in a compact set attains its extreme values. In our case the unit sphere is compact and the quadratic function $x^T A x$ is continuous.

Lemma 0.2. *(The maximum and minimum are attained at eigenvectors). For any symmetric* matrix *$A \in \mathbb{R}^{d \times d}$, a vector $u_1 \in \mathbb{R}^d$ such that*

$$u_1 = \arg \max_{\|x\|_2 = 1} x^T A x$$

is an eigenvector of A. There exists $\lambda_1 \in \mathbb{R}$ such that

$$A u_1 = \lambda_1 u_1$$

so that

$$\lambda_1 = \max_{\|x\|_2 = 1} x^T A x$$

Proof. We decompose Au_1 into two orthogonal components, one in the direction of u_1 and the other one in the tangent plane to the sphere,

$$\begin{aligned} Au_1 &= u_1^T Au_1 u_1 + x_\perp \\ x_\perp &:= Au_1 - u_1^T Au_1 u_1 \end{aligned}$$

Our goal is to show that if u_1 attains the maximum then $x_\perp$ must be zero. Let for some $\epsilon > 0$ to be chosen later. The vector $x_\perp$ is orthogonal to u_1 so by Pythagoras' theorem

$$\begin{aligned} \|y\|_2^2 &= \|u_1\|_2^2 + \epsilon^2 \|x_\perp\|_2^2 \\ &= 1 + \epsilon^2 \|x_\perp\|_2^2 \end{aligned}$$

We now show that unless $x_\perp$ is zero, normalizing y yields a unit-norm vector that achieves a value larger than $f(u_1)$:

$$\begin{aligned} \left(\frac{y}{\|y\|_2}\right)^T A \frac{y}{\|y\|_2} &= \frac{y^T Ay}{\|y\|_2^2} \\ &= \frac{u_1^T Au_1 + 2\epsilon x_\perp^T Au_1 + \epsilon^2 x_\perp^T Ax_\perp}{1 + \epsilon^2 \|x_\perp\|_2^2} \\ &= \frac{u_1^T Au_1 + 2\epsilon \|x_\perp\|_2^2 + \epsilon^2 x_\perp^T Ax_\perp}{1 + \epsilon^2 \|x_\perp\|_2^2} \\ &= u_1^T Au_1 + \frac{2\epsilon \|x_\perp\|_2^2 + \epsilon^2 \left(x_\perp^T Ax_\perp - \|x_\perp\|_2^2 u_1^T Au_1\right)}{1 + \epsilon^2 \|x_\perp\|_2^2} \\ &> u_1^T Au_1 \end{aligned}$$

if we choose

$$\epsilon < \frac{2 \|x_\perp\|_2^2}{\left|x_\perp^T Ax_\perp\right| + \|x_\perp\|_2^2 \left|u_1^T Au_1\right|}$$

$\square$

With these lemmas we can show the final result by induction.

Theorem 0.3. *(Spectral theorem for symmetric matrices). If $A \in \mathbb{R}^{d\times d}$ is symmetric, then it has an eigendecomposition of the form*

$$A = \begin{bmatrix} u_1 & u_2 & \cdots & u_d \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & \lambda_d \end{bmatrix} \begin{bmatrix} u_1 & u_2 & \cdots & u_d \end{bmatrix}^T$$

where the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ are real and the eigenvectors $u_1, u_2, \ldots, u_n$ are real and orthogonal. In addition,

$$\begin{aligned} \lambda_1 &= \max_{\| \||_2=1} x^T Ax \\ u_1 &= \arg \max_{\| x|_2=1} x^T Ax \\ \lambda_k &= \max_{\|x\|_2=1, x\perp u_1,\ldots,u_{k-1}} x^T Ax, \quad 2 \leq k \leq d-1 \\ u_k &= \arg_{\|x\|_2=1, x\perp u_1,\ldots,u_{k-1}} x^T Ax, \quad 2 \leq k \leq d-1 \end{aligned}$$

Proof. The proof is based on induction on the dimension d. For the case $d = 1$ we have $A = a \in \mathbb{R}$. Therefore, we can set $u_1 := 1$ and $\lambda_1 := a$.

