# Assignment 1

- *Auteur* -
Hamza Benjelloun

1er décembre 2020

# 0.1 Problem 1

## 0.1.1 Question i

Let A be a real symetric matrix and x, $\lambda$ an eigen pair of A. We note $x^H = \overline{x^T}$ e.g the conjugate trasnpose of x.

We have $Ax = \lambda x$ Then $x^H Ax = x^H x = \lambda ||x||_2^2$ $||.||_2$ is the canonical norm in $\mathbf{C}^n$ .
And $x^H Ax = \frac{1}{\lambda} x^H A(\lambda x) = \frac{1}{\lambda} x^H AAx = \frac{1}{\lambda}(Ax)^H(Ax)$ because A is a real symmetric matrix. Then $x^H Ax = \frac{1}{\lambda}||Ax||_2^2$
Therefore $\lambda^2 = \frac{||Ax||_2^2}{||x||_2^2}$ (we know that $x \neq 0$ Then $||x||_2 \neq 0$ )
Conclusion $\lambda \in \mathbf{R}$

## 0.1.2 Question ii

Let A be a real symmetric matrix.

We suppose that A has at least two different eigenvalues $\lambda$ and $\mu$ because if it is not the case we just take an orthogonal basis from the only eigenspace.

Let x and y be the corresponding eigen vectors of $\lambda$ and $\mu$. We suppose that for example $\lambda \neq 0$ We have $x^T y = \frac{1}{\lambda}(\lambda x)^T y = \frac{1}{\lambda} x^T A^T y = \frac{\mu}{\lambda} x^T y$ because A is symmetric.

Then $(1 - \frac{\mu}{\lambda})x^T y = 0$ Therefore $x^T y = 0$ because $\mu \neq \lambda$

So, the eigenspaces are mutually orthogonal.

We get an orthogonal eigenvectors basis by concatenating orthogonal basis from each eigenspace.

First we will show that A is diagonalizable. We have from the fundamental theorem of algebra the characteristic polynom of A has exactly n complex roots and we know that those roots are the eigenvalues of A. Since A is symmetric we have all eigenvalues in $\mathbf{R}$ Then the characteristic polynom has exactly n real roots (counted with multiplicity). Therefore A is diaonalzable.

Let $x_1, x_2, ...x_n$ be n orthonormal eigenvectors of A, we put $Q = (x_1, x_2, ...x_n)$ and $\Lambda = diag(\lambda_1, \lambda_2, ...., \lambda_n)$

We have $AQ = (Ax_1, Ax_2, ...., Ax_) = (\lambda_1 x_1, \lambda_2 x_2, ...., \lambda_n x_n) = Q\Lambda$

And $(Q^T Q)_{i,j} = x_i^T x_j = \begin{cases} 1, & \text{if i = j} \\ 0, & \text{else} \end{cases}$

Then $Q^T Q = I_n$ Therefore $Q^{-1} = Q^T$

Then $Q^T Q = I_n$ Therefore $Q^{-1} = Q^T$

Therefore $A = Q\Lambda Q^T$

Finally $\exists Q \in \mathbf{GL_n}(\mathbf{R}) \exists \Lambda \in \mathbf{D_n}(\mathbf{R}) A = Q\Lambda Q^T$

### 0.1.3 Question iii

Let A a positive semidefinite matrix and x, $\lambda$ an eigenpair of A. Then $Ax = \lambda x$ Therefore $x^T Ax = \lambda x^T x = \lambda ||x||_2^2 \geq 0$ Then $\lambda \in \mathbf{R}^+$.

### 0.1.4 Question iv

Let A be a positive semidefinite matrix and D a matrix so that $D_{i,j} = A_{i,i} + Aj, j - 2A_{i,j}$

We have A is positive semidefinite matrix then it can be factorized with Cholesky method. Then $\exists L \in \mathbf{GL}(\mathbf{R})_n A = L^T L$ Let $(e_1, e_2, ...e_n)$ be an Orthonormal basis.

We take $V_i = \sum_{J=1}^n L_{i,j} e_j$.

