Geometric Methods in Machine Learning
Report

Alexis Saïr

We report here some interesting figures and remarks. Further information can be found in the notebook and in the code files. We first describe our code and then report the results obtained on synthetic data (to check the functionality of our code) and then on the Weekly Sales dataset.

1 Code Presentation

As part of the project of studying Time Series using Dynamic Time Warping, a certain amount of code had to be written. We wrote all the code used to lead this anlysis at the exception of the time series averaging. First, we had to implement the computation of the Dynamic Time Warping, then Metric MultiDimensionnal Scaling (MMDS) and then Isomap, which relies on MMDS, building adjacency matrices and Floyd-Warshall algorithm to compute shortest paths. We also wrote a simple clustering algorithm : the KMeans algorithm. The whole code and a notebook with figures can be found on my GitHub here and is written in Python.

We are going to shortly present the code we wrote for a better understanding.

- *dynamictimewarping.py* implements the class DTW which can be fitted on the data of shape (N, L) where L is the length of the time series and N the number of time series and returns an array D where $D[i, j]$ is the Dynamic Time Warping distance between the time serie i and the time serie j. The distance we use to compute it is the l^2 norm but any dissimilarity measure can be used. Our implementation is a regular implementation whose complexity is $\mathcal{O}(L^2N^2)$.

- *mmds.py* implements the Metric Multi-Dimensionnal Scaling. An instance of MMDS takes as parameters the desired number of components and a number of iterations for the SMACOF algorithm. It must be fitted on a dissimilarity matrix D.

- *floydwarshall.py* is the implementation of Floyd-Warshall algorithm. For a graph of N vertices, its complexity is $\mathcal{O}(N^3)$

- *isomap.py* contains our implementation of Isomap algorithm. We implemented two methods to find the nearest neighbors : the *knn* method which connects a vertice to its k nearest neighbors based on a dissimilarity matrix D and the *epsilon* method which connects a vertice to the vertices located at a distance inferior to εm where m is the mean dissimilarity. In theory, after applying Floyd-Warshall, the shortest path between data points not belonging to the same connected component would be infinity, but it would be impossible to run our MMDS algorithm on it and to represent our data. As a consequence we decided to set the distance between such data points to fM where M is the maximum shortest path observed and f a certain factor (e.g 3).

- *kmean.py* is our implementation of the K-Means algorithm. It runs until convergence (the centroids don't change anymore) or when a certain number of iterations is reached. We used Forgy initialization (random centroids among the data points as initialization).

To compute time series averaging, we used F.Petitjean's code here based on [1] , [3] and [4]. This averaging is very interesting because it is consistent with DTW. If we took the arithmetic mean as an averaging of two time series that are similar we could loose a lot of information if for instance one is a bit translated with the other. The resulting shape could be a lot *attenuated* : maximas of one time series could correspond to minimas of the other for instance whereas their global shapes are very lookalike.

2 Results

2.1 Synthetic Data

To check that our code was indeed efficient and functional, we evaluated it on synthetic data. We used the classic Two moons dataset as synthetic data. It is shown on Figure 1. Clearly, performing a k-means algorithm on that dataset isn't a great idea. It is shown on Figure 2. Actually, Euclidean Metrics are not suited to this dataset : on the contrary, graph-based metrics are since the clusters are transmited locally. As a consequence, we decide to perform our implementation of Isomap on that dataset. The results in 2D are displayed on Figure 3. We can check that during the training, the stress is indeed decreasing, as shown in Figure 4. Applying a K-means algorithm now would get without any problem the right clusters. We can now apply our code to the dataset of interest, namely the Sales Transactions Weekly Dataset.

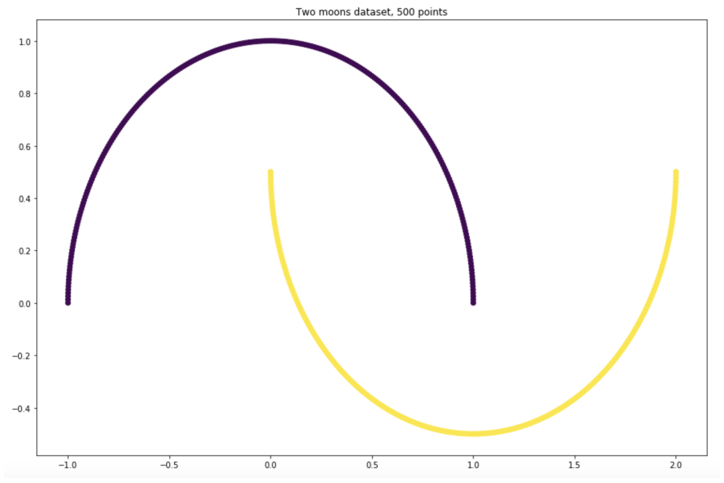


Figure 1: Representation of the Two Moons dataset. We notice that it is not linearly separable.

