LOW DIMENSIONAL EMBEDDING OF ENVIRONMENTAL VARIABLES

EA MAP581

13 mars 2018

Flore Martin and Lorraine Roulier

ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Table des matières

1 Introduction

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. 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.

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. 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. These variables are gathered in the table below

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

Figure 1 – First row of our dataset

The latitude parameter varies from -90 to 89 and the longitude parameter varies from -180 to 179. the negative values are for the south hemisphere, the positive ones for the north. 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 differenciate climate data was the latitude.

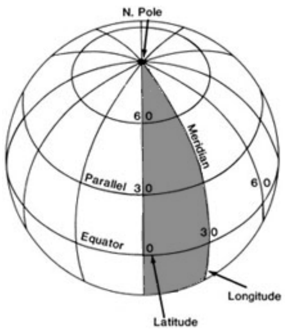

Figure 2 – An example of subset in grey

We classified the data according to the latitude, creating five classes listed in the table below :

	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 – First lign of our dataset

2 Principal Component Analysis - PCA

2.1 Method

Principal Component Analysis detects tendencies in the data by maximizing the variance of the dataset matrix. This 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, PCA maximizes the following equation :

$$\|X - MM^TX\|^2$$

subject to $M \in \mathcal{O}^{d \times r}$ where $r < d$.

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 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

2.2 Results

2.2.1 Two dimensions

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

$$y_1 = \begin{bmatrix} -0.94220902 \\ -0.31329122 \\ 0.10889051 \\ -0.04573547 \\ 0.01191187 \end{bmatrix}$$

and

$$y_2 = \begin{bmatrix} 0.02637512 \\ -0.40428947 \\ -0.9117768 \\ 0.04144017 \\ -0.05291649 \end{bmatrix}$$

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.

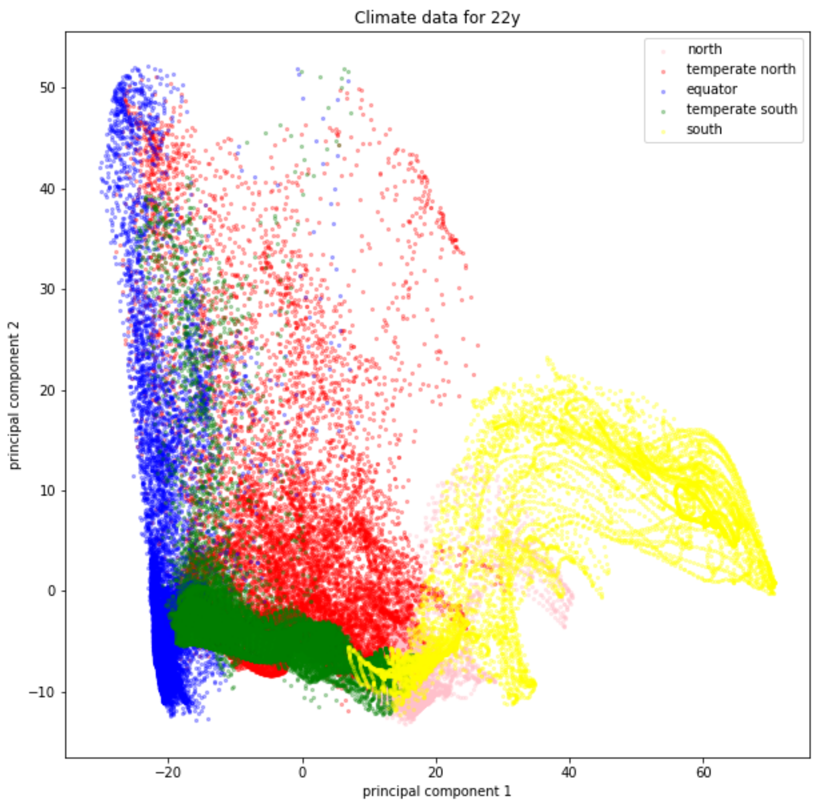

Figure 4 – PCA with two components

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. We plot the eigenvalues to see their relative importance in the dimension reduction.