We have in this case $< V_i, V_j >= \sum_{k=1}^n L_{i,k} L_{k,j} = A_{i,j}$

Then $||V_i - V_j||_2^2 =< V_i, V_i > + < V_j, V_j > -2 < V_i, V_j >= A_{i,i} + A_{j,j} - 2A_{i,j}$

## 0.2 Problem 2

Yes, asingle SVD operation is sufficient to perform PCA both on the rows and the columns of a data matrix. If we have $Y = U\Sigma V^T$, to perform pca on columns it is enough to take $W = U_k$ to maximize the trace. However to perform pca on rows it is enough to take $W = V_k$ to maximize the trace by swishing the role of U and V.

## 0.3 Problem 3

We have :

$$\mathbf{Tr}(y^T W w^T y) = \mathbf{Tr}((U\Sigma V^T)WW^T U\Sigma V^T) = \mathbf{Tr}(V\Sigma U^T WW^T U\Sigma V^T)$$

$$= \mathbf{Tr}(WW^T U\Sigma V^T V\Sigma U^T) = \mathbf{Tr}(WW^T U\Sigma^2 U^T) = \mathbf{Tr}(W^T U\Sigma^2 U^T W)$$

$$= \mathbf{Tr}((\Sigma U^T W)^T (\Sigma U^T W)) = ||\Sigma U^T W||_F^2$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} \sigma_i^2 (\sum_{l=1}^{m} U_{l,i} W_{l,j})^2 = \sum_{i=1}^{n} \sigma_i^2 \sum_{j=1}^{k} < U_i | W_j >^2$$

Let note $c_i = \sum_{j=1}^{k} < U_i | W_j >^2$ we have

$$c_i = \sum_{j=1}^{k} < U_i | W_j >^2 \leq ||U_i||^2 = 1$$

because $W_j$ are orthonormal columns

And we have also :

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n}\sum_{j=1}^{k} < U_i|W_j >^2 = \sum_{j=1}^{k}\sum_{i=1}^{n} < U_i|W_j >^2 = \sum_{j=1}^{k} ||W_j||^2 = \sum_{j=1}^{k} 1 = k$$

We have also

$$\sigma_1 \geq \sigma_2 \geq .... \geq \sigma_n$$

Then the maximum of the value $\sum_{i=1}^{n} \sigma_i^2 c_i$ over all choices of $c_i \in [0,1]$ with $\sum_{i=1}^{n} c_i = k$ is $\sum_{i=1}^{k} \sigma_i$. This is achieved when $c_1 = c_2 = ... = c_k = 1$ and $c_{k+1} = ... = c_n = 0$ .That means $Span(W_1, W_2, ..., W_k) = Span(U_1, U_2, ..., U_k)$

Then :

$$\max_W(\mathbf{Tr}(y^T W w^T y)) = \sum_{i=1}^{k} \sigma_i^2$$

$U_i$ are orthonormal columns and if we take $(\forall i \in [|1,k|])W_i = U_i$ we find :

$$\mathbf{Tr}(y^T W w^T y) = \sum_{i=1}^{n} \sigma_i^2 \sum_{j=1}^{k} < U_i|W_j >^2 = \sum_{i=1}^{n} \sigma_i^2 \sum_{j=1}^{k} < U_i|U_j >^2 = \sum_{i=1}^{k} \sigma_i^2 < U_i|U_i >^2 = \sum_{i=1}^{k} \sigma_i^2$$

Therefore $\mathbf{Tr}(y^T W w^T y)$ is maximized by $W = U_k$

## 0.4 Problem 4

Let assume the existence of a certain matrix Y of points that gives the distance matrix D. That means $d_{i,j} = ||y_i - y_j||$

We have

$$-\frac{1}{2}(d_{i,j}^2 - d_{1,i}^2 - d_{1,j}^2) = -\frac{1}{2}(||y_i - y_j||^2 - ||y_1 - y_i||^2 - ||y_1 - y_j||^2)$$
$$=< y_1, y_1 > + < y_i, y_j > - < y_1, y_i > - < y_1, y_j >=< y_i - y_1, y_j - y_1 >$$

Let consider the translation matrix $T = \begin{pmatrix} y_1 & y_1 & \cdots & y_1 \end{pmatrix}$ and let note $Z = Y + T$, we have $S = Z^T Z$

and we have

$$||z_i - z_j|| = ||(y_i - y_1) - (y_j - y_1)|| = ||y_i - y_j|| = d_{i,j}$$

Then Z conserve the distance matrix. Which means that using $S = Y^T Y$ or $S = Z^T Z$ is equivalent. We conclud that our estimation of Gram matrix is correct.

## 0.5 Problem 5

To prove that MDS is equivalent to PCA we will show that the result of MDS maximize the trace.

