LOW DIMENSIONAL EMBEDDING OF ENVIRONMENTAL VARIABLES

EA MAP581

17 mars 2018

Flore Martin and Lorraine Roulier

ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Table des matières

1 Introduction

Dimension reduction refers to the process of reducing the number of random variables under consideration. It is an efficient way to analyze a large dataset of high dimension. For instance, dimensionality reduction can be used for **visualizing** or exploring structure in data. It provides a **simple geometric interpretation** by mapping in 2D or 3D an original dataset of dimension $d > 3$. A 2D or 3D mapping is thus much easy to interpret for us human. In terms of performance, having data of high dimensionality is problematic because it can mean **high computational cost** to compute. Finally, dimensional reduction is often used for classification. Data points are being clustered, it is a way to establish clusters of gene and of population for example.

In practice, dimension reduction is widely used for denoising or compressing data, and embedding image processing like facial recognition.

2 Dataset presentation

Climate data amounts very quickly to a lot of unused data. In a day, we can collect temperature, pressure, wind data all over the world with satellites, even hourly. We used a set of datasets we found on the NASA website, that gathered various means on climate variables over 22 years at every given latitude and longitude. Those datasets can de found here [6]. AS each variable (temperature, radiation, pressure, wind speed and humidity) was stored in a separate dataset, we first had to merge all of them in a final table presented below :

Latitude	Longitude	Temperature $°C$	Pressure kPa	Relative Humidity $\%$	Wind Speed m/s	Radiation $kWh/m^2/day$

Figure 1 – Variables of our dataset

The latitude parameter varies from -90 to 89 and the longitude parameter varies from -180 (south pole) to 179 (north pole). Depending on the running time of the method, we did not compute the dimension reduction with the 64800 lines, but with a subset. The subset is often a slice of longitudes containing all latitudes, as we assumed that the critical parameter to differentiate climate data was the latitude. We will detail the subset considered in every method.
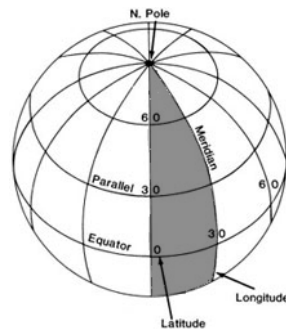
FIGURE 2 – An example of subset in grey

We classified the data according to the latitude, creating five classes listed in the table below. A more visual picture of our classification is given when we introduce the results of the different dimension reduction methods.

	North	Temperate north	Equator	Temperate South	South
Latitudes	90 to 66	66 to 23	23 to -23	-23 to -66	-66 to -90
	🔵	🔴	🟢	🟡	🔴

FIGURE 3 – Our classification

3 Methods

Dimension reduction techniques can be divided in two classes, linear dimension reduction and non linear dimension reduction. However, in all methods, the main goal is to figure out a similarity function between vectors. Such a function will then enable to sort the dataset into classes of vectors with similar features, which would have been more intricate with the initial dataset.

Our project was two sided. First, we familiarized with various dimension reduction techniques, then we attempted to show that the geographical position of a point on the planet - e.g. it's latitude and longitude - were embedded in the climate data one could gather on it.

3.1 Principal Component Analysis - PCA

Principal Component Analysis detects tendencies in the data by maximizing the variance of the dataset matrix. There is an easy approach to understand PCA with a 2D dataset. On Figure ??, we have plotted a 2D dataset. If we implement 2D-PCA, we will find two new axes that reflects at best the variances of datapoints. Those ares the green axes. PCA just 'rotates' the figure to find a more explicit 'point of view'. The yielded axes are a linear combination of the original axes.
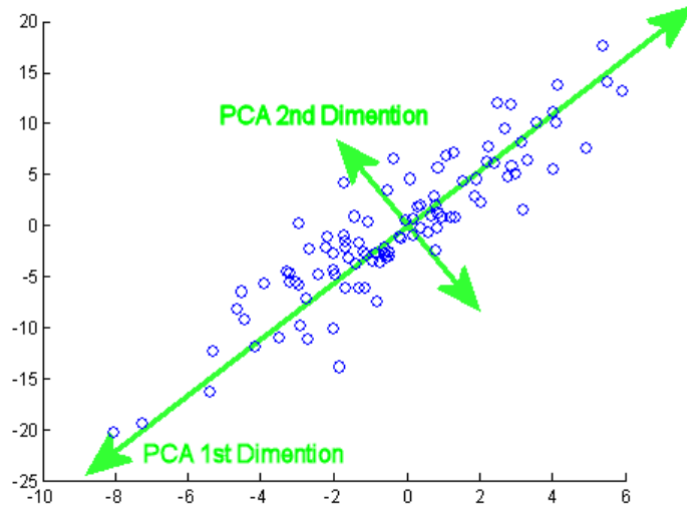
FIGURE 4 – role of PCA - interpretation

From a mathematical point of view, PCA yields an orthonormal matrix that can be diagonalized. The largest eigenvalues point to the eigenvectors that contain the most information about the dataset.

Let $X \in \mathbb{R}^{d \times n}$ be our dataset where d is the number of variables and n the sample size, PCA maximizes the following equation :

$$\|X - MM^T X\|_F^2$$

subject to $M \in \mathcal{O}^{d \times r}$ where $r < d$ and $\mathcal{O}^{d \times r}$ is the space of orthogonal matrix and we use the Froebenius norm.

The maximum of the previous function is reached by the covariance matrix of the dataset : the r greatest eigenvalues of $XX^T = PDP^T$ yield the maximum $M = P_r$ where the columns are the eigenvectors associated to the eigenvalues mentioned above. The largest eigenvalues point to the eigenvectors that contain the most information about the dataset.

The new dataset is thus $Y = P_r^T X$ of dimensions $r \times n$. In this example, there are r principal components for the PCA.

3.2 Kernel PCA

Kernel Principal Component analysis is a non linear method for dimension reduction. Instead of directly maximizing the variance of X, we map it into a larger dimension space called the feature space, using a non linear function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^N$ where $N > d$.

Instead of directly mapping X into the feature space, we use a kernel, that represents a similarity function between pairs. Let $K : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be this kernel, then

$$K(x_i, x_j) = \phi(x_{\hat{\imath}})^T \phi(x_j)$$

where x_i and x_j are vectors of the dataset X

This kernel will enable us to bypass the higher dimension calculation of the variance in the feature space. We only need to compute the pairwise values for ϕ, but there is no need to explicitly define ϕ.

The downside of this method is that since we don't compute directly the eigenvectors and eigenvalues of ϕ, the result is the projection of our dataset onto the eigenvectors. We thus don't have access to a simple interpretation of the principal components.