As needed for induction, the assumption is that the theorem holds for $d - 1$, and let $A \in \mathbb{R}^{d \times d}$ be a symmetric matrix. Lemmas 0.1 and 0.2 gives the eigenvector u_1 that achieves the maximum, equal to λ_1. Consider the matrix

$$A - \lambda_1 u_1 u_1^T$$

Its column space is orthogonal to u_1,

$$\left(A - \lambda_1 u_1 u_1^T\right) u_1 = A u_1 - \lambda_1 u_1 = 0$$

where 0 denotes the n-dimensional zero vector. Its row space is also orthogonal by the same argument. Both subspaces are therefore contained in span $(u_1)^{\perp}$, the orthogonal complement of the span of u_1, which is a subspace of dimension $d - 1$. Let $V_{\perp}$ be a $d \times d - 1$ orthogonal matrix whose columns are an orthonormal basis of span $(u_1)^{\perp}$.$V_{\perp} V_{\perp}^T$ is a projection matrix that projects onto span $(u_1)^{\perp}$, so that

$$A - \lambda_1 u_1 u_1^T = V_{\perp} V_{\perp}^T \left(A - \lambda_1 u_1 u_1^T\right) V_{\perp} V_{\perp}^T$$

We define $B := V_{\perp}^T \left(A - \lambda_1 u_1 u_1^T\right) V_{\perp}$, which is a $d - 1 \times d - 1$ symmetric matrix. By the induction hypothesis there exist $\gamma_1, \ldots, \gamma_{d-1}$ and $w_1, \ldots, w_{d-1}$ such that

$$\begin{aligned}
\gamma_1 &= \max_{\|y\|_2 = 1} y^T B y \\
w_1 &= \arg\max_{\|y\|_2 = 1} y^T B y \\
\gamma_k &= \max_{\|y\|_2 = 1, y \perp w_1, \ldots, w_{k-1}} y^T B y, \quad 2 \leq k \leq d - 2 \\
w_k &= \arg\max_{\|y\|_2 = 1, y \perp w_1, \ldots, w_{k-1}} y^T B y, \quad 2 \leq k \leq d - 2
\end{aligned}$$

For any $x \in$ span $(u_1)^{\perp}$, we have

$$x^T \left(A - \lambda_1 u_1 u_1^T\right) x = x^T A x$$

and $x = V_{\perp} y$ for a vector $y \in \mathbb{R}^{d-1}$ such that $\|x\|_2^2 = y^T V_{\perp}^T V_{\perp} y = \|y\|_2^2$. This implies

$$\begin{aligned}
\max_{\|x\|_2 = 1, x \perp u_1} x^T A x &= \max_{\|x\|_2 = 1, x \perp u_1} x^T V_{\perp} V_{\perp}^T \left(A - \lambda_1 u_1 u_1^T\right) V_{\perp} V_{\perp}^T x \\
&= \max_{|yy\|_2 = 1} y^T B y \\
&= \gamma_1
\end{aligned}$$

Inspired by this, we set $u_k := V_{\perp} w_{k-1}$ for $k = 2, \ldots, d$. Each u_k is an eigenvector of A with eigenvalue $\lambda_k := \gamma_{k-1}$:

$$\begin{aligned}
A u_k &= V_{\perp} V_{\perp}^T \left(A - \lambda_1 u_1 u_1^T\right) V_{\perp} V_{\perp}^T V_{\perp} w_{k-1} \\
&= V_{\perp} B w_{k-1} \\
&= \gamma_{k-1} V_{\perp} w_{k-1} \\
&= \lambda_k u_k
\end{aligned}$$