We have S is symmetric and positive semi-definite, so its eigen-decomposition is : $S = U\Lambda U^T$

The solution of classical MDS is $X = I_{k,n}\Lambda^{1/2}U^T$

We have

$$\begin{aligned}\mathbf{Tr}(Y^TWW^TY) &= \mathbf{Tr}((W^TY)^T(W^TY)) = \mathbf{Tr}(X^TX) = \mathbf{Tr}((I_{k,n}\Lambda^{1/2}U^T)^T(I_{k,n}\Lambda^{1/2}U^T)) \\ &= \mathbf{Tr}(U\Lambda^{1/2}I_{n,k}I_{k,n}\Lambda^{1/2}U^T) = \mathbf{Tr}(I_{k,n}\Lambda^{1/2}U^TU\Lambda^{1/2}I_{n,k}) \\ &= \mathbf{Tr}(I_{k,n}\Lambda I_{n,k}) = \sum_{i=0}^{k}\lambda_i\end{aligned}$$

We know that $\Lambda$ contains ordered eigenvalue of S. And we have in other hand from PCA :

$$\exists U_1 \in R^{n,m} \exists V_1 \in R^{n,n} \Sigma \in D_n(R), Y = U_1\Sigma V_1^T$$

. Then $S = Y^TY = V_1^T\Sigma^2V_1$. Let $V_{1i}$ be the ith column of $V_1$.

We have

$$SV_{1i} = V_1^T\Sigma^2V_1V_{1i} = V_1^T\Sigma^2\begin{pmatrix}0\\\vdots\\1\\\vdots\\0\end{pmatrix} = V_1^T\begin{pmatrix}0\\\vdots\\\sigma_i^2\\\vdots\\0\end{pmatrix} = \sigma_i^2V_{1i}$$

Therefore $V_{1i}$ are the eigenvectors of S and $\Sigma^2$ contains also ordered eigenvalues of S.

Then

$$\sum_{i=0}^{k}\lambda_i = \sum_{i=0}^{k}\sigma_i^2$$

Then classical MDS algorithm is equivalent to PCA.

Let's compare MDS and PCA in terms of complexity.

The main steps in PCA is to compute SVD and to multiply the data (in our case a matrix of size $n \times m$) with the principals components. From literature (see the article of "A Hierarchical Singular Value Decomposition Algorithm for Low Rank Matrices") , Computing the SVD of an $n \times m$ matrix has complexity $O(nm\min(n,m))$ and the multiplication in our case has the complexity of $O(knm)$ ($k = 2$ we transform data to 2D space). Therefore the complexity of PCA is $O(nm(min(n,m)+k))$ in our case $k = 2$ then the complexity of PCA is $O(nm(\min(n,m))$ .

The main steps in MDS is the double centring and eigen decomposition. We have from literature most of the algorithms for eigen decomposition have

a complexity of $O(n^3)$. And we know that the complexity of multiplying two square matrix of size $n \times n$ is $O(n^3)$. And since the double centring is just a multiplication and summation then the complexity of MDS is $O(n^3)$

Most of time we have the number of rows is greater than the number of columns (that means $n \geq m$). Then the complexity of PCA becomes $O(nm^2)$ and $nm^2$ will be in our case less than $n^3$. Then PCA is computationally better than MDS

## 0.6 Problem 6

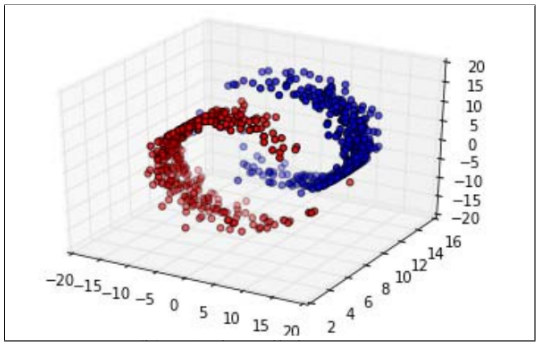


FIGURE 1 – **Two disconnected manifolds**

As we see in the picture above we have two disconnected non linear manifolds. Then by applying isomap algorithm we will have two disconnected graphs, because the distance between the two manifolds is big. The solution is to seek for the density peak point for each manifold.

One of the simplest solution is to increase the number of neighbors until the graph become connected.