A classic kernel that is often used is the RBF function that is defined below :

$$K(x_i, x_j) = \mathrm{e}-\gamma\|x_i - x_j\|$$

3.3 Multidimensional scaling

Multidimensional scaling is a non-linear approach to reduce dimension of a dataset. The principle is quite different from PCA : given a matrix of distance or "'dissimilarity"', MDS aims at reconstructing a map preserving at best the distances between datas. In our example, the MDS algorithm aims to place each object in 2-dimensional space such that the between-object distances are preserved as well as possible. Given a distance matrix D of dimension $n \times p$ (p=5 for us) , MDS attempts to find the datapoints $y_1,... y_t$ in dimension d<p (d=2 for us) that minimizes :

$$\sum_{i=1}^{t}[\sum_{j=1}^{t} d_{ij}^{(X)} - d_{ij}^{(Y)}]$$

with $d_{ij}^{(X)}$ the euclidean distance between pairwise i and j in the original $n \times p$ matrix $D^{(X)}$ and $d_{ij}^{(Y)}$ the euclidean distance between pairwise i and j in the original $n \times p$ matrix $D^{(X)}$ and in the computed $n \times d$ matrix $D^{(Y)}$.

A basic approach to understand MDS is to use the example of cities. The input is the distance matrix between cities on the left on figure ? ?. MDS enabales us to find coordonates of each city based on this distance matrix.
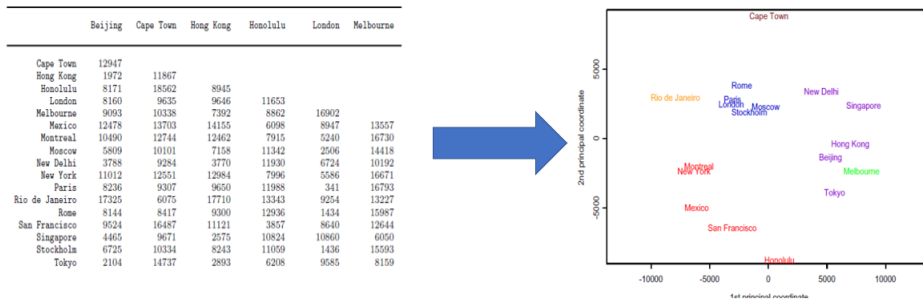

FIGURE 5 – role of MDS - interpretation

3.4   Isomap

Isomap is a low dimensional embedding method similar to MDS. The difference is, distance is not computed with euclidean norm but with geodesic distance. As the earth shape is round, isomap seems more relevant than MDS. ON figure ??, we can see the difference between the geodisic distance (in red) used in isomap, and the eucliean distance used in MDS (in blue).
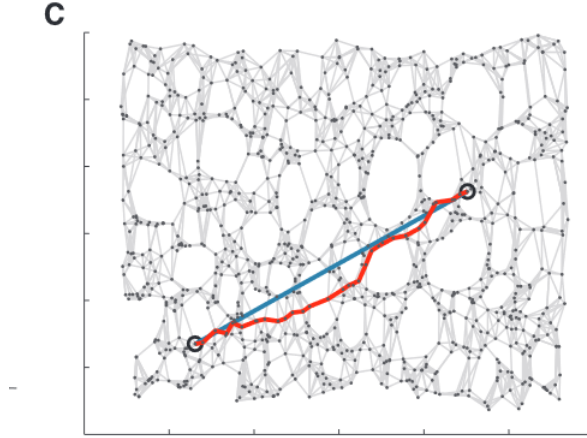

FIGURE 6 – geodisic and euclidean distance

Isomap algorithm aim at minimizing the recontruction error E. Its exact formula is :

$$E = \frac{\|K(D) - K(D_{fit})\|}{n_{samples}}$$

with D the original distance matrix computed with geodisic distance in input, D_{fit} the computed distance matrix in output, $n_{samples}$ the dataset size and K kernel matrix

4   Results

4.1   Graphic Interpretation

4.1.1   PCA

We first implemented PCA and ran it with only two principal components, which yielded the following graph for the whole dataset. The associated eigenvectors were

$$y_1 = \begin{bmatrix} -0.94 \\ -0.31 \\ 0.11 \\ -0.04 \\ 0.01 \end{bmatrix}$$

and

$$y_2 = \begin{bmatrix} 0.03 \\ -0.40 \\ -0.91 \\ 0.04 \\ -0.05 \end{bmatrix}$$

This enables us to understand the meaning of these vectors. y_1 is mostly related to a decreasing temperature and pressure, and y_2 represents decreasing humidity and pressure.
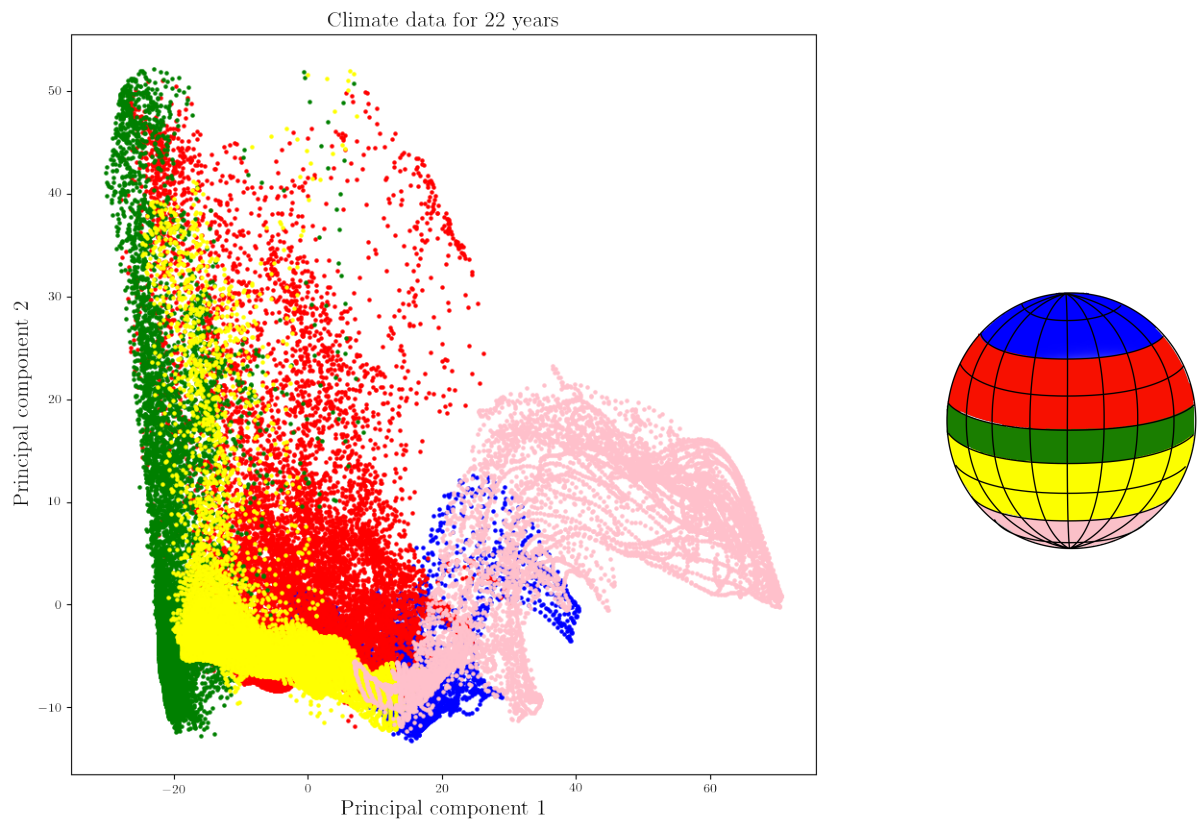