Note that these vectors are all orthogonal and unit norm. Also, we can express any $x \in \operatorname{span}(u_1)^\perp$ orthogonal to $u_{k'}$, where $2 \leq k' \leq d$, as $x = V_\perp y, y \in \mathbb{R}^{d-1}$, where $y \perp w_{k'-1}$, because $u_{k'}^T x = w_{k'-1}^T V_\perp^T V_\perp y = w_{k'-1}^T y$. This implies

$$\begin{aligned}
\max_{\|x\|_2=1, x\perp u_1,\ldots,u_{k-1}} x^T A x &= \max_{\|x\|_2=1, x\perp u_1,\ldots,u_{k-1}} x^T V_\perp V_\perp^T \left(A - \lambda_1 u_1 u_1^T\right) V_\perp V_\perp^T x \\
&= 1|\text{in}_{\|||_2=1, x\perp u_1,\ldots,u_{k-1}} x^T V_\perp V_\perp^T \left(A - \lambda_1 u_1 u_1^T\right) V_\perp V_\perp^T x \\
&= \max_{|y\|_2=1, y\perp w_1,\ldots,w_{k-2}} y^T B y \\
&= \gamma_{k-1} \\
&= \lambda_k
\end{aligned}$$

□

From this theorem, it follows,

Theorem 0.4. *Let $\mathcal{X}$ contain n vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ with sample covariance matrix $\Sigma_\mathcal{X}$ and let $u_1, \ldots, u_d$, and $\lambda_1 > \ldots > \lambda_d$ denote the eigenvectors and corresponding eigenvalues of $\Sigma_\mathcal{X}$. We have*

$$\begin{aligned}
\lambda_1 &= \max_{\|v\|_2=1} \operatorname{var}\left(\mathcal{P}_v \mathcal{X}\right) \\
u_1 &= \arg\max_{\|v\|_2=1} \operatorname{var}\left(\mathcal{P}_v \mathcal{X}\right) \\
\lambda_k &= \max_{\|v\|_2=1, v\perp u_1,\ldots,u_{k-1}} \operatorname{var}\left(\mathcal{P}_v \mathcal{X}\right), \quad 2 \leq k \leq d \\
u_k &= \arg_{\|v\|_2=1, v\perp u_1,\ldots,u_{k-1}} \operatorname{mar}\left(\mathcal{P}_v \mathcal{X}\right), \quad 2 \leq k \leq d
\end{aligned}$$

For more detailed results, consult [3]. We can apply the results of theorem 0.4 to the variance of the projection $X = WY$, which is $C_{xx} = W^T C_{yy} W$. Taking into account that $C_{yy} = V \Lambda V^T$ we get the same result as minimizing the reconstruction error, that is, $W = V I_{k\times d}$. It has been shown that the best linear projection such that C_{xx} is maximized is given by the eigenvectors of C_{yy} that correspond to the greatest eigenvalues.

Question 1.2.5: Explain in English what is the intuitive reason that the "double centering trick" is necessary in order to be able to solve for S given D.

The double centering trick tells us that in order to solve S for given D we need to have S centered, and thanks to that, when multiplying D on both sides by the centering matrix, we get the matrix S.

The reason why this happens is that we are centering a matrix that is already centered, therefore, the centering can be removed. Again, this happends because Y is column centered.

As shown in [4], the centering matrix is $J = I - n^{-1}11'$, and c is the vector representing the elements in the diagonal of YY',

$$\begin{aligned}
-\frac{1}{2} JDJ &= -\frac{1}{2} J\left(c1' + 1c' - 2YY'\right) J \\
&= -\frac{1}{2} Jc1'J - \frac{1}{2} J1c'J + \frac{1}{2} J(2S)J \\
&= -\frac{1}{2} Jc0' - \frac{1}{2} 0c'J + JSJ = S.
\end{aligned} \tag{9}$$

Question 1.2.6: Use the same reasoning as in the previous question (1.2.5) to argue that s_{ij} can be computed as $s_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{1i}^2 - d_{1j}^2)$, where d_{1i} and d_{1j} are the distances from the first point in the dataset to points i and j, respectively. In particular, argue that although the solution obtained by the "first point trick" will be different than the solution obtained by the "double centering trick", both solutions are correct.

As said before, Y is column centered. However, instead of removing the expectation y from each point y_i, we can choose a different origin. By choosing the first point as the origin, this is called the "first point trick", which allows to calculate S for given D. This approach is correct because not neccesarily the centroid of Y has to be the same as its origen of configuration.