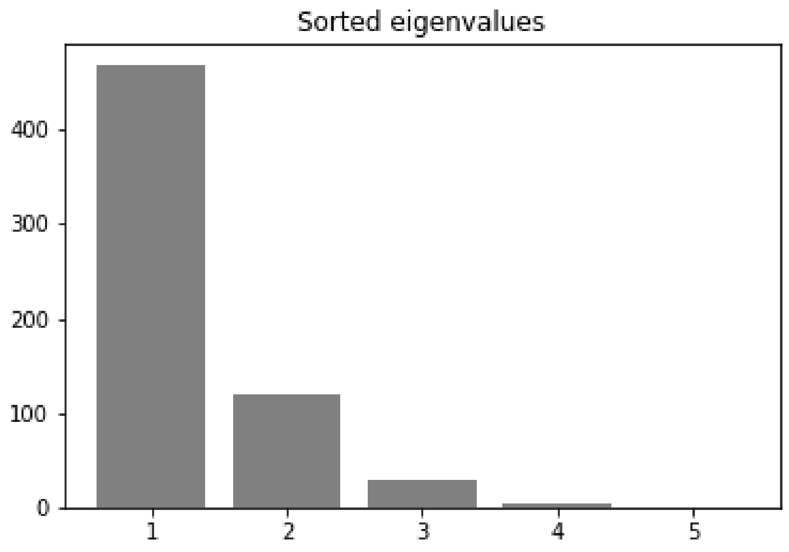

Figure 5 – PCA with two components

2.2.2 Three dimensions

We attempted to run the PCA method with three principal components,

3 Kernel Principal Component Analysis

3.1 Method

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 ϕ.

Instead of directly mapping X into the feature space, we use a kernel, that represents a similarity function between pairs. Let K be this kernel, then

$$K(x_i, x_j) = \phi(x_i)\phi(x_j)^T$$

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 :

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

3.2 Results

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.

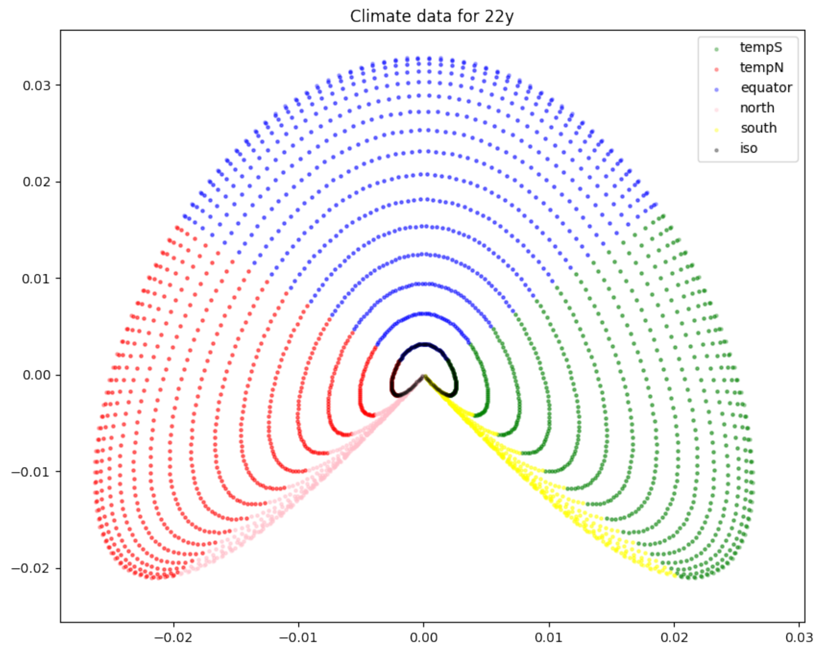

FIGURE 6 – Kernel PCA with two components and the RBF kernel

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 gamma

Another noticable flaw is the peak on the graph when we would expect the dots of same longitude to form a circle. The 3D visualisation of the graphic enables us to understand that peak : as we can see on the figure below, the lines that define one longitude form a sort of infinity symbol, a twisted circle, that can't render smoothly in two dimensions.

