Graph-based normalization for non-linear data analysis (I)

Catherine Aaron
SAMOS-MATISSE
Paris France
catherine_aaron@hotmail.com

Abstract - *In this paper we construct a graph-based normalization algorithm for non-linear data analysis. The principle of this algorithm is to get, in average, spherical neighborhood with unit radius. In a rst paragraph we show why this algorithm can be useful as a preliminary for some neural algorithms as those that need to compute geodesic distance. Then we present the algorithm, its stochastic version and some graphical results. Finally, we observe the e ects of the algorithm on the reconstruction of geodesic distance by running Dijksrta's algorithm [1] and on the performance of Kohonen maps.*

Key words - normalization, geodesic distance, graph, SOM

1 Introduction

For non-linear structures, geodesic (or curvilinear) distance is much more relevant than euclidian one.

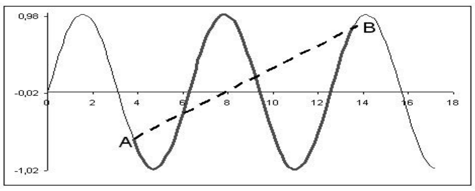

Figure 1: curvilinear (plain) and Euclidian (dashed) distance for two points on a sinusoïde

Data analysis methods, based on geodesic distance have been built recently (as for instance ISOMAP [2] or curvilinear distance analysis [3]) and seem to be helpful for non-linear data analysis.
Normalization impact on geodesic distance computation has been discussed only in [4] and we will present this work again because we need it for the second part [5]. We will see in the second part of this paper that the normalization algorithm is also helpful for SOM algorithms.
For theoretical and in nite sets of points there is no normalization problem for geodesic distance calculation (see gure 2).
For nite and discrete data a normalization problem appears. Calculation of geodesic distance rst needs the computation of a neighborhood graphs such as MST (Minimal Spanning Tree) or K nearest neighbors which strongly depend on axis scaling (as shown on an example in gure 3). Then, to achieve the calculation Dijkstra's algorithm is computed on the graph.

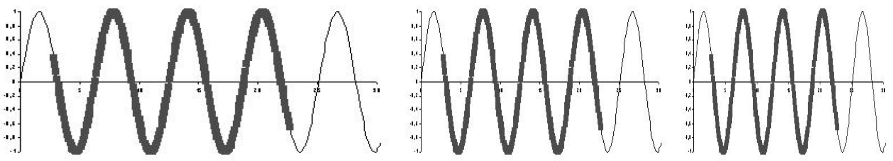

Figure 2: curvilinear distance on a spiral for different scaling

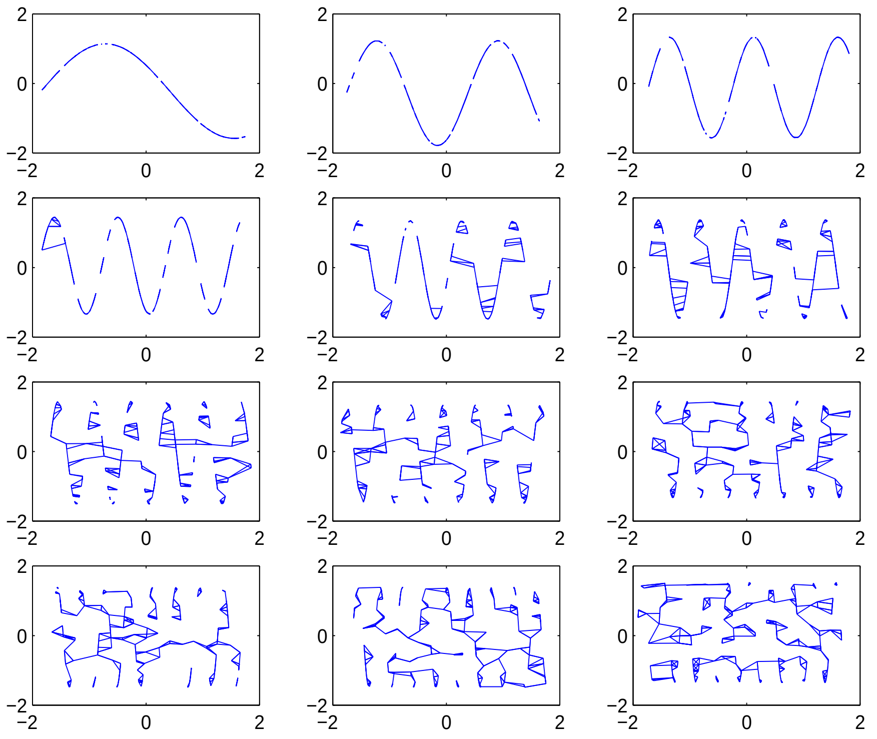

Figure 3: 3− nearest neighbors graph for classically normalized data for $Y = sin(\omega X)$, $X \in \{5, 10, ...55, 60\}$