FIGURE 7 – PCA with two components

North	Temperate north	Equator	Temperate South	South
🔵	🔴	🟢	🟡	🌸

We can see on the graph that even if there is a strong dispersion, equator values are located at higher temperatures. On the contrary, north and south pole values are located at lower temperatures. Green and red classes overlap as these to categories have similar climate conditions.

Although it is an understandable figure, this is not satisfying.

4.1.2 Three dimensions

We attempted to run the PCA method with three principal components, it yielded similar results as linear PCA, but some areas with particular climate conditions stood out more strikingly. For example, deserts are located at the bottom of the peak on the third figure below, that maps the dry and hot areas.
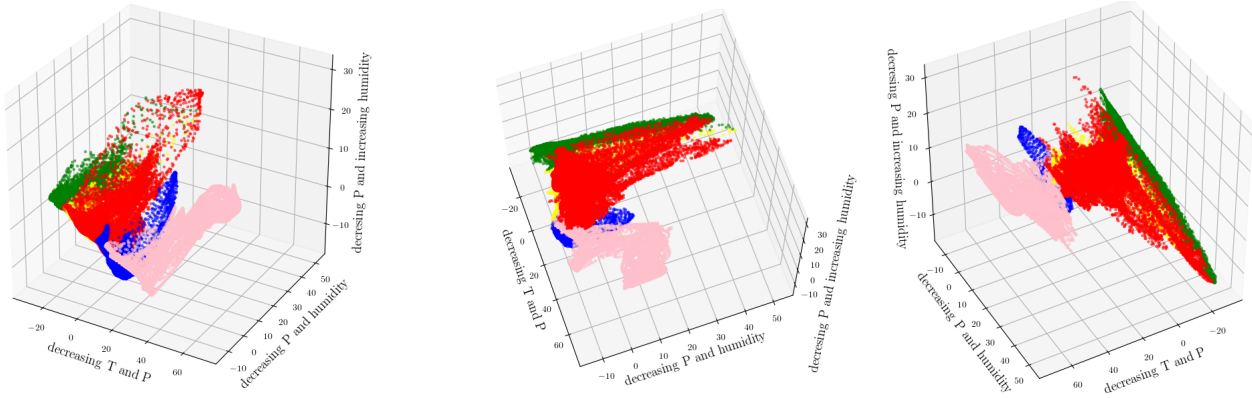

FIGURE 8 – 3D PCA

4.1.3 Kernel PCA

Kernel PCA has a complexity of $O(n^3)$ so we decided to run it on a subset of our dataset. Here is the two dimension result for all latitudes between longitudes -15 and +15.
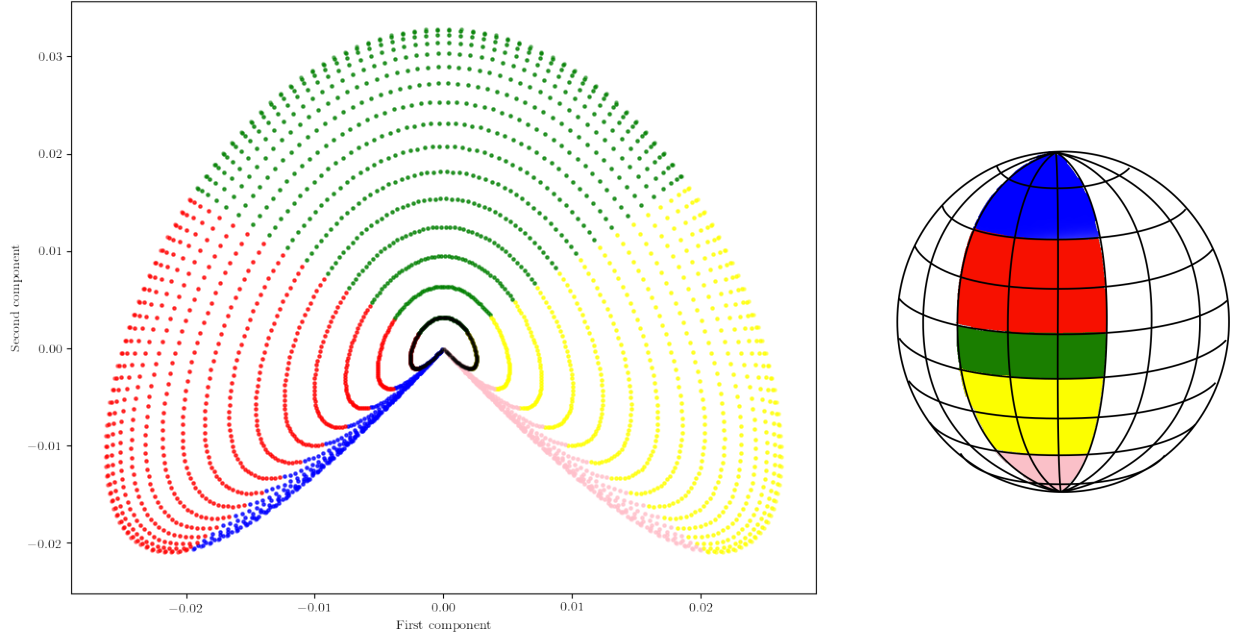
FIGURE 9 – Kernel PCA with two components and the RBF kernel

The black dots are all dots along one longitude, here 15. We see however that there is a degenerescence : -15 and +15 longitude are projected on the same positions. When changing the γ, some lines grow appart but most of them stay degenerated. Another flow of this representation is the space that is taken by the equator area. When representing the temperature in this area, we can see it does not vary very much compared to other regions : the variables should be represented closer together, as it was the case for example in PCA.

