LOW DIMENSIONAL EMBEDDING OF ENVIRONMENTAL VARIABLES

EA MAP581

March 19, 2018

Flore Martin and Lorraine Roulier

Contents

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. 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 samples are being clustered, it is a way to establish clusters of gene and of population for example. It is a current tool in machine learning.

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

In this project, we explored various dimension reduction techniques. The main way to classify the different methods is to observe if they are linear or non-linear. The basic linear dimension method is a linear one, Principal Component Analysis (PCA). In order to familiarize ourselves with dimension reduction, we implemented this method manually. We explored PCA and its non-linear version, Kernel Principal Component Analysis. We also used Multidimensional scaling (MDS) which is a linear method and Isomap, a non-linear method.

2 Dataset presentation

We applied the methods listed above on the dataset we describe here. It is a dataset that contains climate data associated with its position. Our goal is to show that the position on the globe is embedded in the climate data.

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 [**??**]. 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. We did not select every dataset given by the NASA because we considered that temperature related data for example were already very correlated and did not add much information.

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 samples, 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.

Since our goal was to see if the position of a location on earth was embedded in it's climate data, we did not use the first two columns of our dataset when computing the dimension reduction methods.

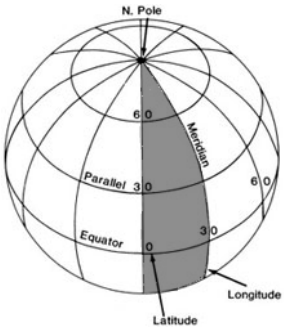

Figure 2: An example of subset in grey

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. The latitudes we chose are universally know parallels. The north temperate zone extends from the tropic of cancer to the arctic circle and the south temperate zone from the tropic of capricorne to the antarctic circle.

	North	Temperate North	Equator	Temperate South	South
Latitudes	90 to 66	66 to 23	23 to -23	-23 to -66	-66 to -90
	●	●	●	●	●
% of Earth surface	4	26	40	26	4

Figure 3: Our classification

3 Methods

In all dimension reduction techniques, 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 can see the plot of a 2D-dataset. If we implement 2D-PCA, we will find two new axes that reflect at best the covariance of the samples. 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.

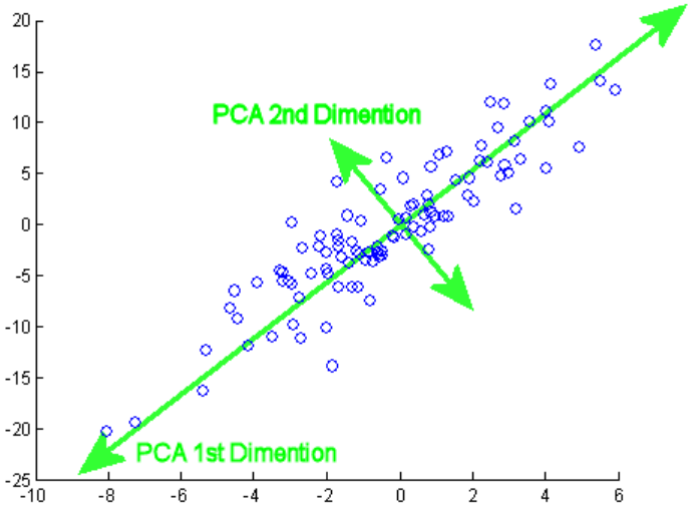

Figure 4: role of PCA - interpretation

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. The dataset can then be projected on a lower dimension using the right eigenvectors.

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 orthonormal 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. Each of the M_{ij} coefficients of the matrix can be interpreted as weights given to the variable they are associated to. This enables us to interpret which variable of the dataset gives more significant information about the data.

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 $\phi : \mathbb{R}^d \to \mathbb{R}^N$ where $N > d$.

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

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

where $(x_i, x_j) \in \mathbb{R}^d \times \mathbb{R}^d$ 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:

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

3.3 Multidimensional scaling

Multidimensional scaling is a linear approach to reduce dimension of a dataset. The principle is different from PCA: given a matrix of distance or "dissimilarity", MDS aims at reconstructing a map preserving at best the distances between datapoints. In our example, the MDS algorithm aims to place each object in a 2-dimensional space such that the between-object distances are preserved as well as possible.

Given a distance matrix D of dimension $d \times d$ (d=5 for us) , MDS attempts to find the datapoints y_1,... y_n in dimension $r < d$(r=2 for us) that minimizes:

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

with $d_{ij}^{(X)} = \|x_i - x_j\|^2$ the Euclidean distance between pairwise i and j in the original $d \times d$ matrix $D^{(X)}$ and $d_{ij}^{(Y)==\|y_i - y_j\|^2}$ the Euclidean distance between pairwise i and j in the computed $r \times r$ matrix $D^{(Y)}$. The new vectors v_1,... v_r created in reduced dimension r should be linear combination of the original vectors v_1,... v_d, which is why MDS is a linear method. In our example, original vectors v_1,... v_5 are temperature, radiation, pressure, wind speed and humidity.

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 enables us to find the coordinates of each city based on this distance matrix.

	Beijing	Cape Town	Hong Kong	Honolulu	London	Melbourne
Cape Town	12947					
Hong Kong	1972	11867				
Honolulu	8171	18562	8945			
London	8160	9635	9646	11653		
Melbourne	9093	10338	7392	8862	16902	
Mexico	12478	13703	14155	6098	8947	13557
Montreal	10490	12744	12462	7915	5240	16730
Moscow	5809	10101	7158	11342	2506	14418
New Delhi	3788	9284	3770	11930	6724	10192
New York	11012	12551	12984	7996	5586	16671
Paris	8236	9307	9650	11988	341	16793
Rio de Janeiro	17325	6075	17710	13343	9254	13227
Rome	8144	8417	9300	12936	1434	15987
San Francisco	9524	16487	11121	3857	8640	12644
Singapore	4465	9671	2575	10824	10860	6050
Stockholm	6725	10334	8243	11059	1436	15593
Tokyo	2104	14737	2893	6208	9585	8159