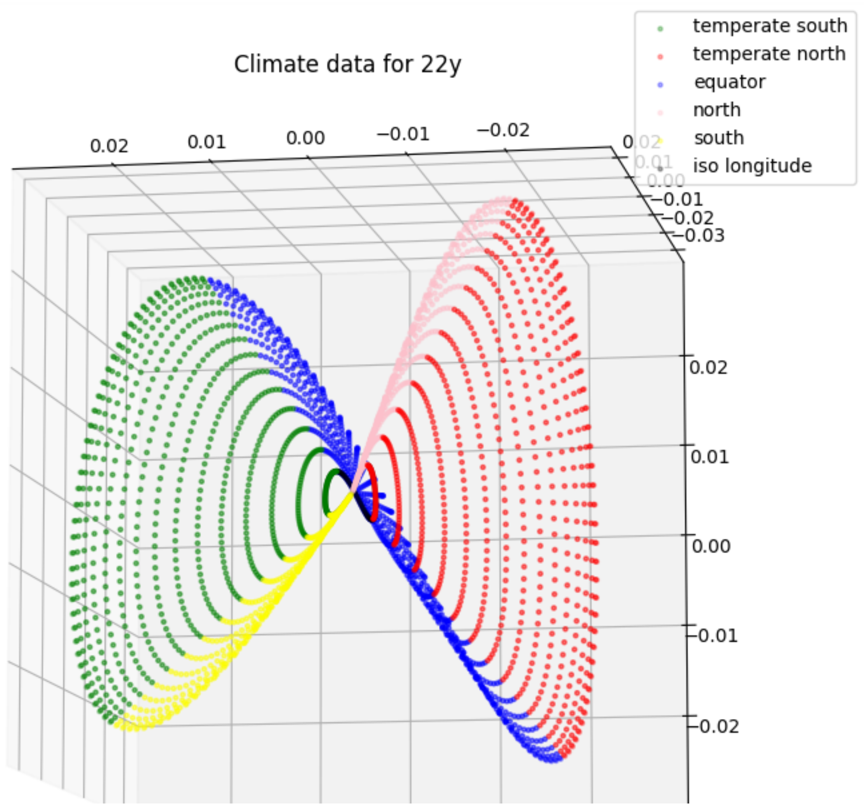

Figure 7 – 3D Kernel PCA

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.

We tried to change the γ parameter, but it had no influence on the degenerescence. 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.

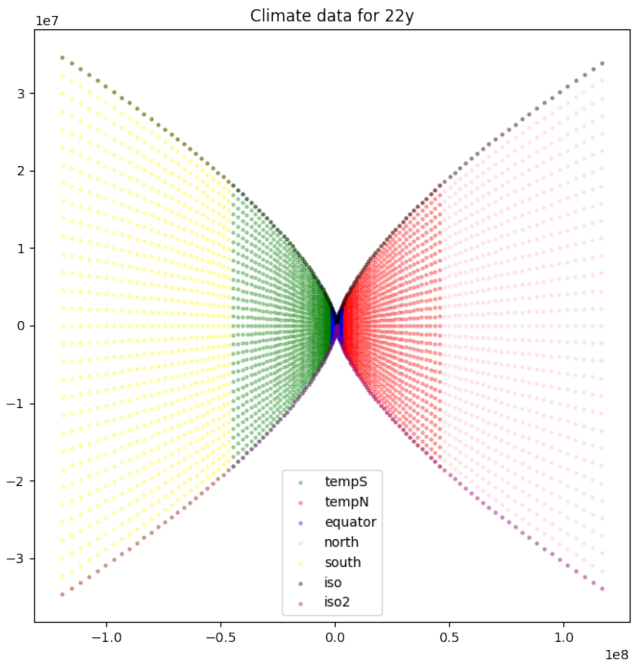

Figure 8 – Kernel PCA with polynomial kernel

4 Multidimensional Scaling - MDS

4.1 Method

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

As the distance matrix $D^{(X)}$ can be converted into a kernel matrix of inner products X^TX by