2 Graph-Based Normalisation

2.1 Principle

Let us rst observe that when sets are non-linear the use of "classical" normalization that studies the global dispersion around a central indicator (usually the barycenter) won't be useful. That may come from to the fact that the barycenter may not be a "good" central indicator for non-linear structure, and, more "philosophically" from the fact that for non-linear sets local indicators may be better than global ones.

To illustrate the lack of interest of "classical" normalization (division by standard deviation), for each example the departure of the algorithm will be done on classically normalized data and we will see that computed graphs are rarely good.

To build an algorithm based on local study, we had the idea to make, in average spherical neighborhoods with unit radius.

2.2 Proposed Algorithm

In this section we denote by X the set of points (N points in $\Re^d$). Notations are
X_i the i^{th} row vector of X ($\in \Re^d$) corresponding to the $i^{the} individual$.
X^j the j^{th} column vector of X ($\in \Re^N$) corresponding to the $j^{the} variable$.
We chose a graph structure (MST or k nearest neighbors) and characterize it by its matrix G^X ($G^X_{i,j} = 1$ if and only if $[X_i, X_j]$ is an edge of the graph and 0 otherwise). The graph de nition leads us to nd neighborhoods (the neighborhood of a point X_i is de ned by all the points X_j such that $G^X(X_i, X_j) = 1$). To perform the idea presented previously of making average neighborhoods spherical with unit ray : we apply rotation and re-scaling on neighborhood to be "better". After these transformations distance between points and same-structured graph changes so we iterate the algorithm.

2.2.1 Exhaustive version

We start with the un-normalized initial data set (in following examples the classically normalized sets to observe that they usually are not good) and iterate :

- Computation of all the "neighborhood vectors" as every $\vec{y} = \vec{X_i X_j}$ if $G^X(i,j) = 1$ or $G^X(j,i) = 1$. We get a data set Y of N' vectors with null mean that represents neighborhoods directions.

- Compute PCA analysis on Y that gives a P isometric matrix.

- Apply $X := PX$ that transforms axis such that the new neighborhood vectors have a diagonal covariance matrix (as the P matrix is isometric $G^X = G^{PX}$)

- To make the average radius equal to 1, we de ne the weight w_j of the j^{th} axis as follows.

$$w_j = \frac{1}{N'} \sum_{i \neq j, G(i,j)=1} |Y_i^j|$$

  And, nally, we apply $X^j = X^j / w_j$ to normalize edge's size to 1.

So we obtain a transformed data set which has a neighborhood structure that we expect to be "better" and iterate transformation on this new data.
We now present some results. For each example we rst present initial data and graph (classical normalization), then PCA cumulated inertia, indicator of "rotation" convergence during the algorithm, weight of axis during algorithm, and, nally, graph-based normalized data and graph)
The rst examples are sinusoïdal sets with di erent frequencies.
The second ones are sinusoïdal sets with gaussian perturbation and $\pi/4$ rotation. $X^2 = sin(\omega X^1) + \varepsilon$ with $\varepsilon \rightarrow N(0,1)$ and $X = QX$ with Q the $\pi/4$ rotation matrix.
We observe quite encouraging results even if, for too big frequencies, we can't reconstruct data organisation. This saturation e ect occurs more quickly when data are rotated.