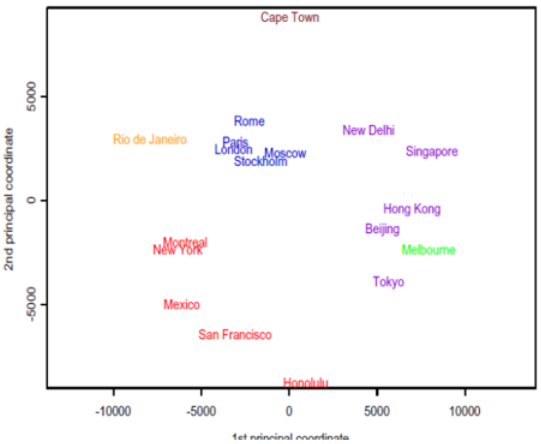

Figure 5: role of MDS - interpretation

3.4 Isomap

Isomap is a low dimensional embedding method similar to MDS. The difference is that distance is not computed with Euclidean norm but with geodesic distance. For this, we consider that each point in our dataset has a finite number x of 'nearest neighbors', that is to say the x nearest points computed with the Euclidean distance. To go from point A to point C in our dataset, one can only 'jump' from one of the nearest neighbor of point A (let's call him B), then one of the nearest neighbor of point B ... Until we reach point C. The geodesic distance is the sum of the distance browsed from nearest neighbor to nearest neighbor between point A and C. As the earth shape is round, isomap seems intuitively more relevant than MDS. On figure **??**, we can see the difference between the geodesic distance (in red) used in isomap, and the Euclidean distance used in MDS (in blue).

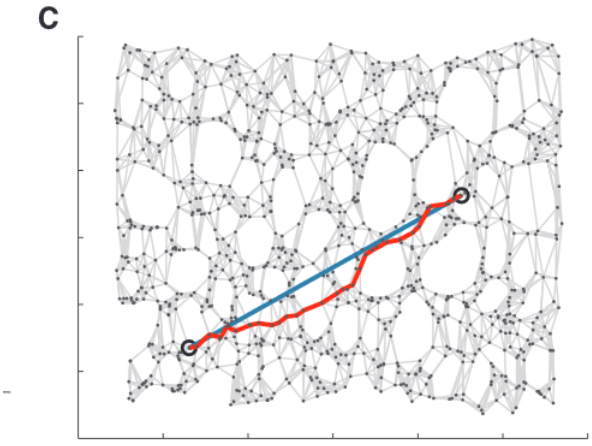

Figure 6: geodesic and Euclidean distance

4 Results

For each method, we aimed at reducing the 5D dataset into a 2D or 3D dataset so that we could plot it and interpret it. We analyzed how the five geographical zones were distributed across the map and how much information was preserved. Then, we compare the running time.

4.1 Graphic Interpretation

Graphic interpretation was the key analysis of this dimension reduction project. We plotted the results for each method in 2D or 3D to analyze climate variance and climate similarity across the earth. We also used graphic interpretation to check how accurate the method was at representing the dataset.

4.1.1 PCA

We first implemented PCA and ran it with only two principal components, which yielded the graph plotted in figure **??** for the whole dataset. We wrote the eigenvectors y_1 and y_2 that were used horizontally to make it easier to understand their meaning.

Temperature	Pressure	Relative Humidity	Wind Speed	Radiation

$$y_1 = [-0.94 \quad -0.31 \quad 0.11 \quad -0.04 \quad 0.01]$$

$$y_2 = [0.03 \quad -0.40 \quad -0.91 \quad 0.04 \quad -0.05]$$

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.

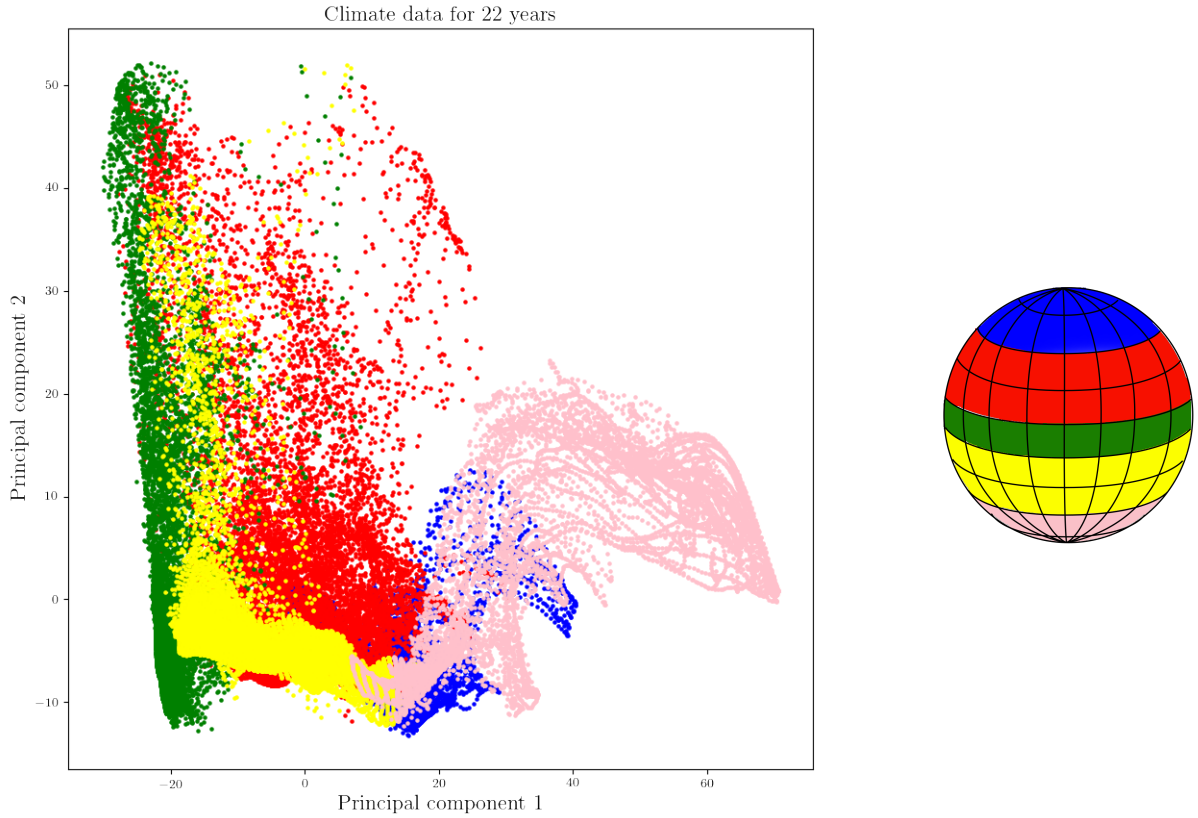

Figure 7: PCA with two components

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 strik-

ingly. For example, deserts are located at the bottom of the peak on the third figure below, that maps the dry and hot areas.