$$X^TX = -1/2HD^{(X)}H$$

H can be written as $H = I - ee^T$ and e is the vector of ones. Thus, ? ? ? becomes :

$$\sum_{i=1}^{t}\sum_{j=1}^{t} x_i^Tx_j - y_i^Ty_j$$

It can be shown that the solution is $Y = \Lambda^{1/2}V^T$ with V the eigenvectors of X^TX of the top d eigenvalues present in Λ. The only and main parameter of this method is the norm used to calculate the distance matrix. By default we use the euclidean norm here.

4.2 Results

Figure 6 shows the results of MDS with our previous dataset. We use the same grandeurs (temperature, radiation, wind speed, humidity and pressure) but not the same number of lines. As MDS was extremeley low and seemed to use a lot of memory, we could not run the programm for more than 600 rows. Here are the result for 546 rows : we filtered our dataset with latitudes < 0 (ie. south hemisphere) and longitude between 0 and 5 degrees.

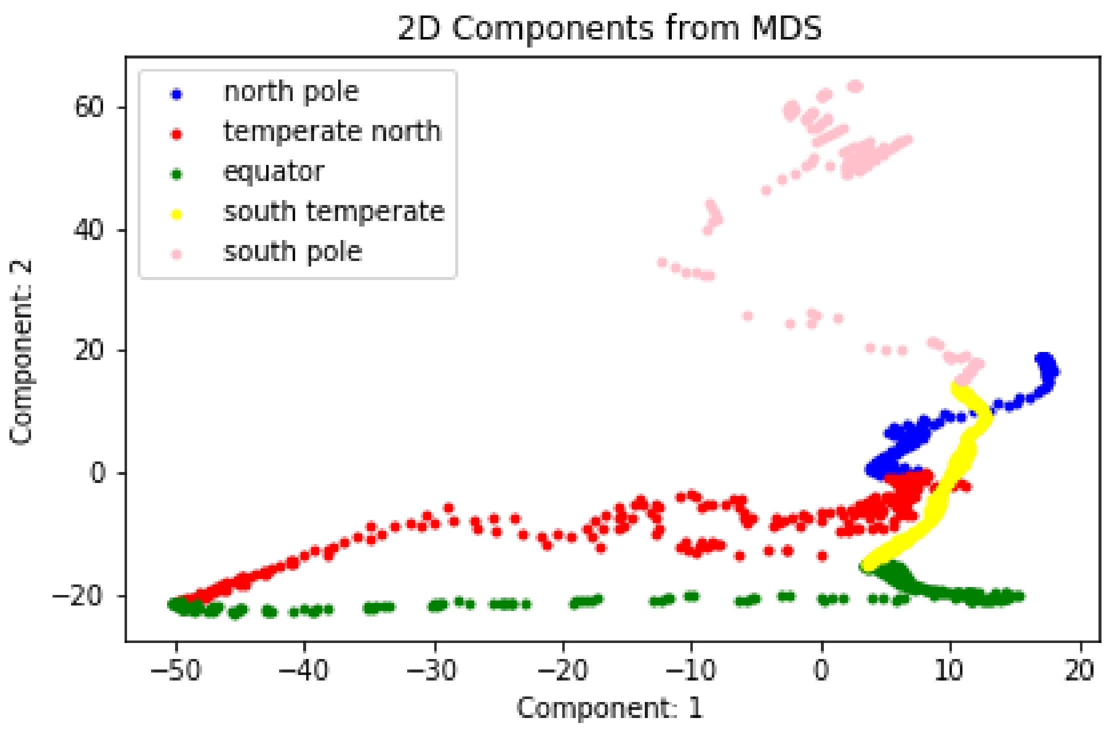

Figure 9 – results of MDS in 2D

One indicator to estimate of the reconstruction is accurate is the '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 :