The peak present in the center of the figure is a consequence of the fact that data from each side of the peak even if they are close numerically are not close spatially : they are the data from both north and south pole.
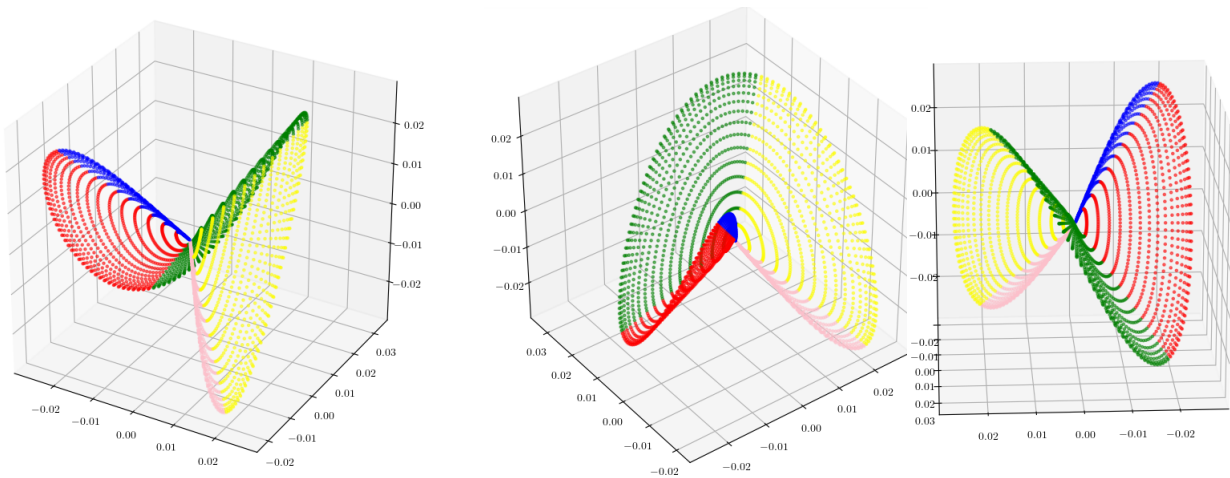
Figure 10 – 3D Kernel PCA

The dots representing the south and north poles are still closer together however, which might come from the smaller range of temperatures and climate conditions reached in these areas.

Changing the kernel function to a polynomial one however destroyed the degenerescence, but created a node on equatorial values as it can be observed just below. The purple and black lines represent the -15 and 15 longitudes : they are clearly different.
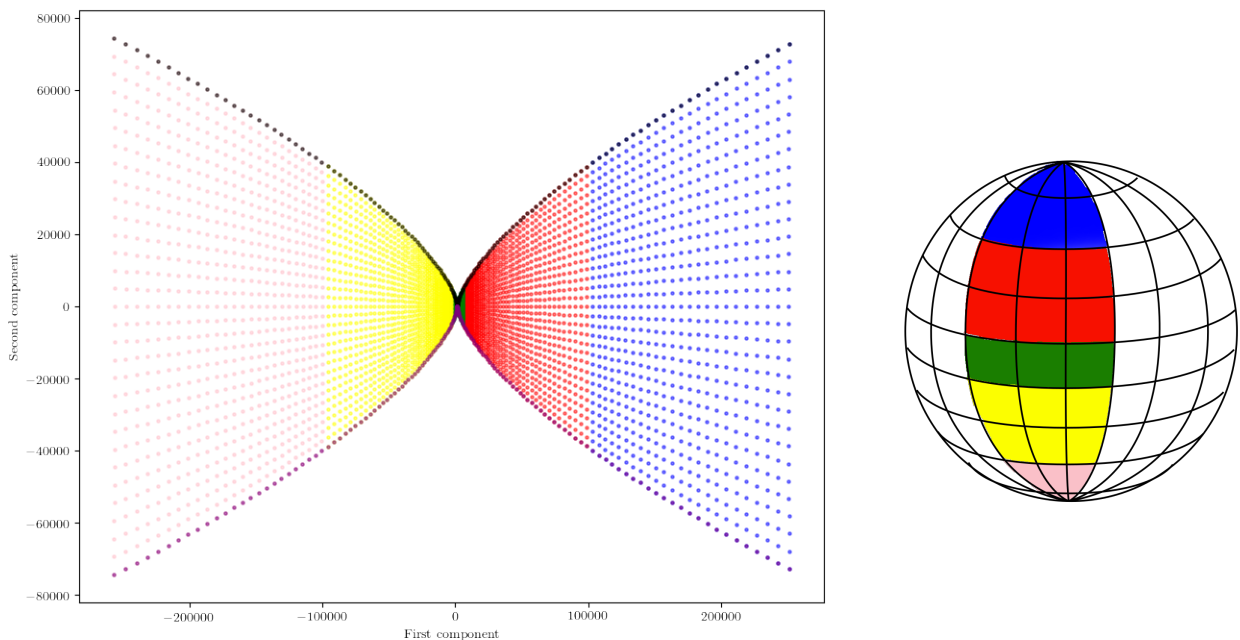