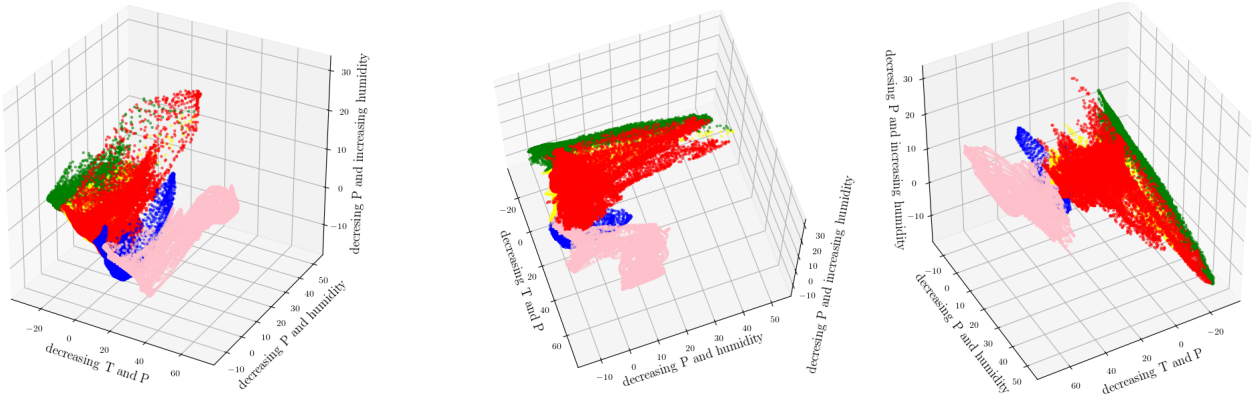

Figure 8: 3D PCA

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.

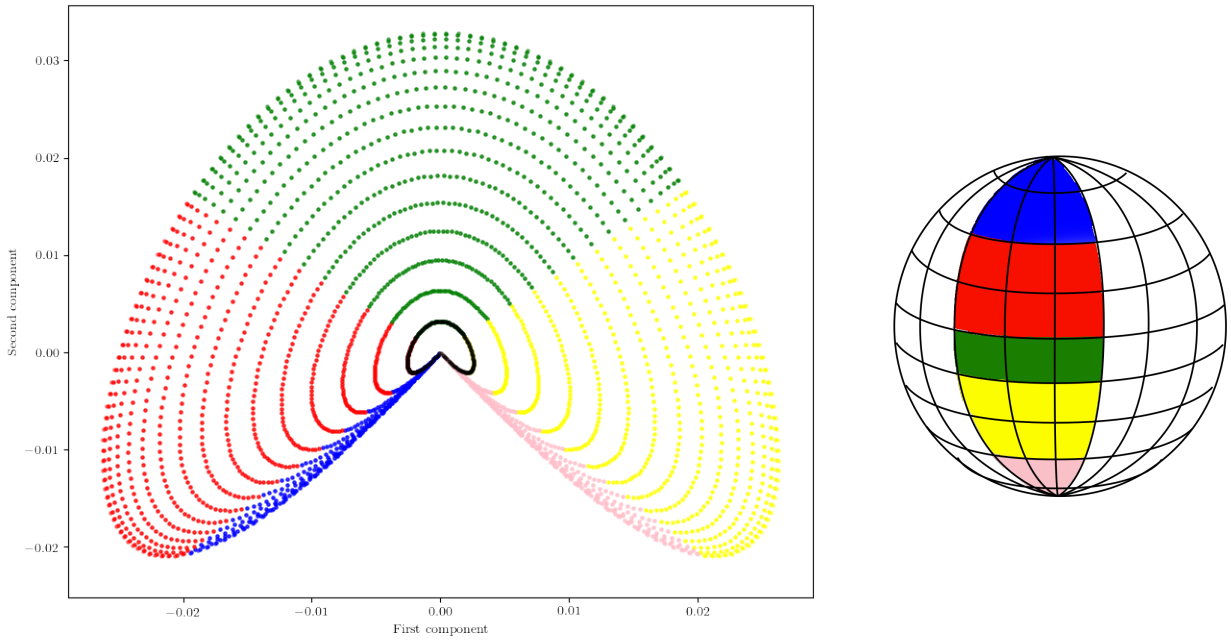

Figure 9: Kernel PCA with two components and the RBF kernel

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.

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.