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

The aim of MDS is to minimize this stress. Figure 7 presents the value of stress depending on the number of lines. We can see it is relatively low (<0,002) for more than 500 rows.

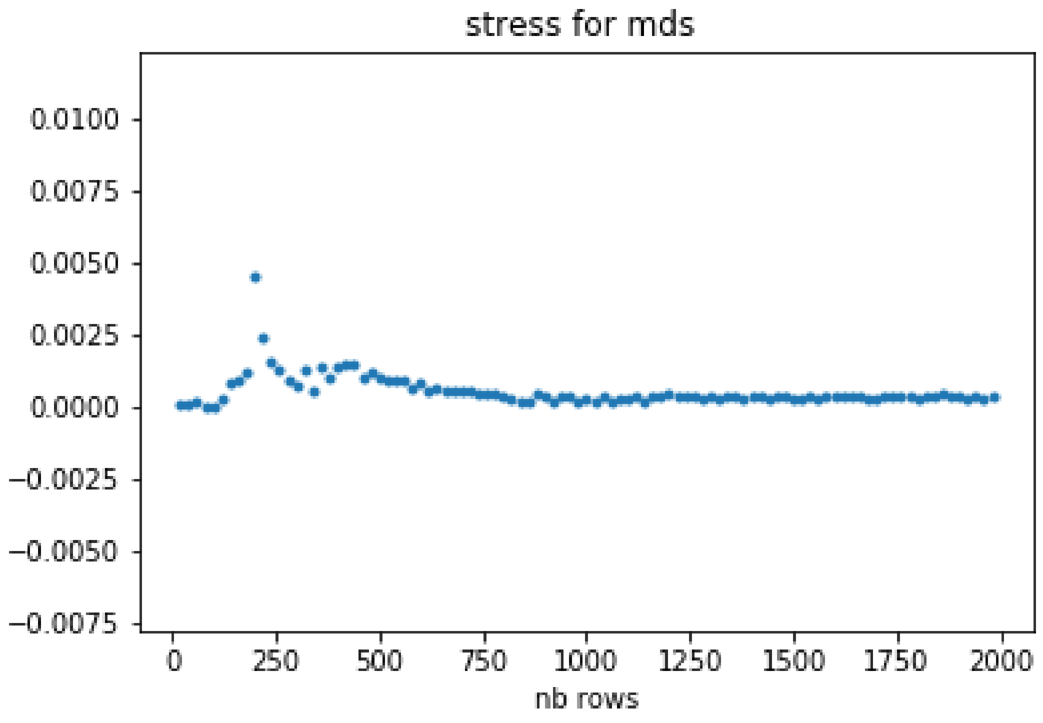

Figure 10 – stress as a function of number of rows

5 Isomap

5.1 Method

Isomap is a low dimensional embedding method similar to MDS. The difference is, distance is not computed with euclidean norm but with geodesic distance.

5.2 Results

As the main parameter of the method is the number of nearest neighbors taken into account, we present infigure 8 the result of isomap for number of neighbour from 1 to 11 neighbors.