Figure 11 – Kernel PCA with two components and polynomial kernel

4.1.4 Multidimensional Scaling - MDS

As MDS was extremeley low and seemed to use a lot of memory, we could not run the programm for more than 600 datapoints (ie 600 rows).Therefore, we used a new subset of longitudes between 0° and +5°, reducing the dataset size to 546 datapoints. Figure 6 shows the results of MDS. As we can see, datas from South pole and North pole are quite compact, that is to say that all points are near each other. There is not a wide range of temperature, pressure, radiation, humidity and wind speed in those regions. This is quite what we expected. On the contrary, regions like equator and north temperate show a wide range of climatic parameters as some points are quite far from each other. It would mean that there are more diverse climate in equator and north temperate. It also seems like south temperate shows less diversity of climate as all the points seem compact. This is not what we expected, as south and north temperate are supposed to have same characteristics. This could be explained by two reasons. **Either we are missing information because we have reduced too much the dimension, or south hemisphere is quite different because it has much more ocean than the north hemisphere.** To check the first possibility, we implemented MDS in 3D, here are the results on figure ??. As we can see, the yellow points (south temperate) are not as widely spread as the green (equator) or red (north temperate) one. So it seems like the second possibility explain why we have fewer variance of climatic parameters in south temperate.
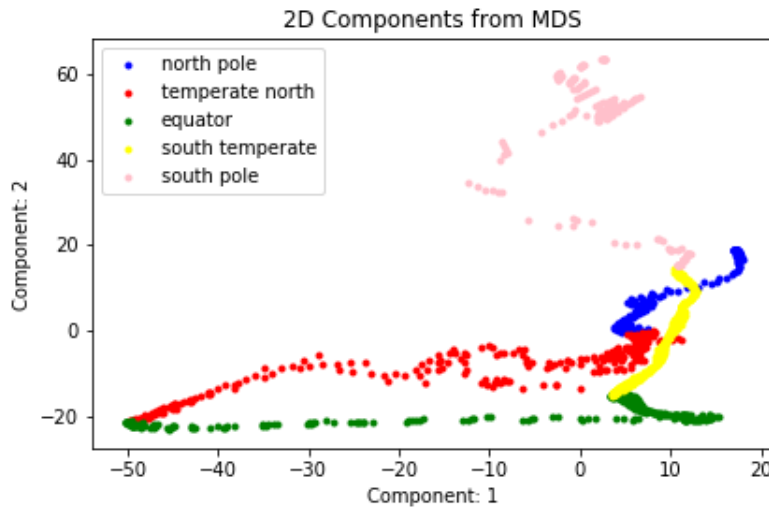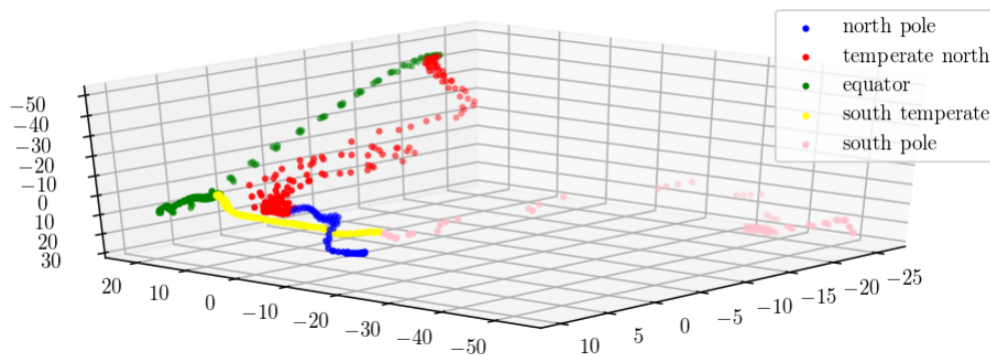

FIGURE 12 – Results of MDS in 2D

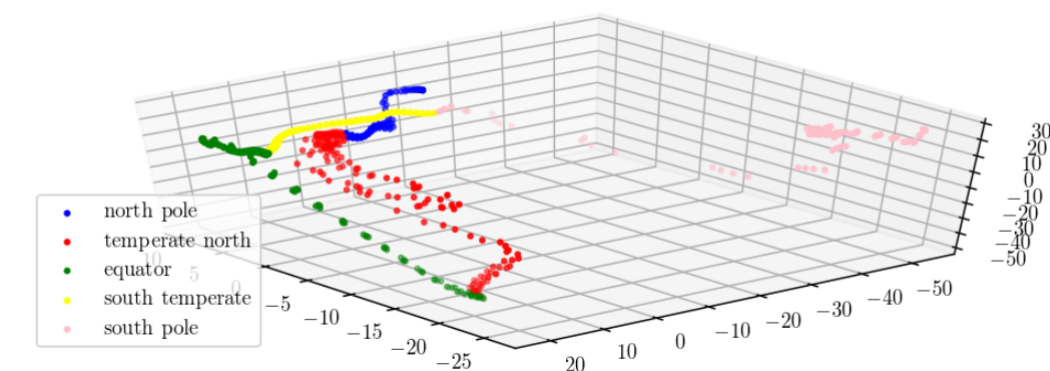Figure 13 – Results of MDS in 3D

Figure 14 – Results of MDS in 3D

4.1.5 Isomap

One of the key parameter of isomap is the number of nearest neighbors taken into account. Therefore, we present in figure ?? the result of isomap for number of neighbour between 1 to and neighbors.
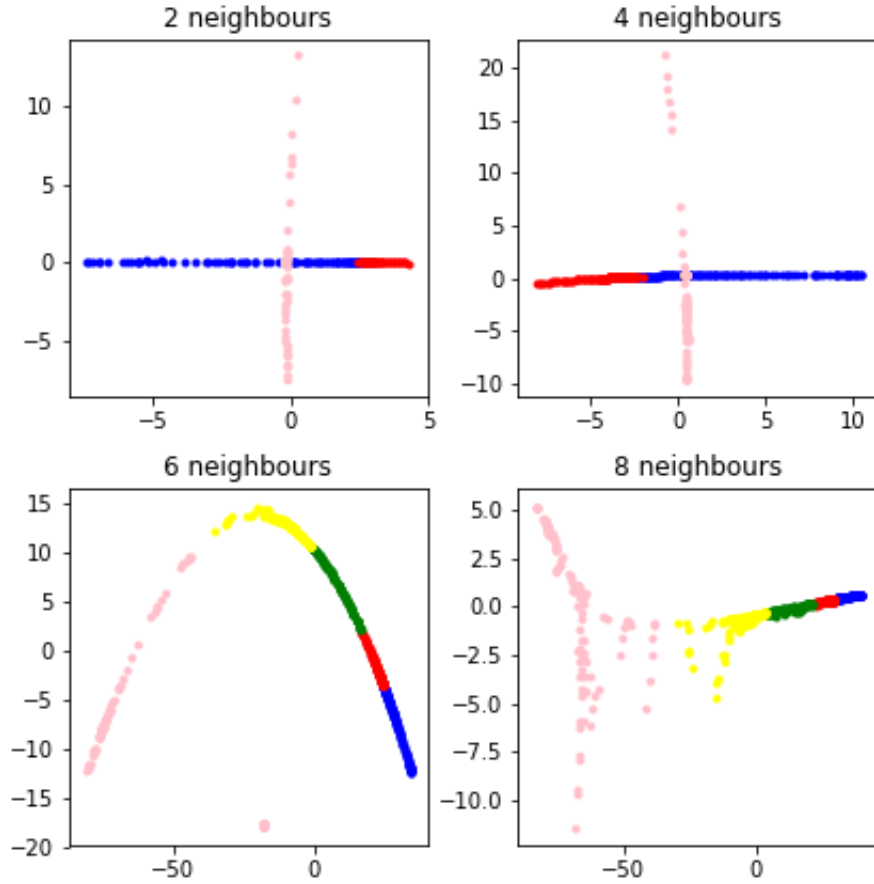
FIGURE 15 – results of isomap depending on the number of neighbors considered from 2 to 8