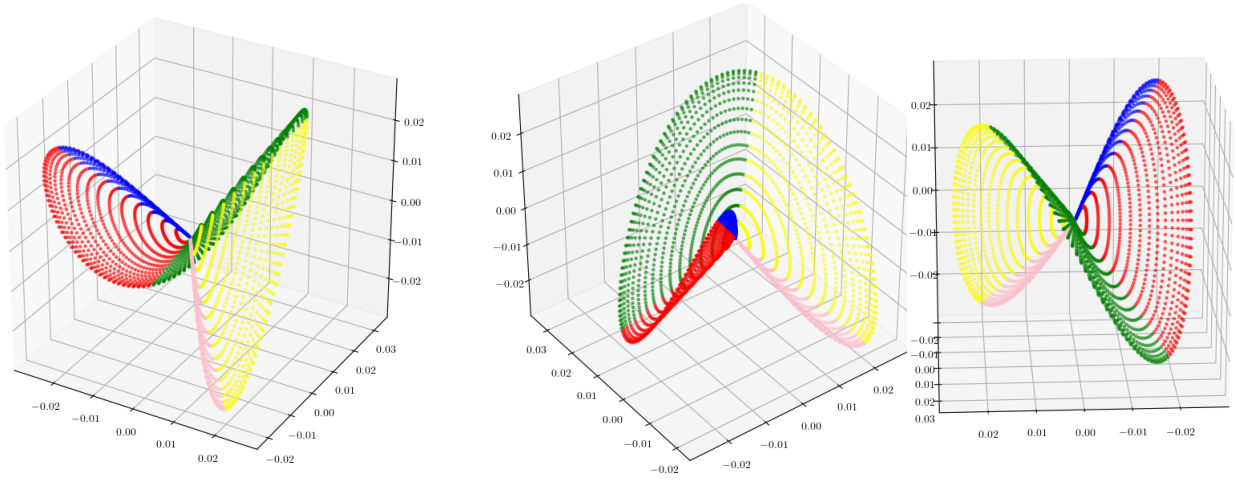

Figure 10: 3D Kernel PCA

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.

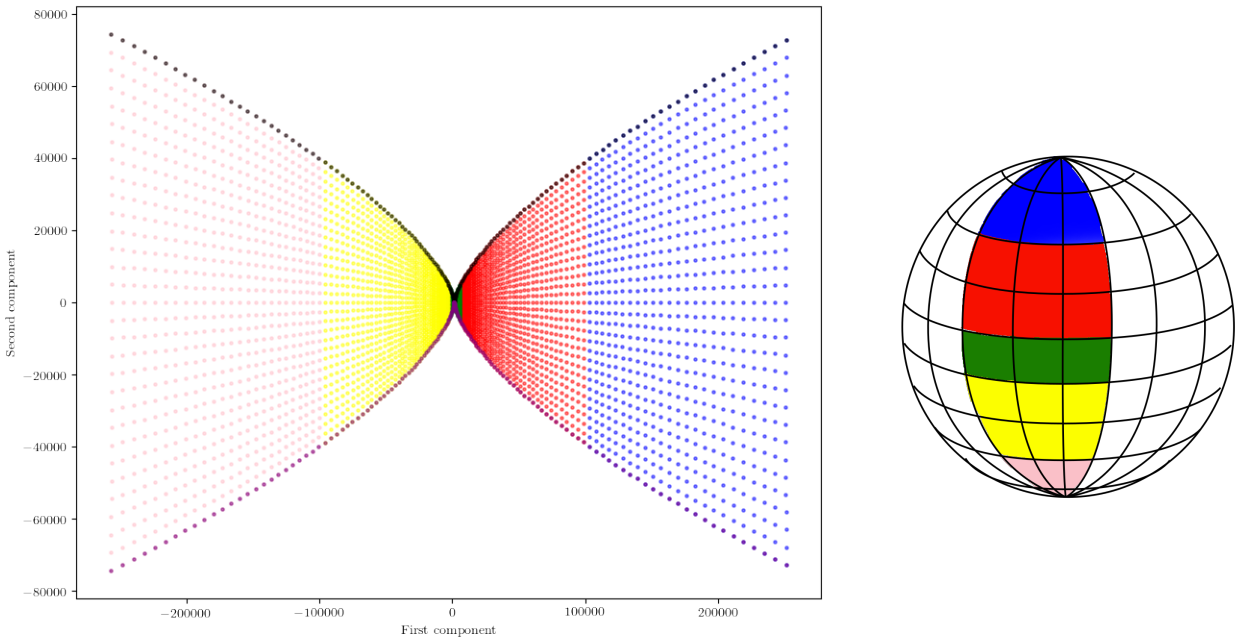

Figure 11: Kernel PCA with two components and polynomial kernel

However, a flaw 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, however, other variables especially humidity vary a lot as this area gathers tropical and desert areas. Moreover, we saw in table **??** that the equator area contained 40% of the earth's surface: it makes sense that it should be mapped onto a larger area than the other classes.

4.1.4 Multidimensional Scaling - MDS

As MDS was extremely slow and seemed to use a lot of memory, we could not run the program for more than 600 datapoints (i.e. 600 rows).Therefore, we used a new subset of longitudes between 0° and +5°, reducing the dataset size to 546 datapoints. Figure **??** shows the results of MDS. As we can see, observations 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 climate diversity 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. The results are shown on figures **??** and **??**. 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.

It is worth mentioning that in MDS, South temperate observations are overlapping with observations from North pole and North temperate. This is not what we expected. It seems like MDS is not very adapted to represent the dataset. This is why we decided to try with isomap. Indeed, as isomap uses the geodesic distance, much adapted to the round shape of earth, we are expecting more accurate results.