We had $S_{ij} = y_i^T y_j$. Since the new center is the first point, the new y_i' will be $y_i' = y_i - y_1$, and therefore,

$$s_{ii} = y_i^{T'} y_i' = (y_i - y_1)^T (y_i - y_1) = ||y_i - y_1||_2^2 = d_{1i} \tag{10}$$

Same applies to s_{jj}, leading to the expression $s_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{1i}^2 - d_{1j}^2)$. Consequently, the solution by the "first point trick" will be different than the solution obtained by the "double centering trick", but both are correct [4].

Question 1.2.7: Show that the two methods, i.e., classical MDS when Y is known and PCA on Y, are equivalent. Which of the two methods is more efficient? (Hint: Your answer may involve a case analysis).

As expressed in literature [2], when Y is known in MDS, it is possible to calculate $S = Y^T Y$. By doing EVD on S, we obtain $S = U\Lambda U^T = (\Lambda^{\frac{1}{2}} U^T)^T (\Lambda^{\frac{1}{2}} U^T)$, where $U, \Lambda \in \mathbb{R}^{nxn}$. The non-negative eigenvalues of S are sorted in descending order in the diagonal of Λ.

Then, the $k-$dimensional latent variables represented in the matrix X can be expressed as

$$X = I_{kxn} \Lambda^{\frac{1}{2}} U^T \tag{11}$$

To show the equivalence between PCA and MDS, we first decompose the covariance matrix, which is proportional to YY^T. We take into account that

$$C_{yy} \propto YY^T = V\Sigma^T U^T U \Sigma V^T = V\Sigma\Sigma^T V^T = V\Lambda_{\text{PCA}} V^T \tag{12}$$

where $\Lambda_{PCA} = \Sigma\Sigma^T$. The result is $X_{\text{PCA}} = I_{k\times d} V^T Y$.

For the case of MDS,

$$S = Y^T Y = U\Sigma V^T V \Sigma U^T = U\Sigma^T \Sigma U^T = U\Lambda_{\text{MDS}} U^T \tag{13}$$

To show the equivalence,

$$\begin{gathered} X_{PCA} = X_{MDS} \\ I_{kxd} V^T Y = I_{kxd} \Lambda_{MDS}^{\frac{1}{2}} U^T \\ I_{kxd} V^T V \Sigma U^T = I_{kxd} \Sigma^T \Sigma_{MDS}^{\frac{1}{2}} U^T \\ I_{kxd} \Sigma U^T = I_{kxd} \Sigma U^T \end{gathered}$$

We can decide which one is more efficient based on the number of points and dimensions the dataset has. For a large number of points but few dimensions, it's preferable to use PCA, given that the matrix multiplication YY^T results in a smaller matrix than Y^TY. In the other case, where the number of points is small, but the dimension is high, it is more efficient to use MDS. Another possibility is to perform SVD on Y for both PCA and MDS, this would be an intermediate solution [2].

As written in question 1.3.10, for a data matrix with n data points and d dimensions, time complexity for PCA is $\mathcal{O}(d^2n) + \mathcal{O}(d^3)$. On the other hand, for MDS, to calculate the distance matrix $\mathcal{O}(n^2d)$ operations are needed and $\mathcal{O}(n^2)$ memory entries. The eigenvalue decomposition time complexity is $\mathcal{O}(n^3)$ [2].

Question 1.2.8: Argue that the process to obtain the neighborhood graph G in the isomap method may yield a disconnected graph. Provide an example. Explain why this is problematic.

Disconnected graphs might happen because of several reasons, such as missing data or multiple clusters in the data. By using in isomap, an example is the rule of k nearest neighbours, k-NN. We can have a disconnected graph when using a low number of neighbours, i.e., when k is low. The easiest way to see it is with k=1 neighbour. By choosing k to be one, it is very likely that in a sequence of n points, there will be two points i, j such that i will be the closest point to j and all the way round, j will be the closest point to i. These points won't be connected to any other point.