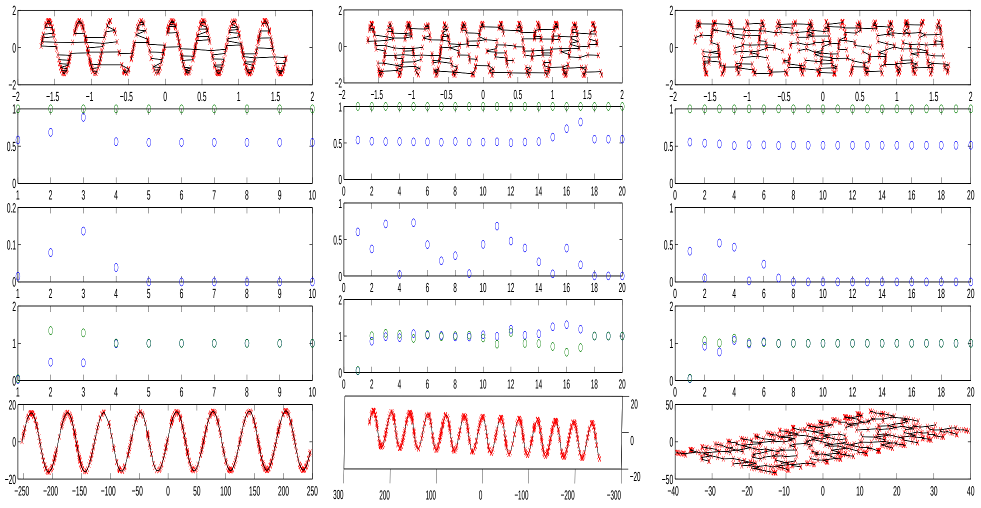
Figure 4: 500 points with $X^1 = unifrnd(0,1)$ and $X^2 = sin(\omega X^1)$ with $\omega \in \{50, 80, 100\}$

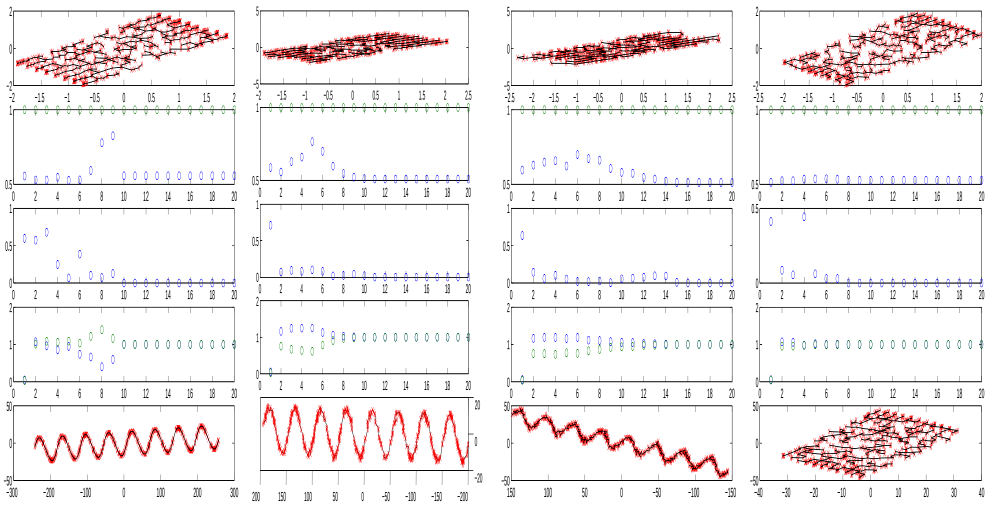
Figure 5: Rotated examples : (1) $\omega = 50, \sigma = 0$,(2) $\omega = 50, \sigma = 0.1$, (3) $\omega = 50, \sigma = 0.2$,(4) $\omega = 70, \sigma = 0$

2.2.2 Stochastic Version for the Algorithm

As the running time may be long because of the computation of di erent graphs at each step of the algorithm, we propose, here, a stochastic version for huge data sets (in practice for more than 1000 points).