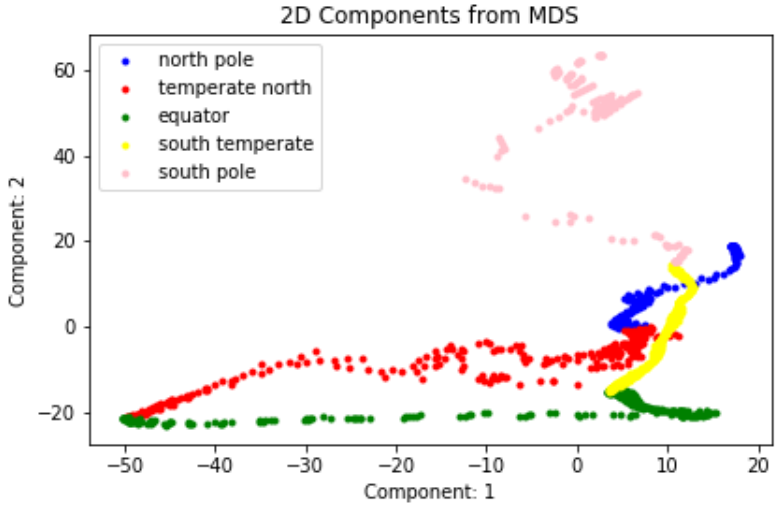

Figure 12: Results of MDS in 2D

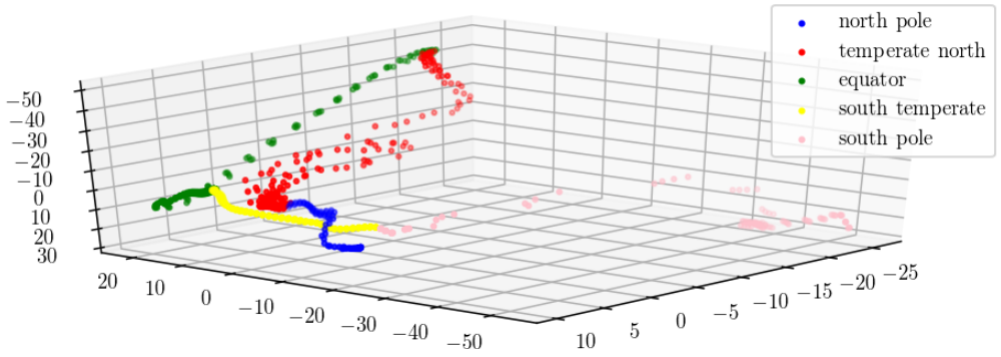

Figure 13: Results of MDS in 3D

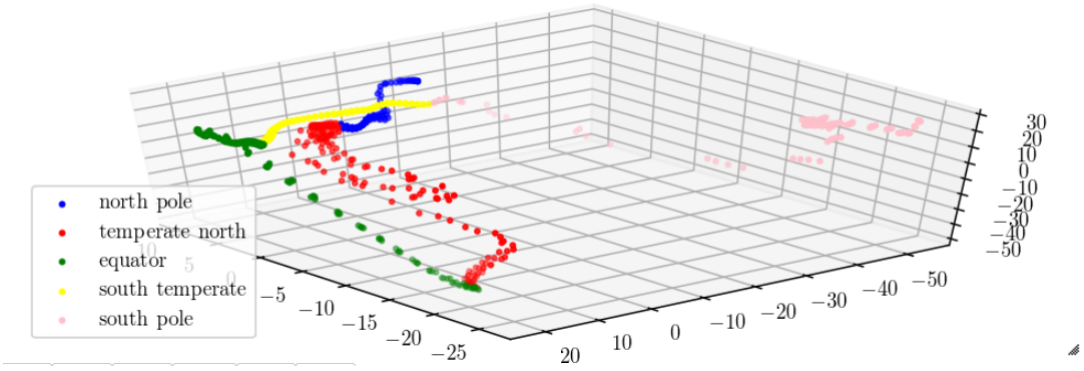

Figure 14: Results of MDS in 3D

To check how accurate MDS is, we used 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:

$$\sqrt{\frac{\sum_{ij}^{n} d_{ij}^{(X)} - d_{ij}^{(Y)}}{\sum d_{ij}^{(X)}}}$$

with $d_{ij}^{(X)} = \|x_i - x_j\|^2$ the Euclidean distance between pairwise i and j in the original $d \times d$ matrix $D^{(X)}$ and $d_{ij}^{(Y)} = \|y_i - y_j\|^2$ the Euclidean distance between pairwise i and j in the computed $r \times r$ matrix $D^{(Y)}$.

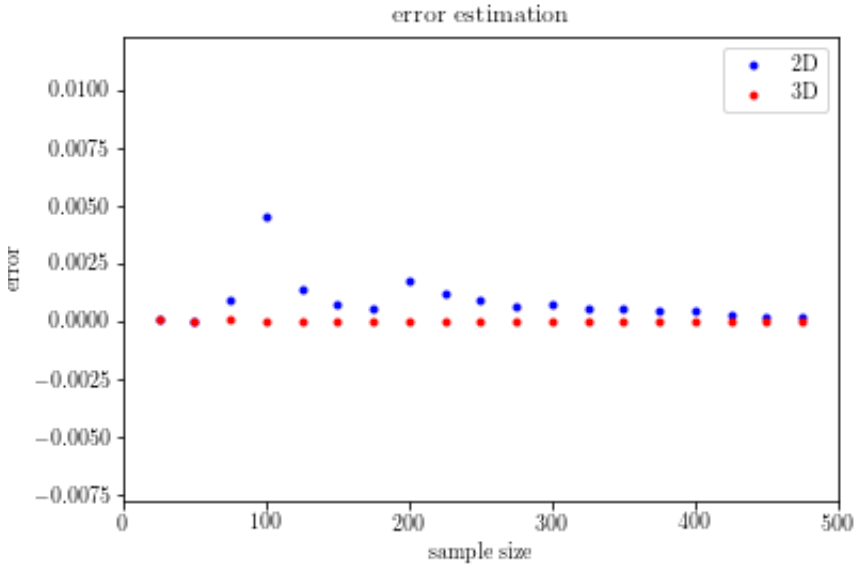

Figure 15: reconstruction error as a function of the sample size for MDS

As expected, stress is higher in 2D than 3D, because we reduce more dimension (or we suppress a degree of freedom and put more constraint). But this difference gets negligible when sample size increases.

4.1.5 Isomap

As we have mentioned before, Isomap does not compute the Euclidean distance but a geodesic distance. For this, we consider that each point in our dataset has a finite number x of 'nearest neighbors', that is to say the x nearest points computed with the Euclidean distance. To go from point A to point C in our dataset, one can only 'jump' from one of the nearest neighbor of point A (let's call him B), then one of the nearest neighbor of point B ... Until we reach point C. So the number of nearest neighbor is a key parameter that can completely reshape our final results. For example, if we set a number of nearest neighbor equal to our data set size n, the geodesic distance will be ... the Euclidean norm, and MDS and Isomap would yield the same results. Therefore, we present in figure **??** the result of isomap for a number of neighbors between 2 to 8.