This is problematic because isomap works by computing local distances, and we would not able to compute distances between the points that are disconnected. The neighborhood relationships that should be preserved won't be captured. This means that the isomap won't reflect properly the neighbourhood relationships. When computing the affinity matrix A, the value a_{ij}, which represents the relationship between the points i and j, won't be defined, because there is no relationship defined between the point i and j. Later it's not possible to obtain a low-dimensional representation if in the matrix A there are elements not defined.

Question 1.2.9: Propose a heuristic to patch the problem arising in the case of a disconnected neighborhood graph. Explain the intuition of your heuristic and why it is expected to work well. How does your heuristic behave in the example you provided in the previous question?

There are several approaches to try to solve this issue. In the article [5], the authors find a heuristic to patch the problem of disconnected graphs. The method is called enhanced neighborhood graph (ENG), and it consists of adding iteratively and adaptively edges to the neighbour graph, until it's not disconnected anymore. In a neighbour graph, each data point is connected to the k nearest neighbours.

What ENG does is connecting each component of the graph to its nearest neighbour, forming pair-wise components. The pair-wise components are combined into a new connected component. After this step, there will be less connected components. This process is repeated until there's only one component and the neighborhood graph is connected. A more detailed description can be found in [5]. It is relevant the choice of how many pair-wise components should be connected, and which ones.

Nevertheless, there are other methods also used. One heuristic would be to connect each component with the rest of components . The other heuristic consist of connecting small components to larger components.

For the example of the last question, we could just take the points i,j as a pair and connect them to the closest neighbour from the other bigger graph.

Question 1.3.10: Provide a qualitative comparison (short discussion) between the two methods, PCA vs. Johnson-Lindenstrauss random projections, in terms of (i) projection error; (ii) computational efficiency; and (iii) intended use cases.

i) Projection error: Minimizing the reconstruction error (mean squared error) means minimizing the distorsion introduced by doing the mapping of Y to the lower dimensional space and then mapping it back. It is very important to recall that what is minimized is the expectation, that is, the average. Consequently, reconstruction error guarantees distance preservation on expectation. This doesn't prevent some samples from having a high error, for outliers the error can be very large. Here is when random projections (RP) come into play, because in RP the worst case error is minimized, instead of the average error [6].

ii) Computational efficiency: RP are more efficient. For a data matrix with n data points and d dimensions, time complexity for PCA is $\mathcal{O}(d^2n) + \mathcal{O}(d^3)$, whereas RP is $\mathcal{O}(dkn)$, where k is the size of the projection. The larger d is, the greater the difference becomes. With a sparse matrix, the difference can be even more noticeable, the time complexity becomes $\mathcal{O}(ckn)$, where c is the non-zero values per column [7].

iii) Intended use cases: RP might be less accurate than PCA, but in certain cases the difference in accuracy might be neglectable. RP should be use when the data is very high dimensional, whereas PCA is better in the cases that time complexity is not an issue. For example, in the cases o high resolution images, the matrices are large, therefore RP might be a better option. Besides, data streaming is suited for RP.

Question 1.4.11: (Data collection)

Instead of an API, a dataset was taken from Kaggle [8], with more than 200 cities from all around the world, its longitude, latitude, continent and country they belong to. After that, the dataset was cleaned and processed (for example, remove na values). The geodesic distance of all the points could be calculated thanks to the library geopy.distance. It was done for all the pair of points, creating a symmetric matrix D, whose elements are the square value of the distance. There are cities from the 5 continents, mainly representing Europe, Africa, Asia, Australia, North America, Central America and South America. More information can be found in the appendix, where the jupyter notebook is appended as a pdf. The heatmap of the distances is shown in figure 1.