An other solution is to try to connect the separated graphs

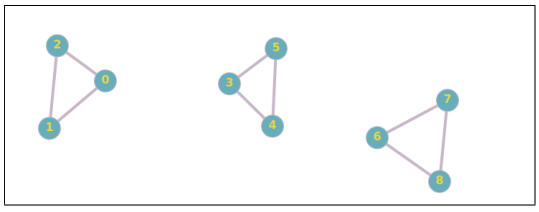


FIGURE 2 – **disconnected graph**

For example in this image we have tree separated graphs. We will connect each connected component to the nearest one. In our example we will connect the vertex 0 to the vertex 3 and the vertex 4 to the vertex 6. Therefore the graph becomes :

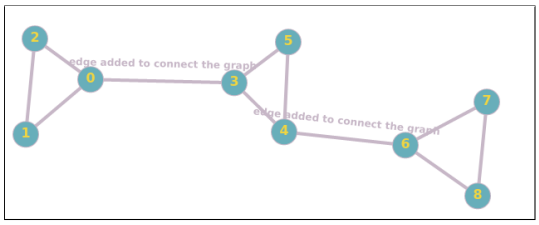


FIGURE 3 – **connected graph**

## 0.7 Problem 7

The code is in KTH git : https://gits-15.sys.kth.se/hamzabe/ML_advanced

Before applying methods of dimensionality reduction we should process a little bit our data. I choose to change the column 13 by using one-hot encoding to have only booleans. There was a small improvement in the 3 methods.

Let start by PCA :

Listing 1 – Insert code directly in your document

```
# Use implemented pca in python
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(data_centred)
# plot is a function implemented to draw points with the same type with
    the same color
plot(principalComponents, type)
```

We obtain :

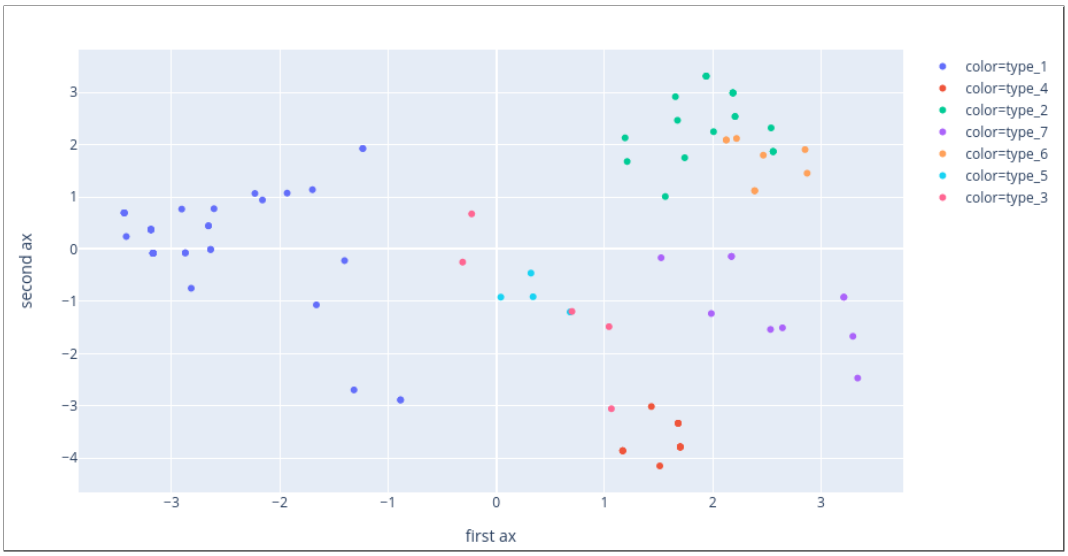


FIGURE 4 – **performing PCA on data**

We see that the points with the same type are grouped together but not perfectly separated. We can also perform PCA manually by importing SVD in python and the code will be :

Listing 2 – Insert code directly in your document

```
# apply SVD on centred data
U, S, Vt = svd(np.transpose(data_centred), full_matrices=True)
# extract the two principales componentes
U_2 = U[:, [0, 1]]
#transform data
X_pca = np.dot(np.transpose(U_2), np.transpose(data_centred))
plot(np.transpose(X_pca), type)
```

We obtain the same result as the PCA implemented in python.

Let's compute the error committed by PCA :

$$Error = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |||x_i - x_j||_2^2 - ||f(x_i) - f(x_j)||_2^2|$$

We obtain for pca an error of 1.7246.