First, we choose a graph structure (k nearest neighbors, Minimal spanning tree...) which will be used in the whole algorithm.
Iterate :
While $it <= NbIt$

- (1) randomly draw $N' < N$ points that form the set X'

- (2) compute $G^{X'}$
- (3) compute Y' edge vector and run PCA (obtain P' isometric)
- (4) apply $X := P'X$ and $Y' := P'Y'$
- (2) $\forall j$ compute w'_j on Y'
- (3) $\forall j$ $X^j := X^j / w_j$

As the edges of the partial data-set (X') are bigger than those of the complete one, this algorithm won't converge to a weight of 1 for all directions on the complete data set, but to a graph that has the same weights for all directions.
We present here results for $3 \quad D$ sinusoïde with $N = 1000$, $N' = 500$ and $it = 50$ with MST based normalization and representation of the 8 Nearest Neighbor graph

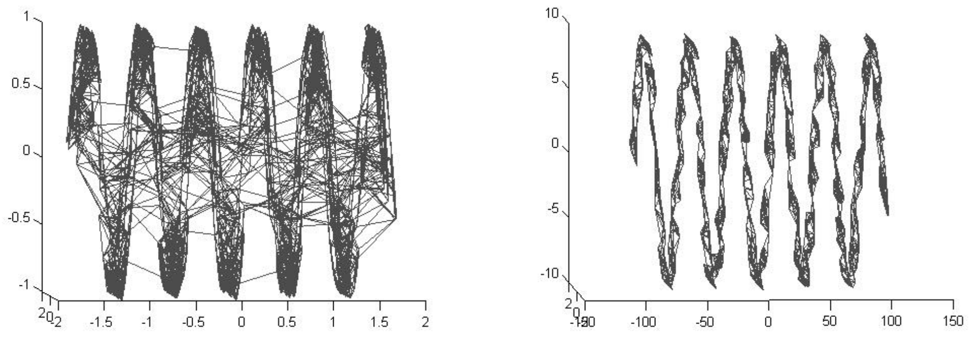

Figure 6: Results for $3 - D$ sinusoïde

3 Impact on the Geodesic Distance

In this section we want to observe the impact of our normalization on the computation of geodesic distance. As the theoretical study is in progress we can only give empirical results based on examples.
Here, we simulated sinusoïdal sets $X^1 \hookrightarrow U[0,1]$ and $X^2 = sin(\omega X^1)$ so that the "true" geodesic distance $d_c(X_i, X_j)$ only depends on $|X_i^1 \quad X_j^1|$.
We randomly draw 100 points.
We used 8 nearest neighbors graph of X for normalization and for geodesic distance computation. The two examples presented in gure 7 can be read by row : rst 8 nearest neighbors graph and scatter plot of $d_c(\hat{Y_i}, Y_j$ v.s. $|X_i^1 \quad X_j^1|$ (for each couple of points) for classical normalization then same gures for graph-based normalization. The two examples use di erent values of ω.
As there are only 100 points (Dijkstra's algorithm is very slow) we can only test small frequencies but results for geodesic computation are quite promising.

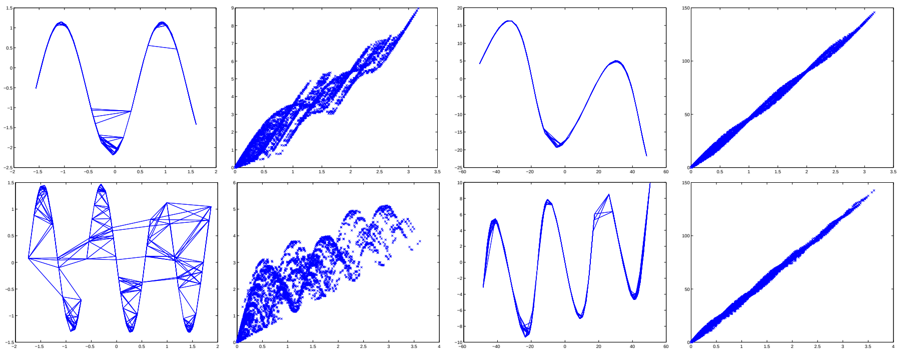

Figure 7: 8−nearest neighbors graph and scatter plot of $d_c(\hat{Y_i}, Y_j)$ v.s. $|X_i^1 - X_j^1|$ for classical normalization and MST-based normalization 1 $\omega = 10$ and 2 $\omega = 20$

4 Impact on Kohonen Maps

It appears that Kohonen maps are very sensitive to normalization. This is due to the fact that, at each iteration a random observation is computed. Then its nearest weight vector (and all its neighbors) are adapted. The problem, here is obviously, which is the "true" nearest weight vector ?