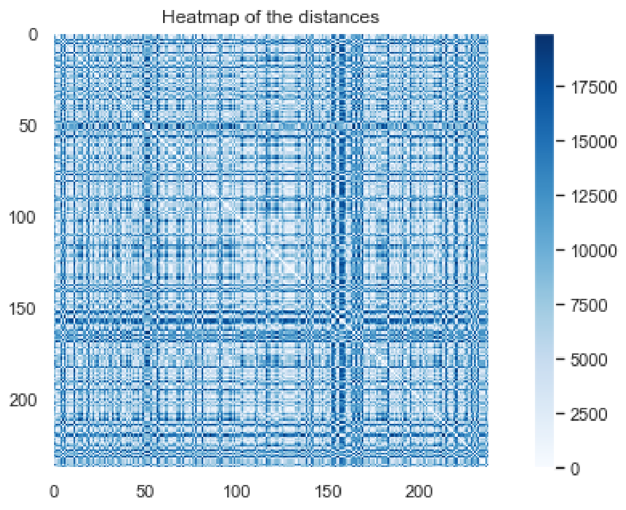

Figure 1: Heatmap of the distances.

Question 1.4.12: (Classical MDS).

The similarity matrix S was computed by doing the double centering trick, and then its EVD was performed with the library numpy. Thanks to this, the two dimensional projection could be obtained. The plot of the cities is shown in figure (3). A simpler example with some European cities is shown in figure (4).

The results seem very good for Classical MDS, even thought not all the relative positions of the cities are as expected. In the plot of some European cities, the triangle Madrid, Rome, Paris seem to be a good enough representation. In the north, there are Helsinki and Stockholm, and also distances there seem to be respected. However, there are also some errors, such as the relative position of Rome and Athens, given than Athens is further south than Rome. Similar is the case of Berlin and London.

Looking at the representation of 238 cities from different continents, we can take a look to the relation of Europe and Africa. Europe is above Africa, which is correct, but at some point some cities of Europe and Africa almost overlap, which should not be happening. The most visible example is in America, where North America, Central America and South America are a bit overlapped. At least we can see in a general view that there's Europe, below Africa, on the left America, on the Right Asia and below Oceania, which makes sense. In general, the map is a good reconstruction, despite having some errors.

Question 1.4.12: (Metric MDS).

Parameters tuning. I have used the sklearn library sklearn.manifold. There are several parameters that can be tuned:

-n_init. Number of times the SMACOF (Scaling by MAjorizing a COmplicated Function) algorithm [9] will be run with different initializations. The final results will be the best output of the runs, determined by the run with the smallest final stress

-max_iter. Maximum number of iterations of the SMACOF algorithm for a single run.

-eps. Relative tolerance with respect to stress at which to declare convergence.

-n_components. The number of components to select. In this case we are interested in two components.

There are some parameters that couldn't be tuned, such as the weights in the stress function.

By tuning max_iter we can see in figure 2 that the algorithm converges in few iterations. By changing the variable epsion or n_init the results don't seem to be different. Once the maximum number of iterations is large enough, the other two variables don't have a great influence. The final value of the set of parameters is: max_iter=50, eps=1e-3, n_init=4.

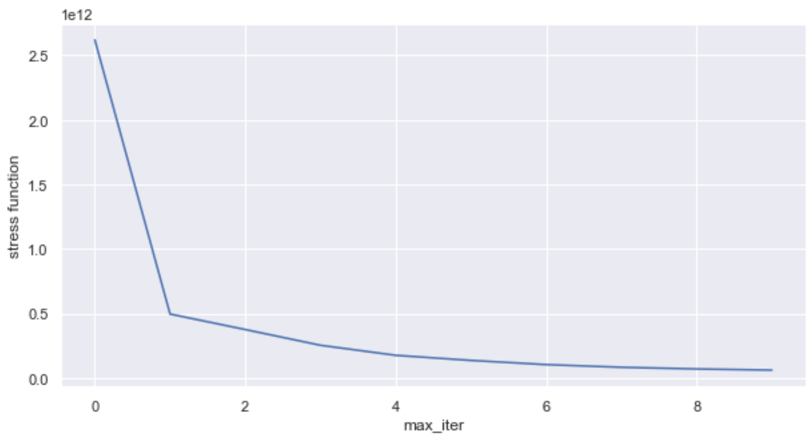

Figure 2: Variation of the stress function depending on the number of iterations.