It is obvious that the number of neighbours considered strongly influence the 2D distribution of our dataset. To know which graph is the most accurate, we refer to the 'reconstruction error'. This reconstruction error is a function embedded in scikit. We will detail it later in section 4. For the moment, we only need to know that it is an indicator of how much isomap mapping is accurate and respect original distance. We computed this reconsruction error with respect to the number of neighbours considered. The results are presented in figure ??
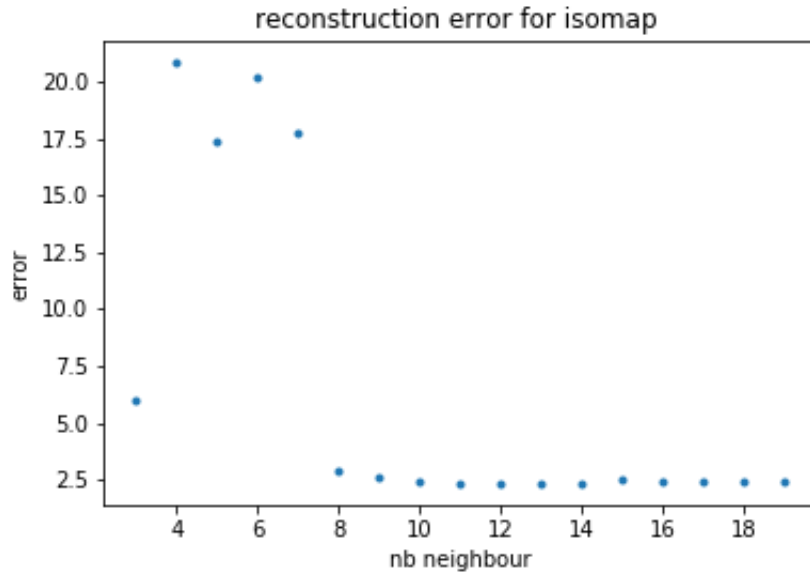
FIGURE 16 – reconstruction error as a function of number of neighbours considered

It is interesting to see that from eight neighbors, error seems to stabilize around 2,5. So the most 'accurate' representation should be with 8 neighbors. If we interpret the graph with 8 neighbors, we can see that we can browse the entire planet from north pole to south pole, going through all zones in the order, **which is not the case in MDS**. So isomap better preserve the geometry of the dataset. It is interesting to see that pink points(ie south pole) are much more spread in isomap than MDS.

4.2 Time comparison

We implemented manually PCA, but for the other methods we used the built in fonction of the scikit package on python. PCA's running time varies linearly with the dataset size as we can see on the graphic below
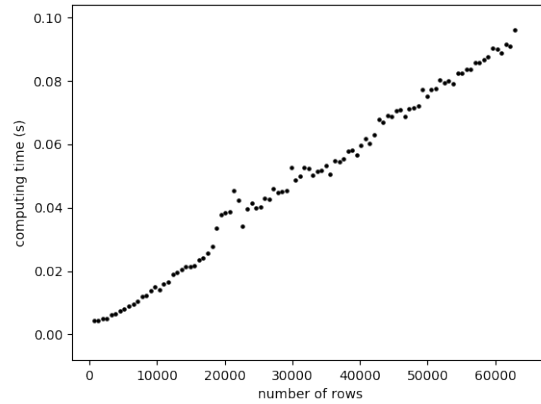
FIGURE 17 – running time for PCA

Kernel PCA was slower, we did not compute its running time for all the dataset, only until 12000 rows. It yields the following graph.
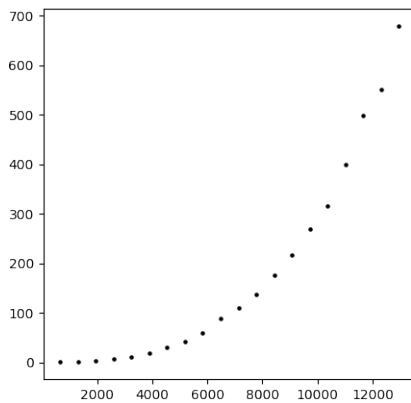

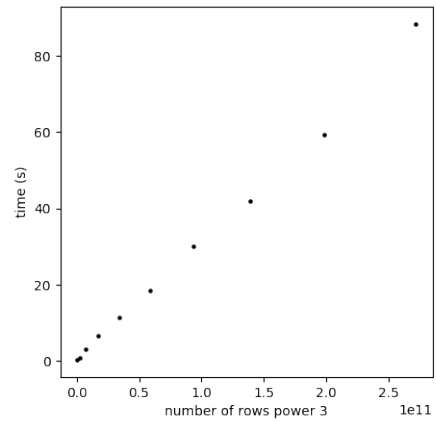FIGURE 18 – running time for Kernel PCA

FIGURE 19 – Kernel PCA running time is a $O(n^3)$

In figure 10, we present computing time for the four different methods as a function of the sample size. From this we can conclude that the quickest method is PCA. Then come isomap, kernel PCA and finally MDS. It is quite surprising to see that isomap is quicker than MDS for the same number of rows. Indeed, isomap uses the same algorithm than MDS but also needs to compute distance with neighbors first. It is thus expected to be slower. We found an explanation on the 'manifold guide' of skicit.

In Scikit-Learn implementation, Isomap algorithm runs faster that Multi Dimensional Scaling on the S-Curve dataset [...]. In the third stage

of algorithm, the implementation uses Partial Eigen Value decomposition instead of MDS which is the version proposed by the researchers

We did not use the 'S curve dataset' as mentioned above, but we may think that due to the round shape of earth, our dataset is quite similar to the S-curve and thus Isomap runs faster than MDS. For a fixed dimension d, the following table present the complexity of every method as it appeared to us.

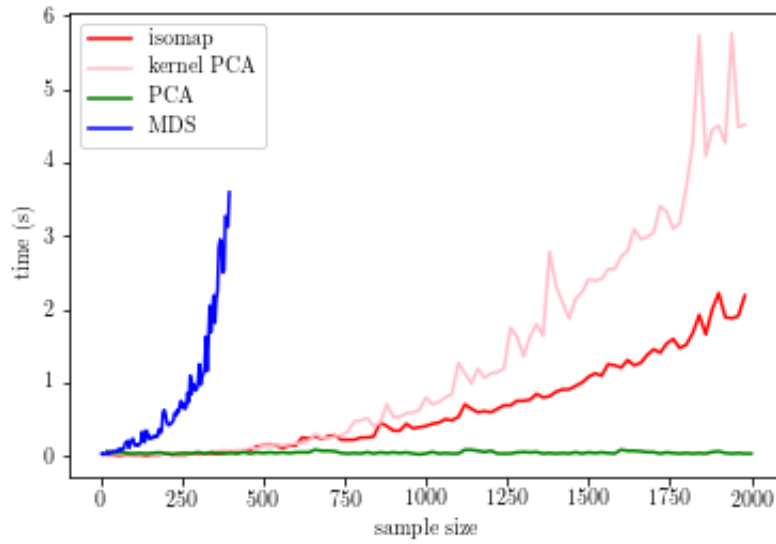| Method	PCA	Kernel PCA	MDS	Isomap
Complexity	$O(n)$	$O(n^3)$	$O(n^3)$	