To illustrate this (gure 8) we simulate an initial condition for Kohonen algorithm with initial weight vector randomly drawn from the data set (big points) and observe the Voronoi's cells of two weight vectors if the drawn point is in one of these two cells it attracts one of the two weight vectors away from the set in the rst case, inside the set in the second one.

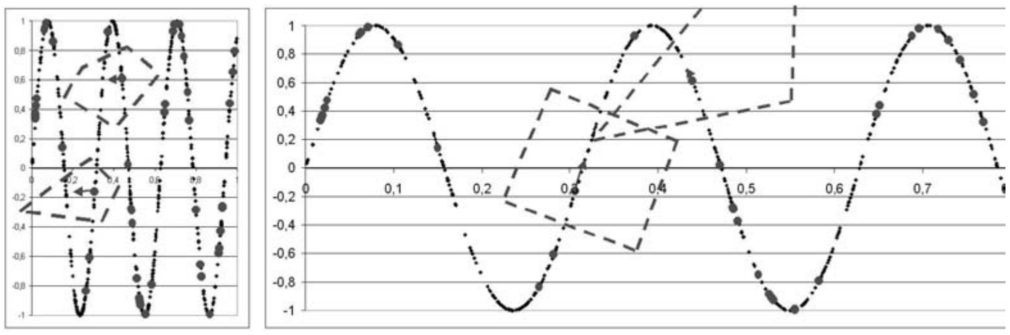

Figure 8: Impact on the normalization on Kohonen algorithm

In fact, we'd observe that Kohonen maps are even much more sensitive to normalization than graphs. This is illustrated on the following gure were we present results of Kohonen maps for initial data and normalized data with a sinusoïde example. Even if MST graphs are almost good in the classical normalization case, Kohonen algorithm fails. After graph-based normalization, results are much better.

The graphic is similar to gure 4 and 5 on which we subplot in dark and plain results of Kohonen strings (5000 iterations) before and after normalization.

To illustrate the performance of graph normalization, we also present results for $3D$ sinusoïdes (only results of SOM algorithm with classical normalization and graph-based one):

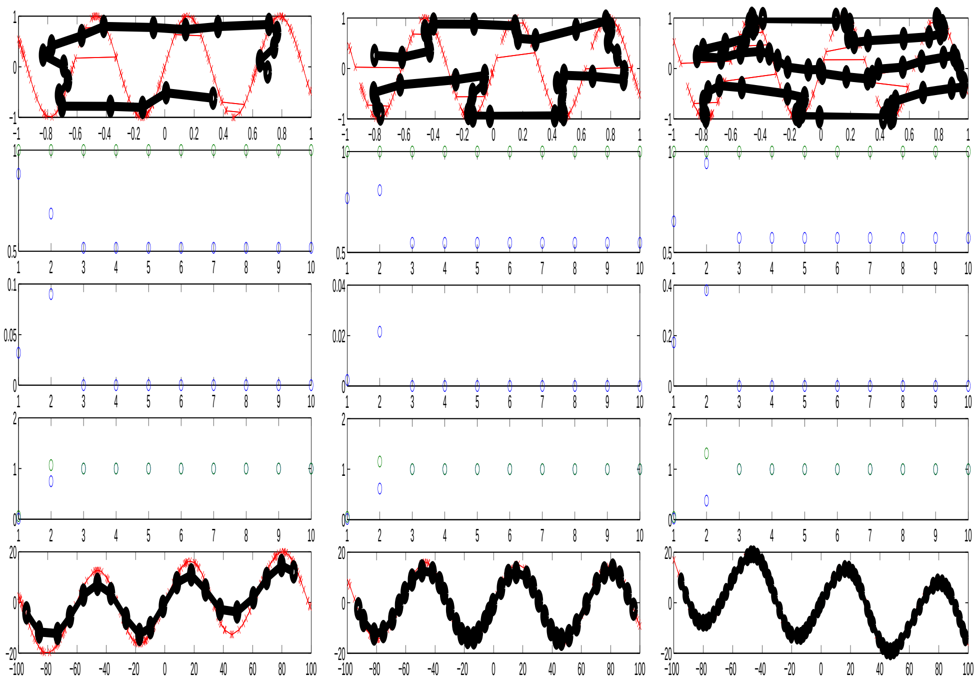

Figure 9: 200 points with $X^1 = unifrnd(-1,1)$ and $X^2 = sin(10X^1)$ results of a Kohonen string after 5000 iterations for 20, 40 and 70 neurons