Plots. By comparing the example of the European cities, in both classic and metric MDS the results are really similar, except for Athens. In metric MDS, Athens is further south than Rome, which makes sense and is a more truthful representation of relative positions. Therefore, in this case the results might be slightly better for metric MDS, even though both of them are almost identical.

For the plot of all the cities, there are more differences. This time the difference between Europe, Africa and Asia is more visible, and there's no overlap among the three of them. Despite that, there is still overlap in America (South, Central and North). I would conclude that metric MDS in this case is slightly better.

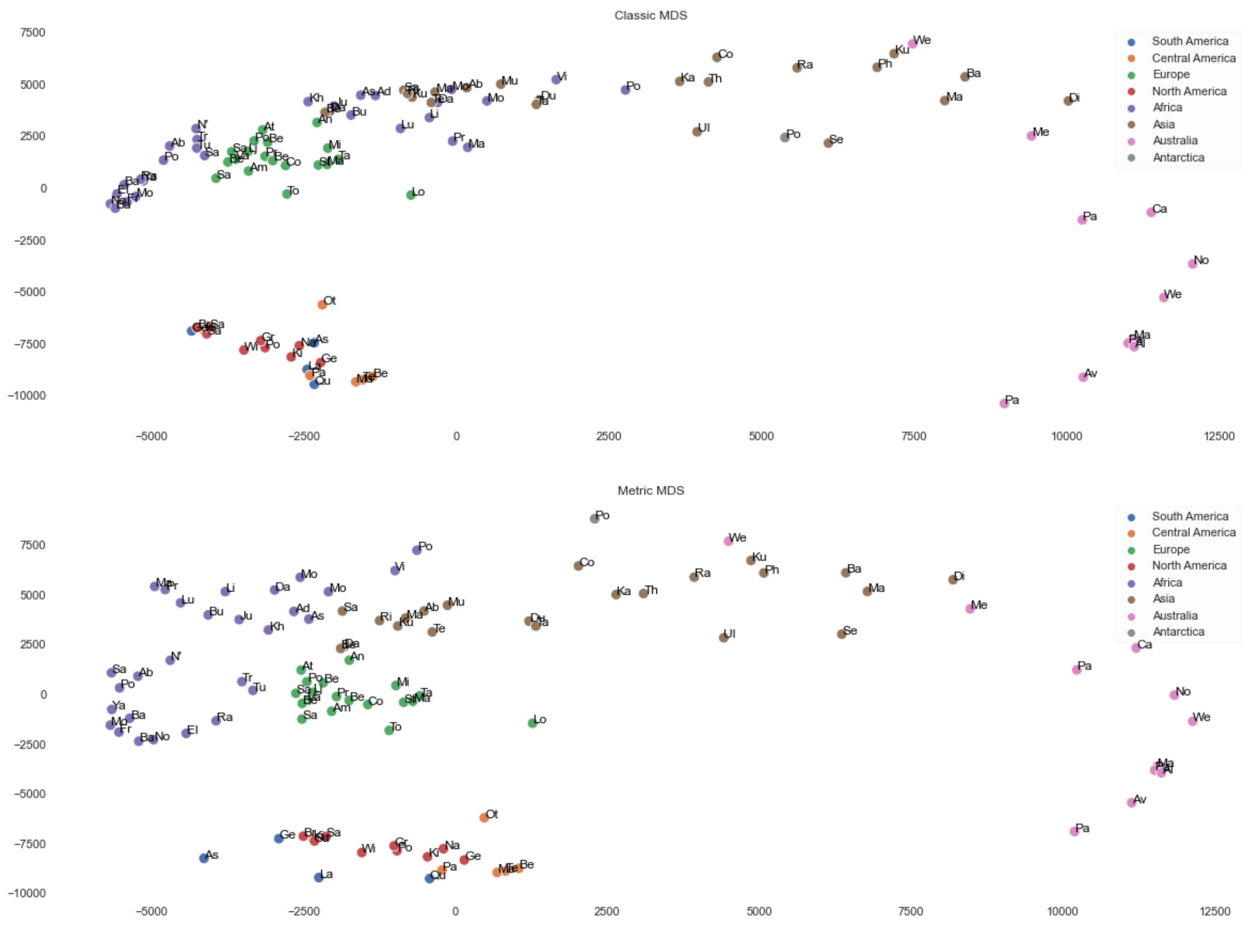

Figure 3: Plot of cities all around the world.