Let's move to the classical MDS :

Listing 3 – Insert code directly in your document

```
#compute S by double centring
ones = np.ones((n, n))
S = - (1/2) * (D - (1/n) * np.dot(D, ones) - (1/n) * np.dot(ones, D) +
    (1/n**2) * np.dot(ones, np.dot(D, ones)))
#EigenDecompostion
eigenvalues, eigenvectors = lg.eig(S)
eigenvalues, eigenvectors = np.abs(np.real(eigenvalues)), np.real(
    eigenvectors)
lambdas = np.sqrt(np.diag(eigenvalues))
X_mds = np.dot(np.eye(k, n), np.dot(lambdas, np.transpose(
    eigenvectors)))
plot(np.transpose(X_mds), type)
```

We obtain :

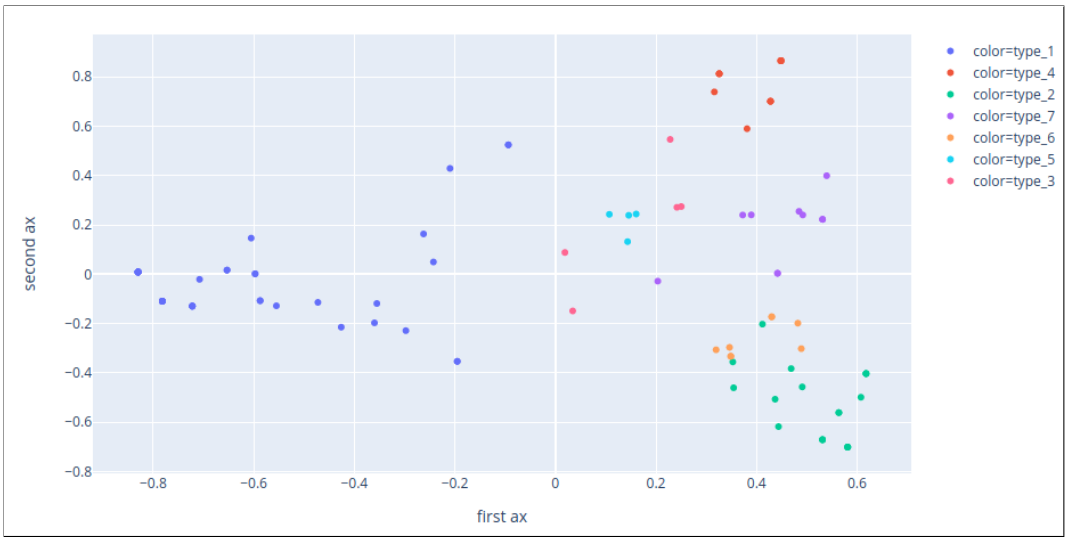


FIGURE 5 – **performing MDS on data**

We notice that the result obtained is similar to the one obtained by PCA. And that confirm what we demonstrate in the question 5.

Let's experiment the influence of feature importance on MDS method.

**Random forest method :**

We can define feature importance as the gain of information provided by random forest algorithm. It offers importance scores based on the reduction in the criterion used to select split points (the entropy).