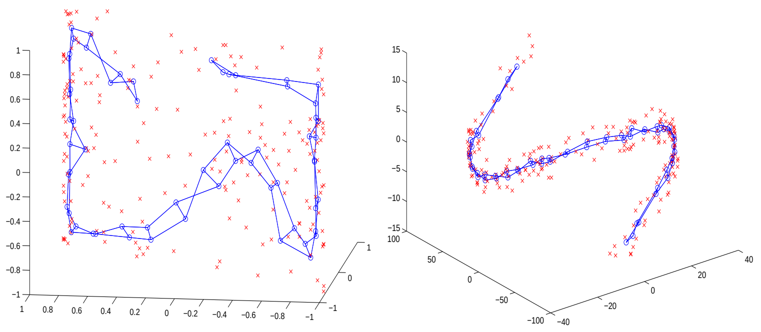

Figure 10: 200 points with $X^1 = unifrnd(-1,1)$ and $X^2 = sin(2X^1)$ and $X^3 = unifrnd(-1,1)$ example of a $(2,50)$ Kohonen maps befor and after normalization

5 Conclusion and Further Work

We shall observe that the proposed algorithm might be useful for preliminary treatment before applying non-linear data analysis methods and that it works very well.

We would now like to know why ? rst works shows that for uniform distribution it is the border e ects that make everything working but calculus are very difficult to do in a general purpose.
We manage to observe on examples that the existence and the unicity of a solution (i.e. a

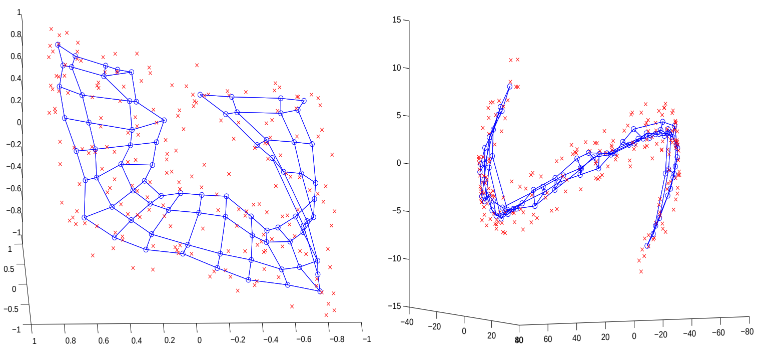

Figure 11: 200 points with $X^1 = unifrnd(-1,1)$ and $X^2 = sin(2X^1)$ and $X^3 = unifrnd(-1,1)$ examples of a $(4,20)$ Kohonen map before and after normalization

scaling and rotation that leads to average neighborhood spherical with unit radius) is not sure when dimension $d \leq 2$. This point lead us to use mostly the stochastic version of algorithm. Another more practical problem of the algorithm is the following one : we have to suppress direction when there weight is null, not to divide by 0, an idea should be to, also, delete them when they don't explain a certain percent of inertia in the PCA step. But this should not be done in the begining of the algorithm when axis may be very compressed, so when and how ?
A rst application of the normalization is presented in [5] were we use geodesic distance to measure the topology preservation of a SOM, to be able to do that we need a preliminary treatment on data that allows to compute "good" geodesic distance and "good" SOM results at the same time.

References

[1] E.W. Dijkstra (1951), A note on two problems in connection with graphs, Mathematik, **vol. 1** p. 269-271.

[2] J.B. Tenenbaum, V. de Silva (2000), A global geometric framework for non-linear dimensionality reduction, *Science*, **vol. 290** p. 2319-2323.

[3] J.A. Lee, A. Lendasse and M. Verleysen (2000), A global geometric framework for non-linear dimensionality reduction, *proceedings of the* 8th *European Symposium on Arti cial Neural Networks*, **vol. 1** p. 13-20.

[4] C. Aaron (2005), Graph-based Normalisation, *proceedings of the* 13th *European Symposium on Arti cial Neural Networks*, **vol. 1**

[5] C. Aaron (2005), Graph-based normalization for non-linear data analysis (II) : application to Kohonen's input structure optimization, *proceedings of the* 5th *Workshop On Self-Organizing Maps*, **vol. 1**