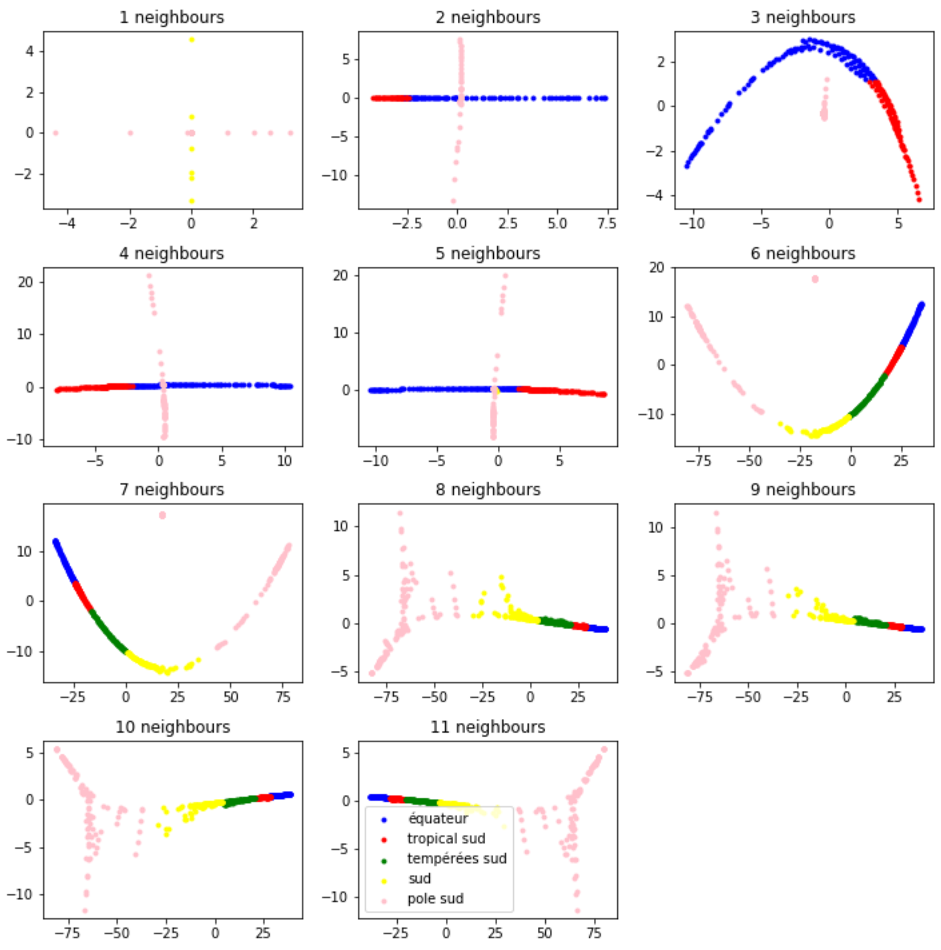

Figure 11 – results of isomap depending on the number of neighbors considered from 2 to 12

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'. We computed this reconsruction error with respect to the number of neighbours considered. The results are presented in figure ? ?

We also computed the error reconstruction as a function of the number of rows in figure 9. This error keeps increasing with the number or rows, so isomap accuracy seems to be limited by the size of the dataset.