2.2 Sales Transactions Weekly Dataset

The dataset is composed of around 800 time series of 52 points. It is quite big for computing the Dynamic Time Warping distances matrix but it can be done in approximately less than 10 minutes. For information, we only used the normalized values.
Dynamic Time Warping is a well known mean to represent distance between time series and can achieve state-of-the-art performance in clustering when used as a preprocessing of a simple K-Nearest Neighbors. We use it here to have a representation of the dissimilarities between the 800 time series. Two time series with a small DTW distance are represented Figure 5.

It is quite hard to have a representation in small dimension (e.g. 2) that preserves the original distances present in the dissimilarity matrix, because these pairwise distances in big dimension (here 52) cannot be entirely represented in 2D. We can see it on Figure 6. Figure 6 is a bit disappointing : it would have been really great to find clear clusters on that representation which is clearly not the case here. We can see some small group of less than 10 points that represent very lookalike time series but that's it. That representation also underlines the trade-off we face when dealing with Isomap :

- when building the graph on which Floyd-Warshall will compute shortest paths, if we set k (or ε) too low, there won't be any path between most of the points
- if we set them too high, all the points will be connected and we won't benefit from the graph based aspect of the method.

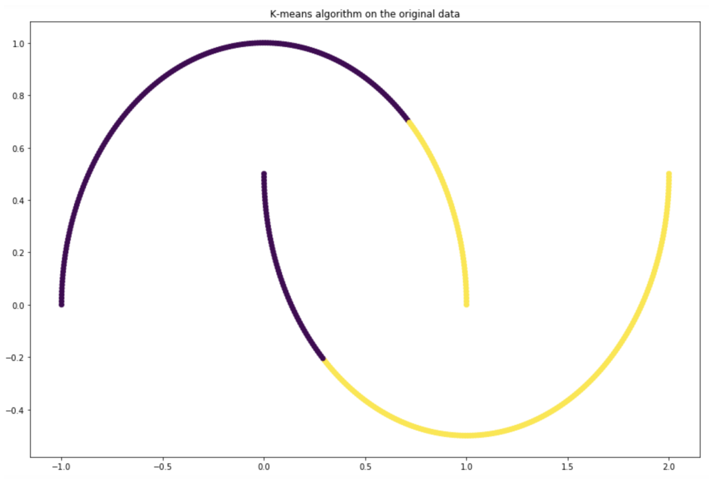


Figure 2: Performing k-means on the Two Moons dataset.

Actually, the main problem we faced is dealing with several connected components : indeed, when the shortest path between two data points returned by the Floyd-Warshall algorithm is infinity (i.e. the two data points are not connected), what distance should be given to the Metric Multi-Dimensionnal Scaling ? We decided to set relatively big distances in that case to model *infinite* distances. However, what we can observe is that the stress, (representing the difference between the real dissimilarities and the dissimilarities in lower dimension) decreases when the dimension increases.

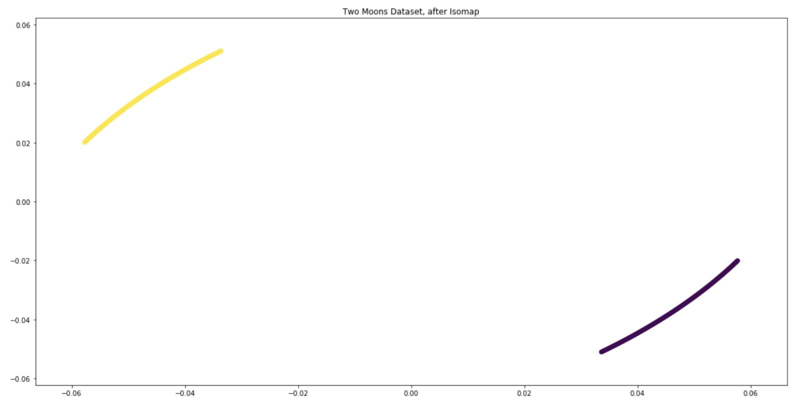


Figure 3: Representation of Isomap in 2D on the Two Moons dataset (iterations = 500, 5 nearest neighbors considered).