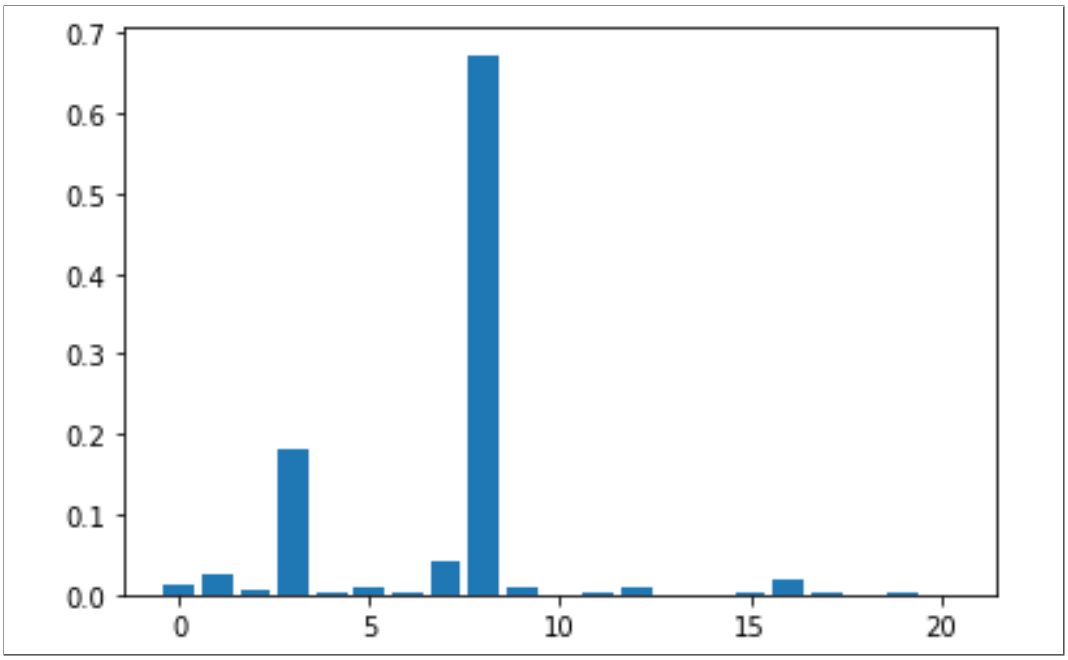


FIGURE 6 – **Feature importance with random forest**

Another method to compute feature importance is "Permutation Feature Importance" . The concept is really straightforward : We measure the importance of a feature by calculating the increase in the model's prediction error after permuting the feature. A feature is "important" if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction.We obtain :

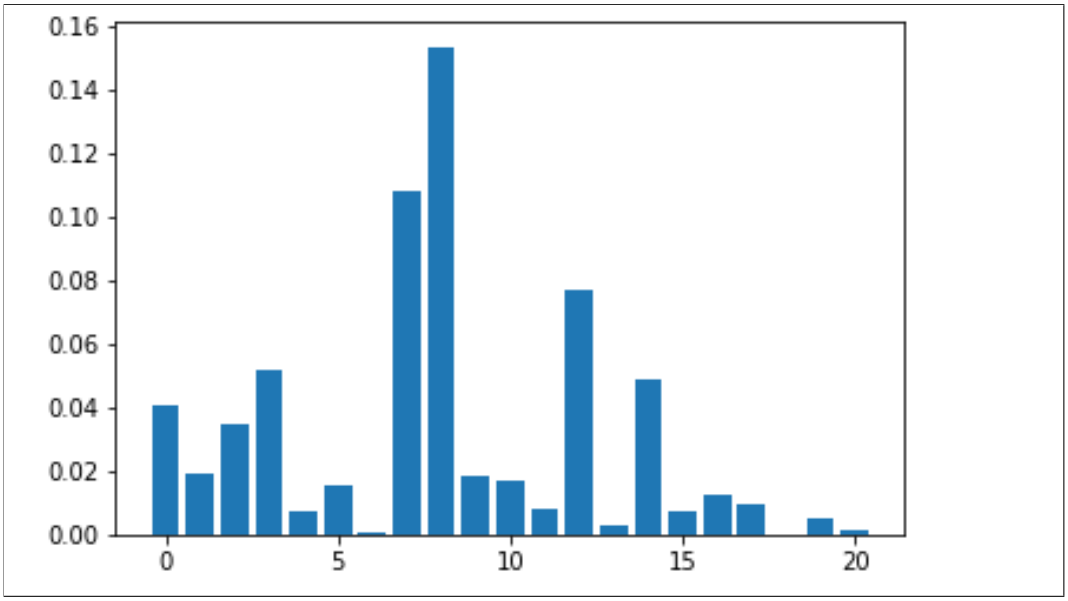


FIGURE 7 – **Feature importance with permutation**

When we combine the two methods and we compute the moyen of the two feature importances we get :