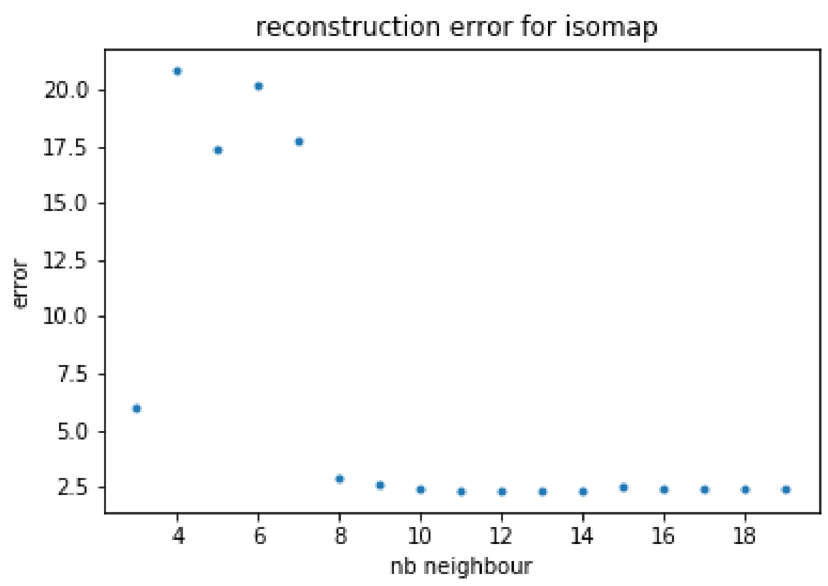

Figure 12 – reconstruction error as a function of number of neighbours considered

It is interesting to see that from eight neighbours, error seems to stabilize around 2,5. Indeed, we can see with graphs of figure 8 that from eight neighbours, graphs look quite all the same. Those are the most 'accurate' representations.

6 Comparing the different methods

6.1 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 number of rows as we can see on the graphic below