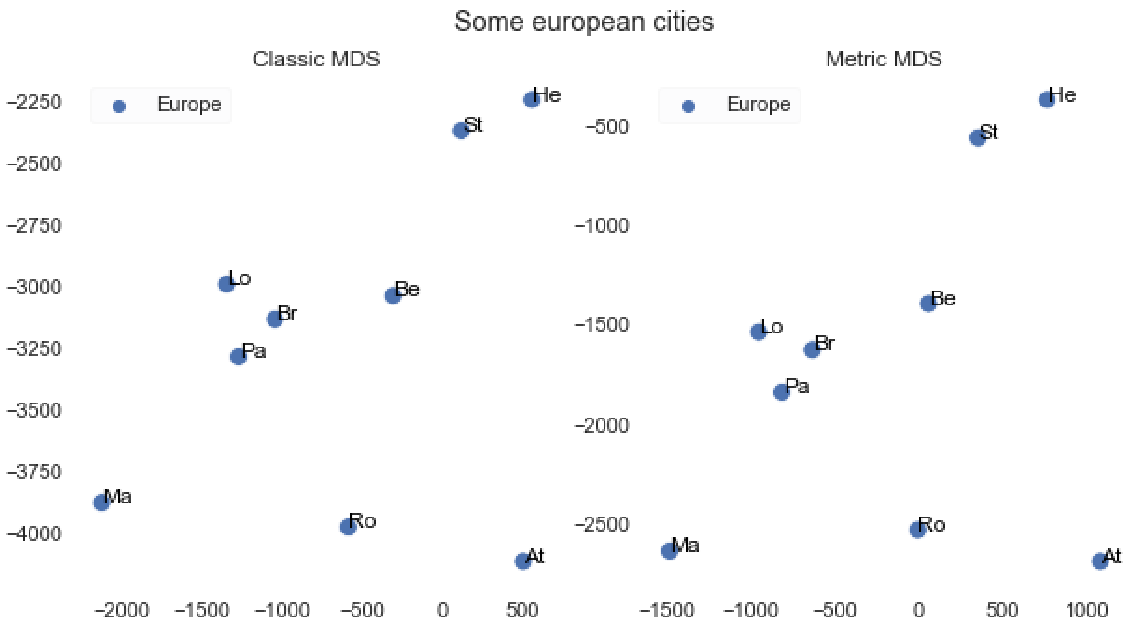

Figure 4: Plot of few european cities.

References

[1] Jorge Cadima and Ian Jolliffe. On relationships between uncentred and column-centred principal component analysis. *Pakistan Journal of Statistics*, Vol. 25(4):473–503, 2009.

[2] Michel Verleysen John A. Lee. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[3] Mathematical tools for data science,spring 2019. nyu. `https://cims.nyu.edu/~cfgranda/pages/MTDS_spring20/notes/pca.pdf`.

[4] Patrick J.F. Groenen Ingwer Borg. *Modern multidimensional scaling*. Springer series in statistics, 2005.

[5] Mingbo Zhao John K. L. Ho Jicong Fan, Tommy W. S. Chow. Nonlinear dimensionality reduction for data with disconnected neighborhood graph. (47:697–716), 3 August 2017.

[6] Advanced machine learning. lecture notes of modules 3 and 4.

[7] Shams S. Research Assistant at Purdue University. Random projection in dimensionality reduction. `https://machinelearningmedium.com/2017/07/28/random-projection-in-dimensionality-reduction/`.

[8] Dataset source. `https://machinelearningmedium.com/2017/07/28/random-projection-in-dimensionality-reduction/`.

[9] sklearn.manifold.smacof. `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.smacof.html`.