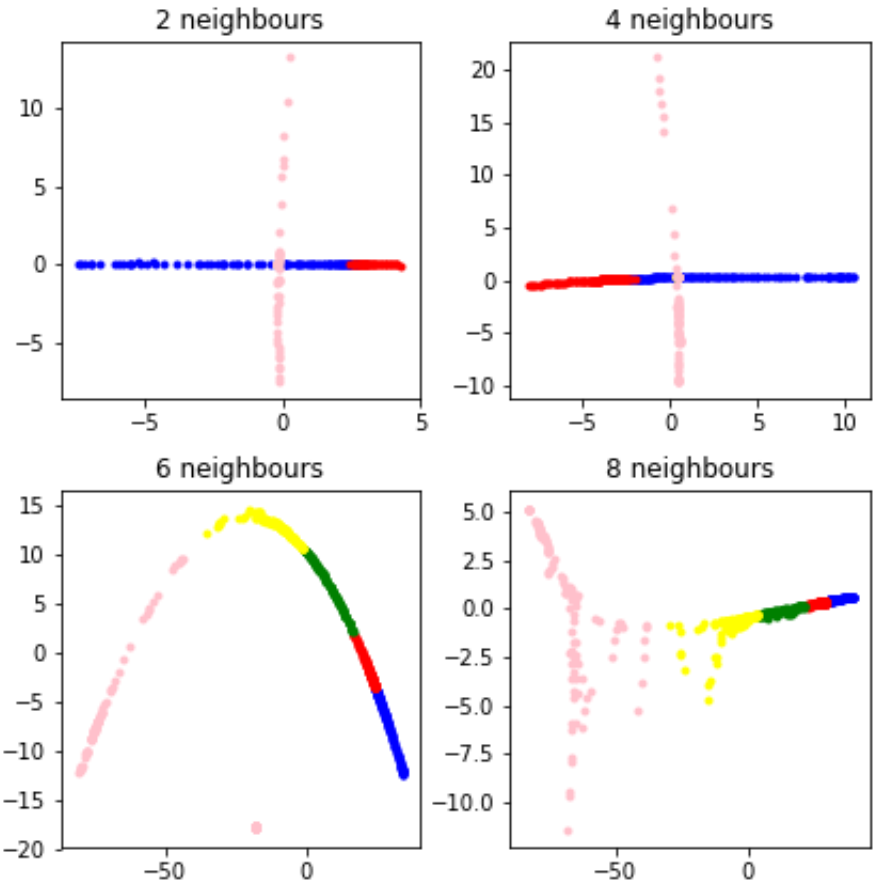

Figure 16: results of isomap depending on the number of neighbors considered from 2 to 8

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-Learn. For the moment, we only need to know that it is an indicator of how much isomap mapping is accurate and respect original distance, it kind of computes difference between the final distance of two items in the isomap mapping and the true distance between them. We computed this reconstruction error with respect to the number of neighbours considered. The results are presented in figure **??**.

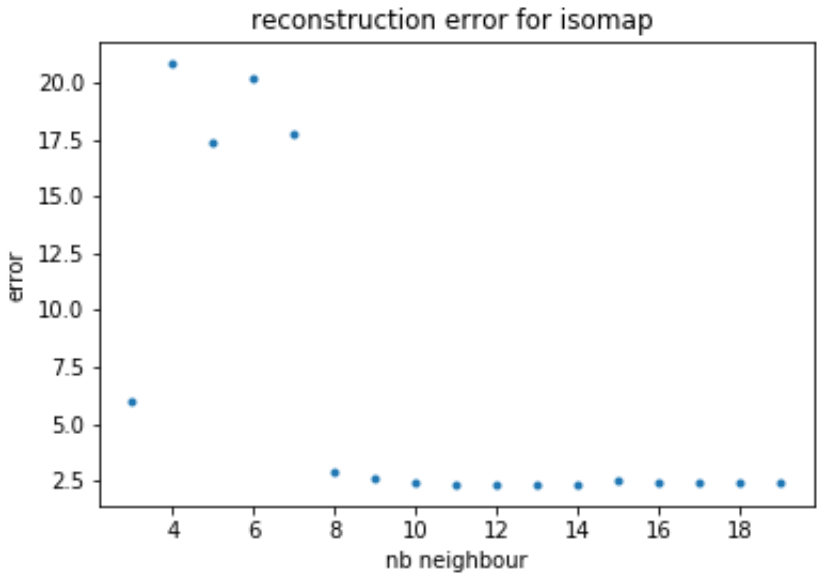

Figure 17: reconstruction error as a function of number of neighbours considered

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. Indeed, representation for 6 and 8 neighbors seemed to be the most accurate, because 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. This is what we expected: geodesic distance is more adapted to the earth due to the round shape of earth.** It is interesting to see that pink points(i.e. south pole) are much more spread in isomap than MDS.

We also checked how error reconstruction evolves with reduced dimension r (=2 or 3). Results are shown on figure **??**. 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.