FIGURE 20 – computing time comparison

We also check how isomap computing time varies with the number of neighbor taken into account. Figure 11 shows that isomap efficiency time increases with the number of neighbors. So isomap efficiency of computing time may be limited by the number of neighbors.
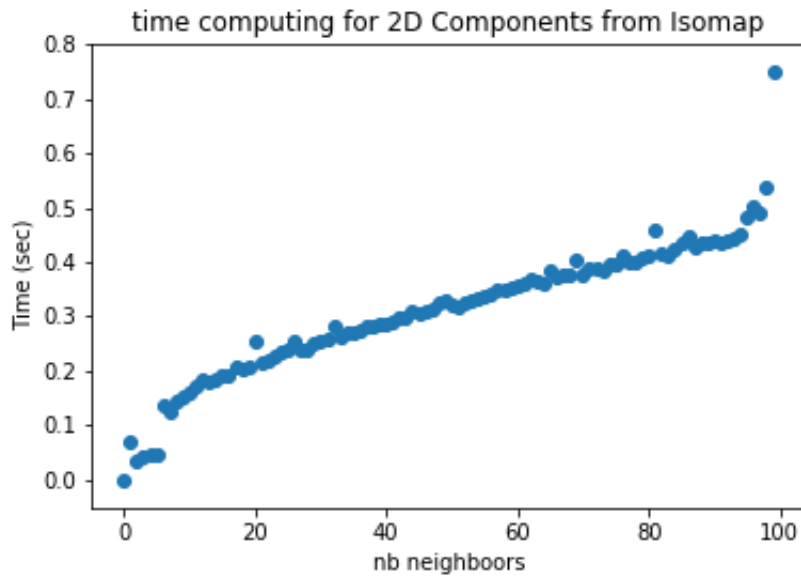
FIGURE 21 – computing time for isomap as a function of the number of neighbors considered

4.3 Error and preserved information

4.3.1 PCA

The main indicator of how much information has been preserved with PCA lies in eigenvalues. We plot the eigenvalues to see their relative importance in the dimension reduction.
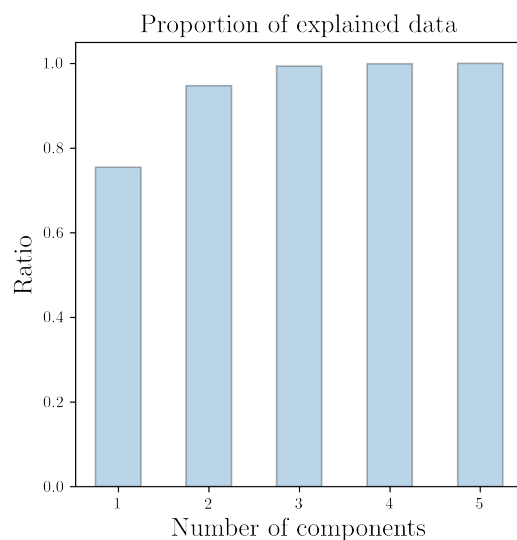
FIGURE 22 – PCA with two components

We can see that with only two components, we obtain an explained variance of 94%, which is quite enough to interpret our dataset. PCA seems to be a relevant technic to analyse data, and in particular to establish classification.

We also reconstructed the dataset from its projection with PCA. It is indeed possible to build an estimated $\hat{X}$ dataset from the projected data of PCA. We plotted the first column of the dataset, the temperature for X and $\hat{X}$ as a function of latitude and longitude. Even if the curve is reversed and the numerical values are not correct, the structure of the data is saved.
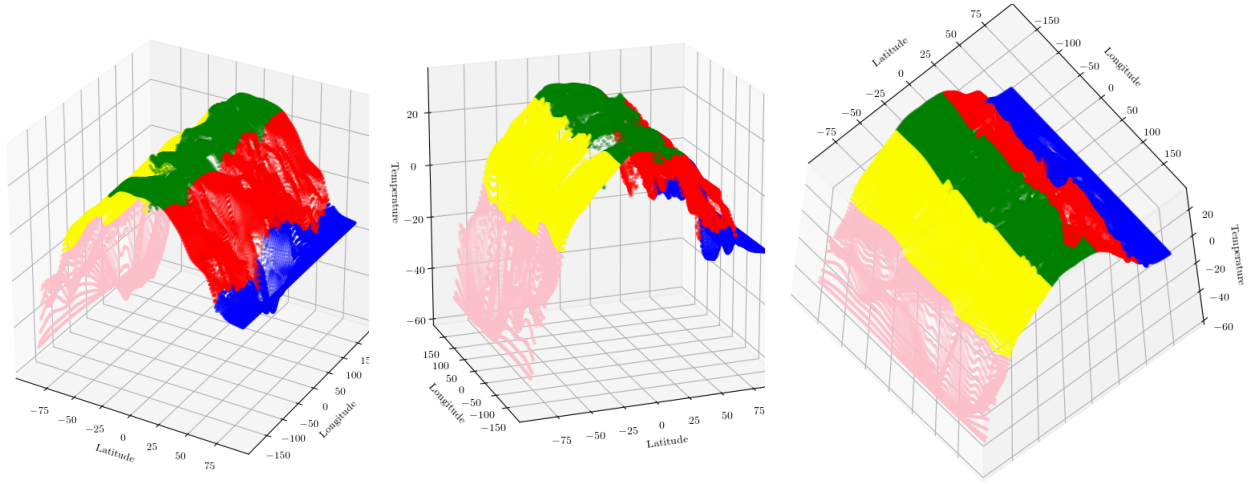
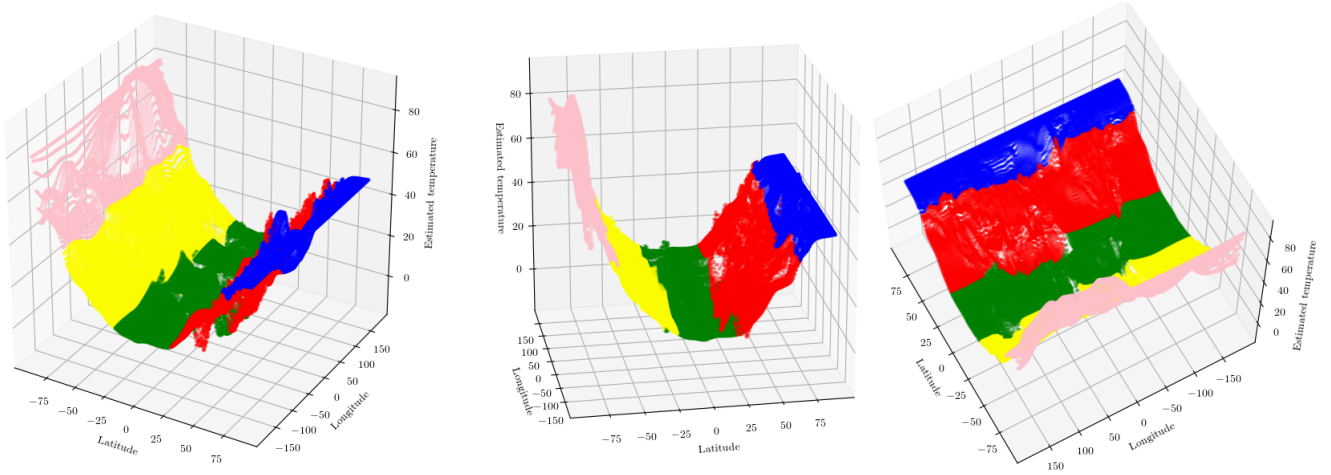FIGURE 23 – Temperature as a function of latitude and longitude in X