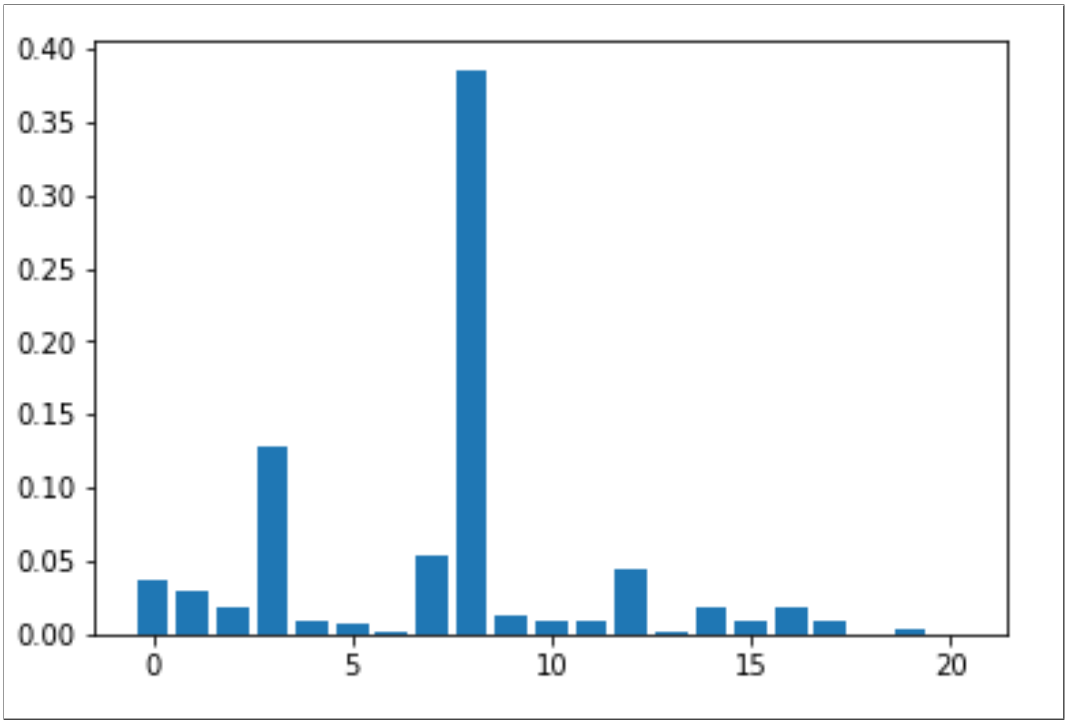


FIGURE 8 – **Feature importance with both permutation and random forest**

We use features importance to define a weighted distance between points :

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} W_k (x_{ik} - x_{jk})^2}$$

with $W_i$ is the weight of each feature. It is clear that $W_i$ should be an increasing function of feature importance, to favor column with high feature importance.

Here one of the best result we obtained

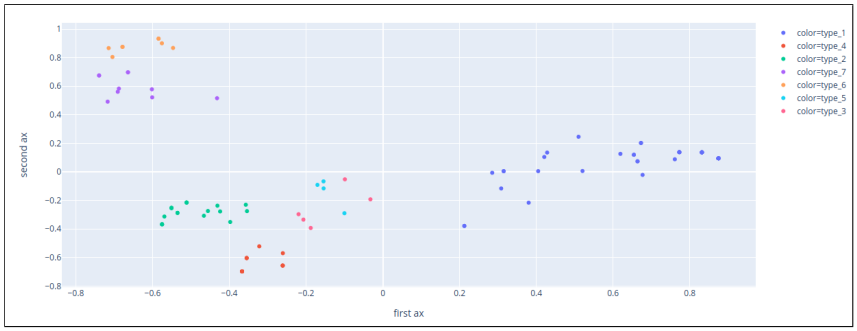


FIGURE 9 – **Mds with weighted data**

We notice that we get a good result and the points with the same type are grouped together.

Let's move to isomap method.

Listing 4 – Insert code directly in your document

```
# define the number of neighbors
nbr_neighbors = 16
# build the graph and compute distances
graph = kneighbors_graph(data, nbr_neighbors, mode='distance')
# perform MDS using the geodesic distance
X_siomap = MDS(distances, 2)
plot(X_isomap, type)
```

We obtain :