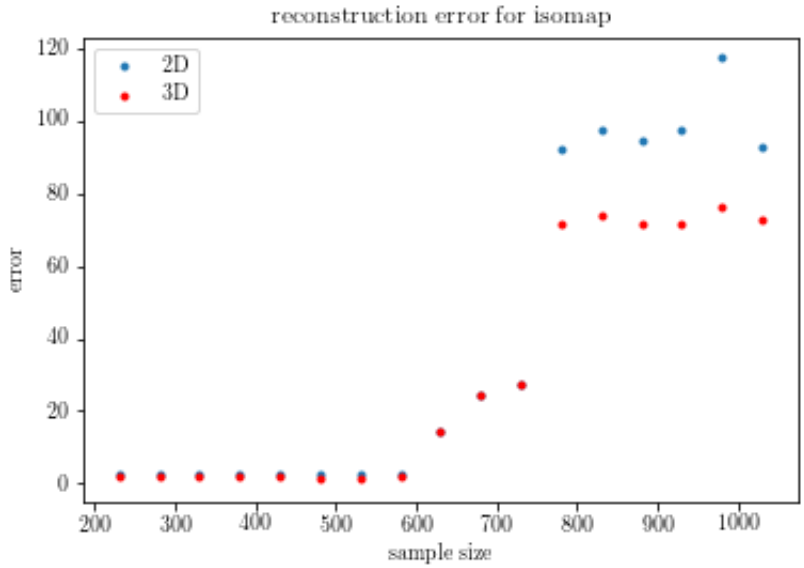

Figure 18: reconstruction error as a function of the sample size for ISOMAP

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

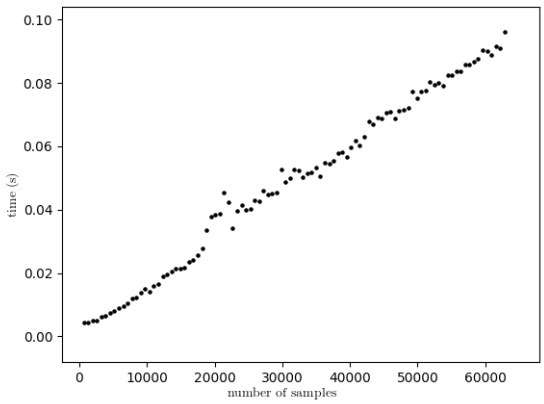

Figure 19: running time for PCA

Kernel PCA was slower, we did not compute its running time for all the dataset, only until

12000 rows. It yields the following graph.

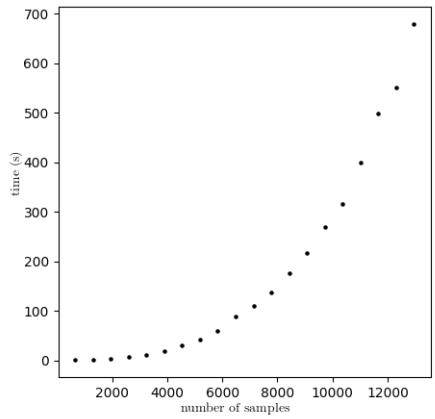

Figure 20: running time for Kernel PCA

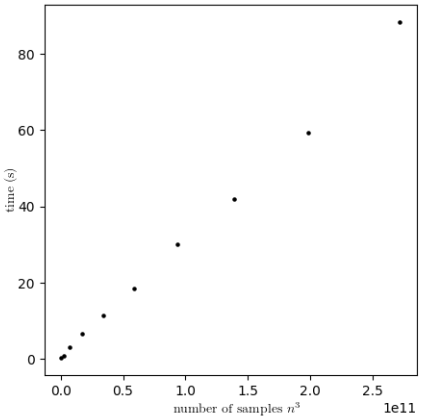

Figure 21: Kernel PCA running time is a $O(n^3)$

In figure **??**, 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 sample size. 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 Scikit-Learn:

> "'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 (=5 here), the following graph presents the complexity of every method.

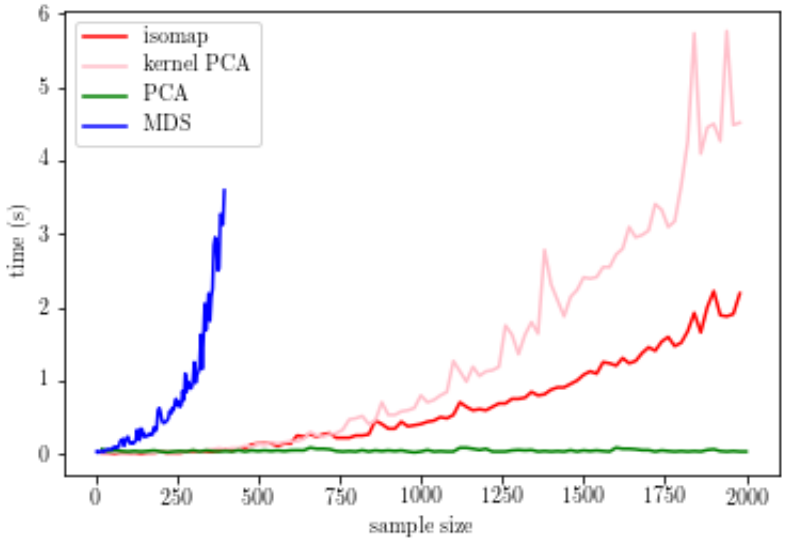

Figure 22: computing time comparison

We also checked how computing time of isomap varies with the number of neighbor taken into account. Figure **??** shows that running time for isomap increases with the number of neighbors. So running time of isomap may be limited by the number of neighbors.

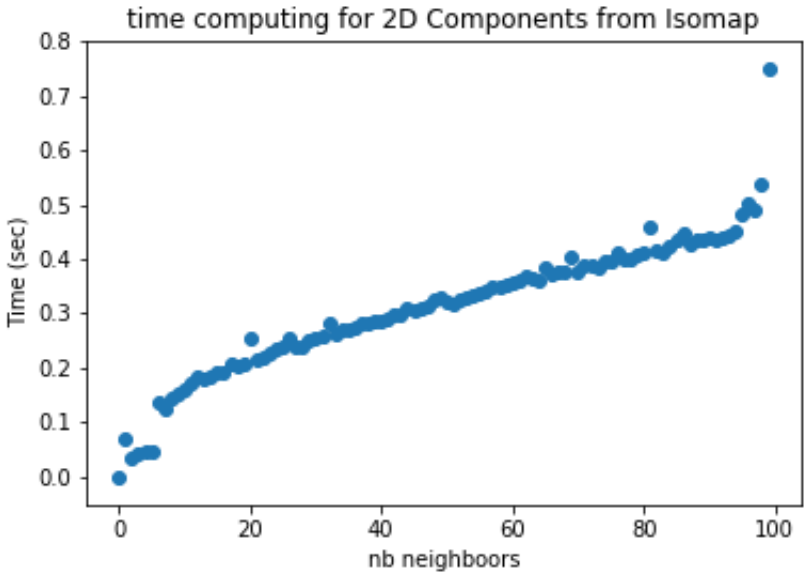

Figure 23: computing time for isomap as a function of the number of neighbors considered

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.