FIGURE 24 – Temperature as a function of latitude and longitude in $\hat{X}$

4.3.2 MDS and isomap

As isomap and MDS function with the same principle, we decided to compare their estimated error. For MDS, we ued an indicator called stress. It is roughly the square difference between the final distance of two items in the MDS model and the true distance between them. The exact formula is :

$$s = \sqrt{\frac{\sum(d_{ij} - d(i,j))}{\sum d_{ij}}}$$

with d_{ij} the original distance between pairwaise i and j and d(i,j) the computed distance. The aim of MDS is to minimize this stress. For isomap, there is an indicator called error reconstruction already embedded in scikit. We will not detail its formula here as it is quite heavy and complicated. All we need to know is that it can be compared to the notion of stress as in MDS in the sense that it computes the difference between the final distance of two items in the MDS model and the true distance between them.
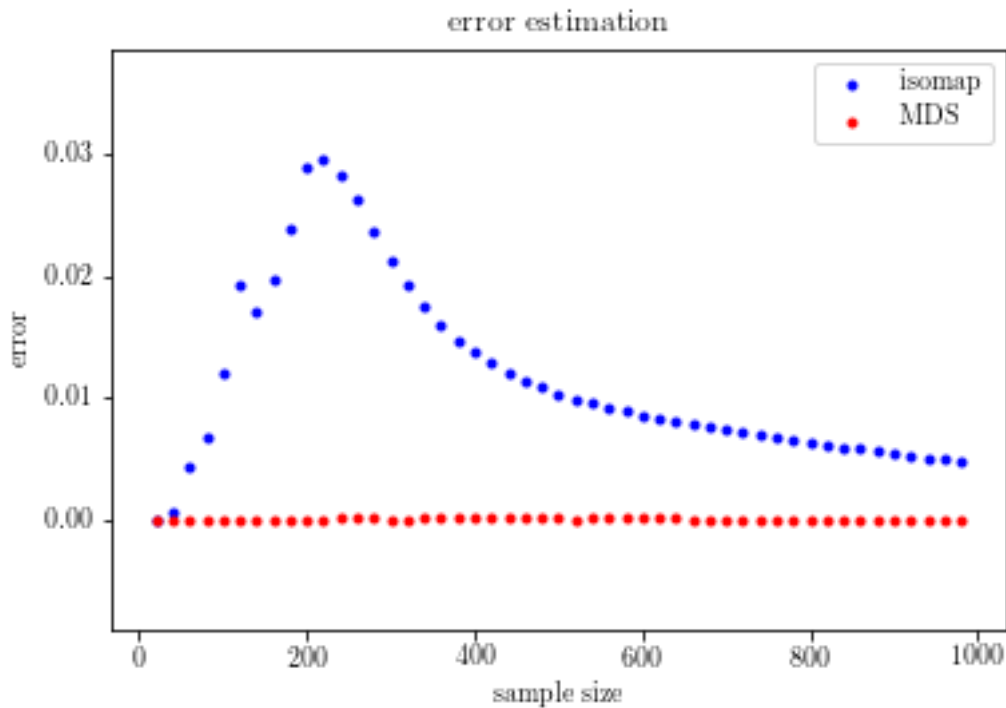

FIGURE 25 – stress as a function of sample size

As we can see, isomap is less efficient than MDS for a subset of size 200. AS the data set size increases, isomap error tends to lessen but is still much more important than MDS error. In our example, it seems like MDS gives a much more accurate representation of our dataset.

Then, we intended to see how reconstruction error evolve with the number of dimension taken into account for both MDS and isomap. On figure ?? we compare how reconstruction error evolve with dimension. We make dimension vary between 2 and 3.
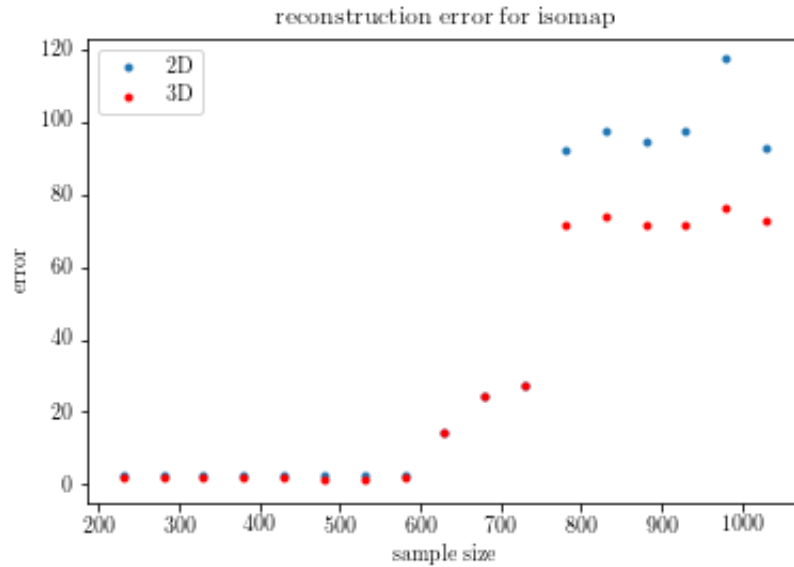
FIGURE 26 – reconstruction error as a function of the sample size for ISOMAP

As the sample size increases, error for 2D isomap keep increasing compare to error in 3D isomap. As expected, error for 2D isomap is bigger than for 3D because we reduce more the dimension.

We did the same for MDS in 2D and 3D.

5 Conclusion

6 Bibliography

https ://eosweb.larc.nasa.gov/cgi-bin/sse/global.cgi ?email=skip@larc.nasa.gov ↑[3]