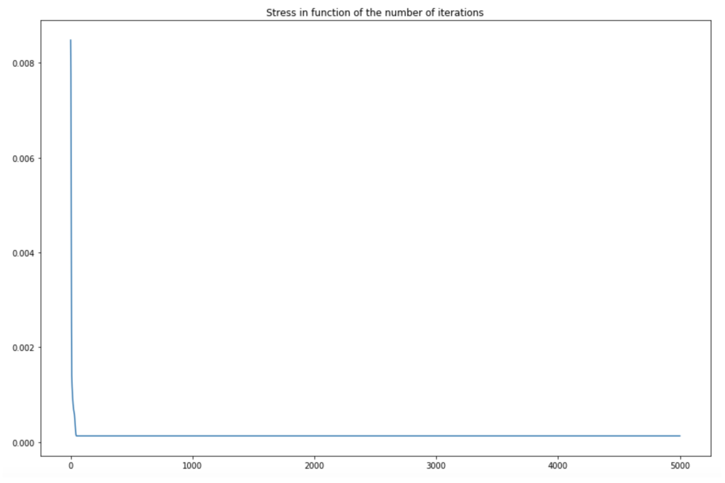
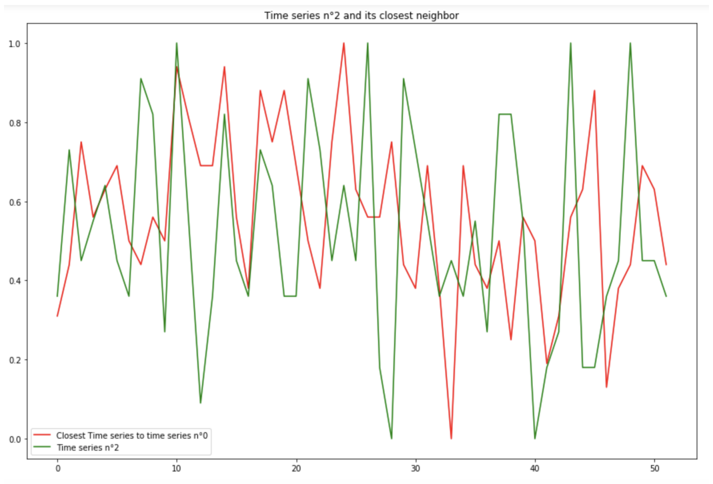


Figure 4: Evolution of the stress.



```
Distance between 3 and its closest neighbor 311 : 0.8337000000000001
```

Figure 5: Two time series whose DTW distance is small.

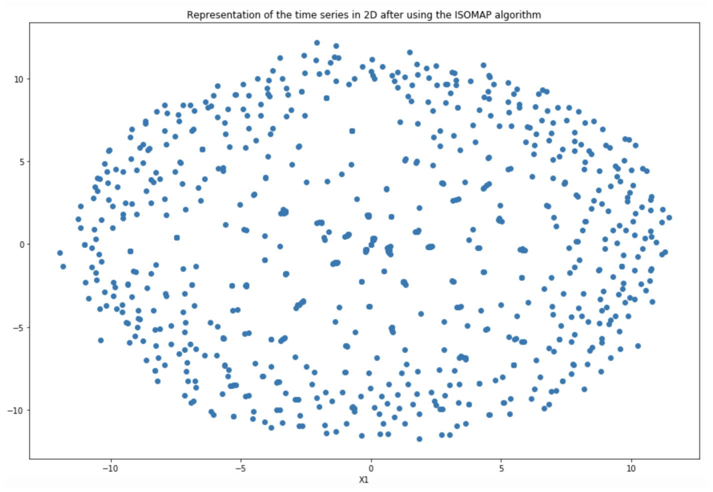


Figure 6: 2D representation of the time series : the number of components is probably too low.

3 Conclusion

In conclusion, when dealing with complex structures in high dimension, using Euclidean distances as a measure of dissimilarity is almost always a bad idea. Here, to compute dissimilarities we decided to use Dynamic Time warping distances because we were dealing with time series. If we had been dealing with images for instance, a good measure of similarity can be the cosine similarity given by the embeddings provided by a Neural Network for instance : it is clear that the pixel wise euclidean distance wouldn't make sense.
When clustering the data, graph based methods such as Isomap can be very efficient : we saw that for the Two Moons Dataset for instance, using Isomap in 2D would allow us to linearly separate the clusters.
We also saw that DBA is a very efficient solution to have the global shape of similar (w.r.t DTW) time series.
Finally, I would like to open on a very interesting paper, puplished in 2018 by Kamnitsas et al. [2] which introduces a new cost function, differentiable, for neural networks that induces a compact representations of clusters in the latent space. Like Isomap, this approach is also a graph based method with a stochastic point of view where dissimilarities are used to represent probabilities to move from a data point to another. This approach is actually particularly useful in Semi-Supervised Learning.

References

[1] Germain Forestier, François Petitjean, Hoang Anh Dau, Geoffrey I Webb, and Eamonn Keogh. Generating synthetic time series to augment sparse datasets. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 865–870. IEEE, 2017.

[2] Konstantinos Kamnitsas, Daniel C. Castro, Loïc Le Folgoc, Ian Walker, Ryutaro Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya V. Nori. Semi-supervised learning via compact latent space clustering. *CoRR*, abs/1806.02679, 2018.

[3] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 470–479. IEEE, 2014.

[4] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.