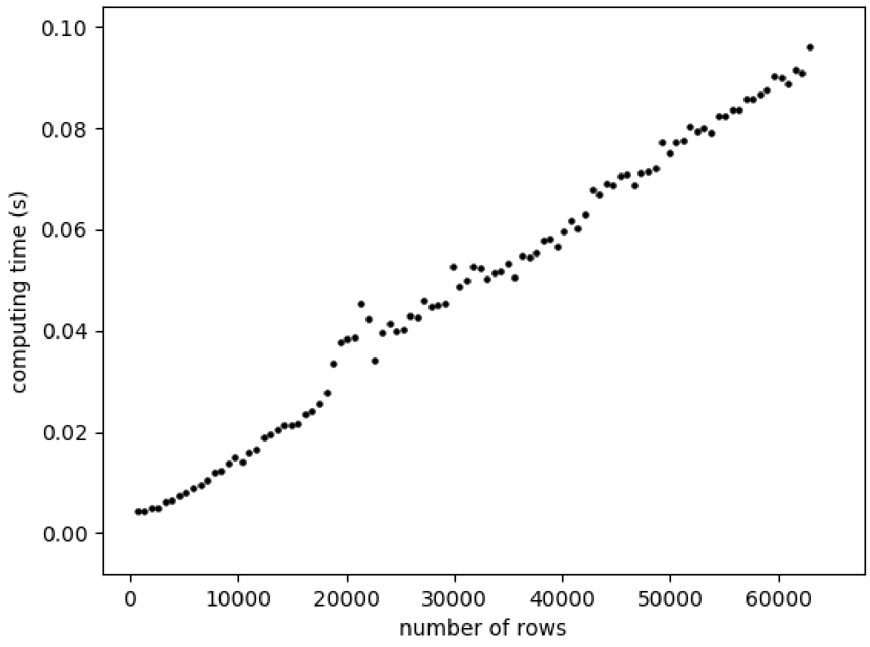

Figure 13 – running time for PCA

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

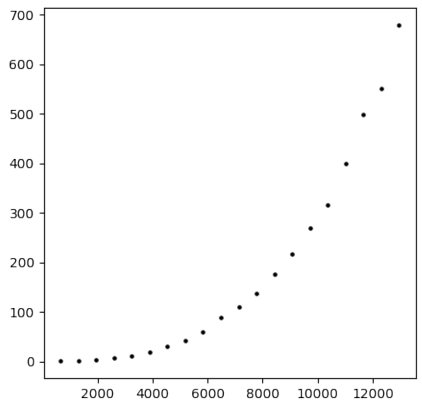

Figure 14 – running time for Kernel PCA

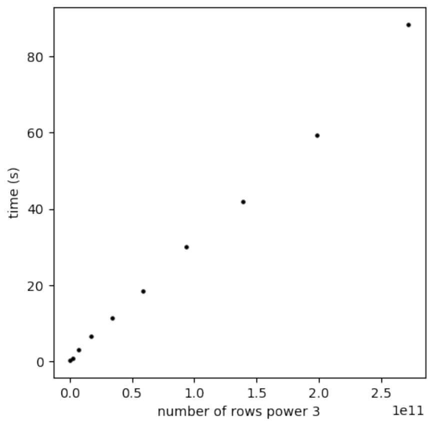

Figure 15 – Kernel PCA running time is a $O(n^3)$

In figure 10, we present computing time for different methods as a function of the number of rows. From this we can conclude that the quickest method is PCA. Then comes kernel PCA, isomap and 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.

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

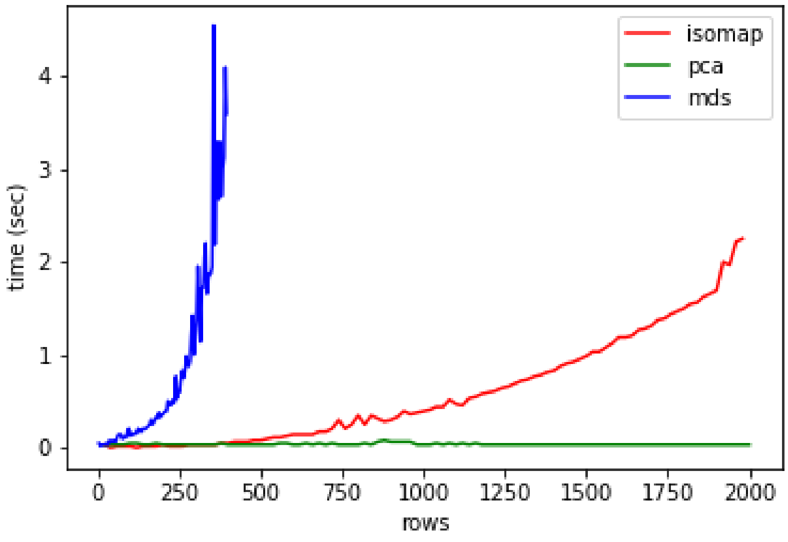

Figure 16 – computing time comparison

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.

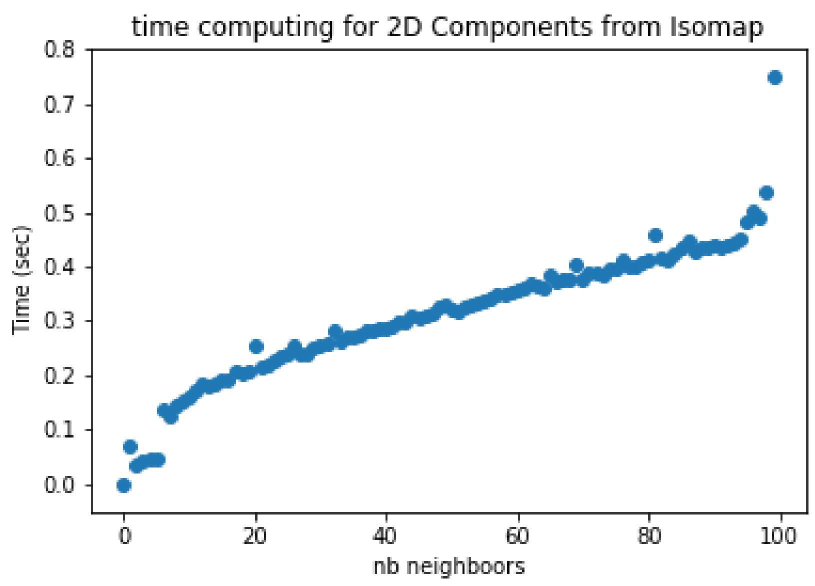

Figure 17 – computing time for isomap as a function of the number of neighbors considered

6.2 Error and preserved information

7 Conclusion

8 Bibliography