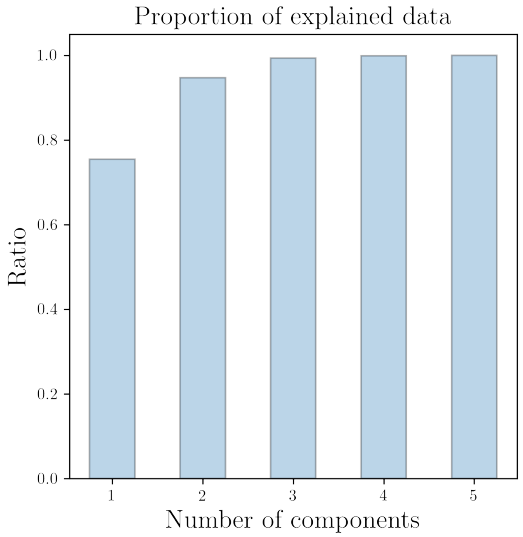

Figure 24: PCA with two components

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.

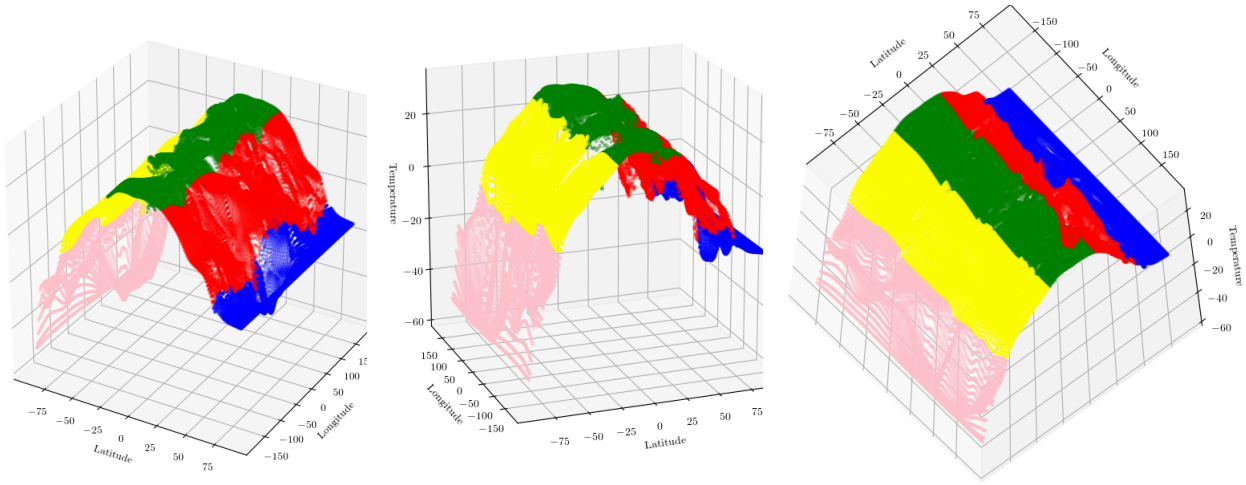

Figure 25: Temperature as a function of latitude and longitude in X

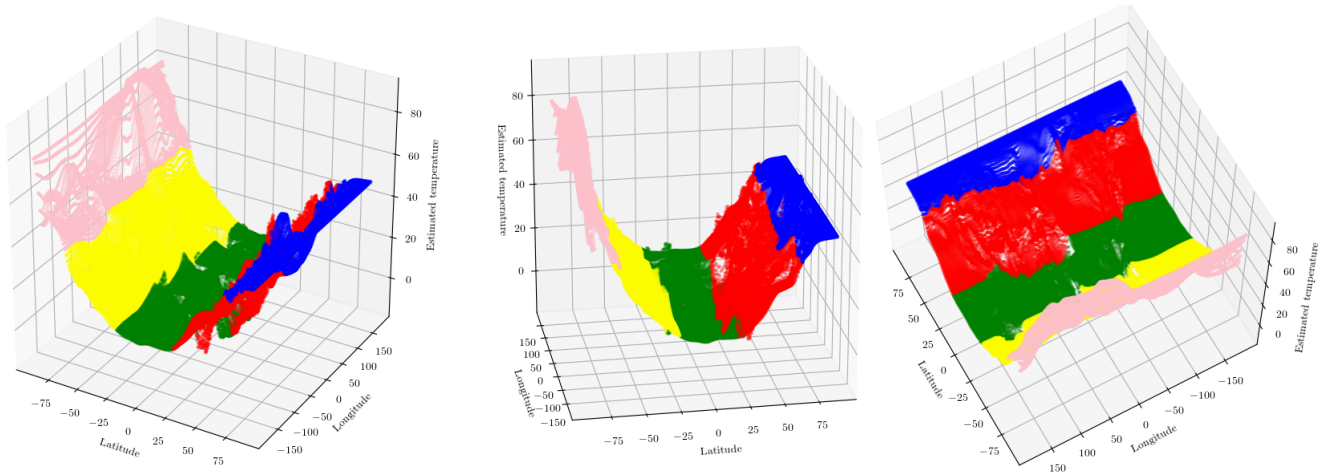

Figure 26: Temperature as a function of latitude and longitude in $\hat{X}$

5 Conclusion

In this project, we have studied four different dimension reduction methods with a climate dataset.

Isomap appears to us as the most adapted method to represent our dataset. That is due to the fact that geodesic distance is much more adapted to the round shape of the earth. Plus, its running time is very satisfying as it is the second fastest method behind PCA. PCA has the

advantage to be fast but fails to represent the dataset. There is a lot of overlapping with this method. MDS and Kernel PCA yields similar results that do not respect the geometry of the earth. Plus, they are slow.

We have seen that those dimension reduction were an efficient way to establish a classification of geographical zones of the earth based on meteorological characteristics. A next possible step would be for example to be able to identify from which zone on earth random climate datas were collected.

6 Bibliography

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

John P. CUNNINGHAM, Zoubin GHAHRAMANI, Linear Dimensionality Reduction: Survey, Insights, and Generalizations, , *Journal of Machine Learning Research 16* (2015) 2859-2900

Alon ORLITSKY, SAJAMA, Supervised dimensionality reduction using mixture models, Proceedings of the 22nd international conference on Machine learning, p.768-775, August 07-11, 2005, Bonn, Germany

Bernhard SCHOLKOPF, Alexander SMOLA,Klaus Robert MULLER, Kernel Principal Component Analysis