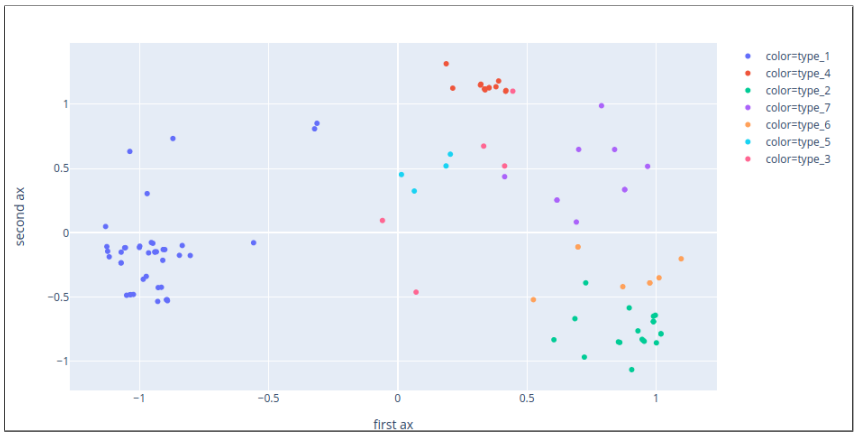


FIGURE 10 – **performing isomap on data**

To assert the connectivity of the graph before applying MDS, I implement a function based on DFS to compute the different connected components. We obtain a full connected graph with a number of nearest neighbors equal to 16.

As we see 16 is big compared with the total length of data (101 points). That is because we have a lot of duplicated points.

Listing 5 – Insert code directly in your document

```
# counting the duplicates
columns = [col for col in data.columns]
duplicated = data.pivot_table(index = columns, aggfunc ='size')
max(duplicated)
```

The result of this code is 10, therefore we have a point duplicated 10 times, then we should use at least 11 nearest neighbors to have a full connected graph. to minimize the minimum of nearest neighbors needed to have a full connected graph we can remove duplicated points. But the problem here is that some points have the same features but different types (it is logical because features represent the number of arms, leg ... , and a lot of animals have the same properties). Another problem concerning duplicated points, even though they did not introduce any new information in the sample but they confirm the

measures made on animals and give more weight to some points. Then it is complicated to remove duplicated points.

Let see the impact of the number of nearest neighbors :

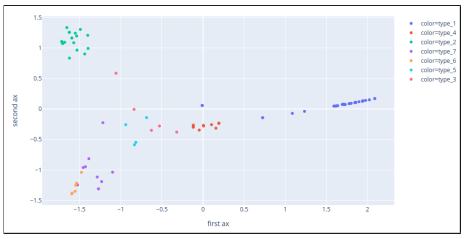

FIGURE 11 – **Isomap with 6 nearest neighbors**

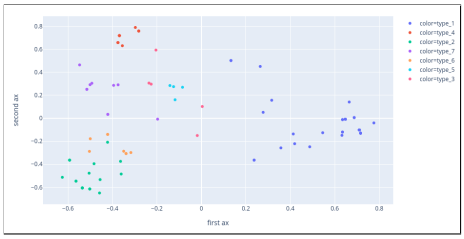

FIGURE 12 – **Isomap with 100 nearest neighbors**

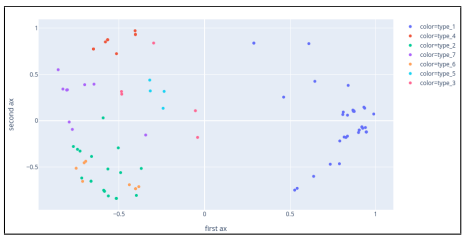

FIGURE 13 – **Isomap with 40 nearest neighbors**

We see that in the particular case : 100 nearest neighbors we find the result of MDS and PCA. Moreover with 6 nearest neighbors we see different disconnected graphs.

Let's compare Isomap with PCA, (classicla MDS is the same as PCA) :

**Reconstruction error :**

For PCA :

For the dimensionality reduction we do : $X_{pca} = U_k^T X$ and for the reconstruction we do : $X' = U_k X_{pca}$

where X is the original data, $X_{pca}$ is the reduced space, $U_k$ are the loadings (PCs) and $X'$ is the reconstructed space.

Finally, we commonly express the error as :

$$Reconstruction\_error = (X' - X)^2$$

an implementation of this approach gives an error of 0.5477.

For Isomap :

It is difficult to implement manually an algorithm that compute the error

of reconstruction in isomap. Therefore we will use the method implemented in python. It gives an error of 1.7219.

Comparaison :

Since the PCA objective places much more emphasis on preserving the large pairwise distances, PCA achieves lower reconstruction.

**Local structure error :**

One way to measure the preservation of local structure would be to compute the nearest neighbor of every point in the original dataset and count how many times the corresponding point's nearest neighbor in the embedding agrees. By applying this method using 16 nearest neighbors we find :

$$Error\_isomap = 0.3050$$

$$Error\_pca = 0.3242$$

**Conclusion :**

We notice that there is no big difference between PCA and isomap in this data. Maybe because our data have only linear relationships and we don't have to use isomap.