Graph-based Normalisation

Catherine Aaron

University Paris I- SAMOS
90 rue de Tolbiac 75013 Paris - France

Abstract. In this paper we construct a graph-based normalisation algorithm for non-linear data analysis. The principle of this algorithm is get, in average, spherical neighborhood with unit ray. In a first 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 effects of algorithm on reconstruction of geodesic distance by running Dijksrta's algorithm [1].

1 Introduction

New data analysis methods, based on curvilinear or geodesic distance (as for instance ISOMAP [2] or curvilinear distance analysis [3]) seem to be relevant to solve non-linear problems. In fact geodesic (or curvilinear) distance between two points in a set E is the minimal length of a continuous path included in E that links the two points. As illustrated in figure 1 such a distance is relevant for non-linear data-set.

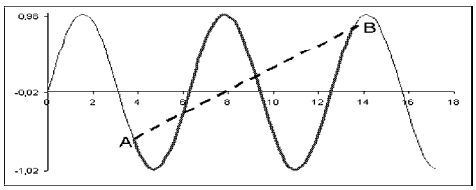

Fig. 1: curvilinear (plain) and Euclidian (dashed) distance for two points on a spiral

For theoretical and infinite sets of points there is no normalisation problem.

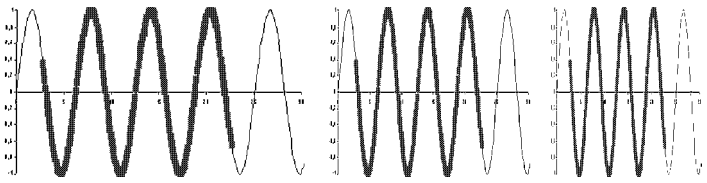

Fig. 2: curvilinear distance on a spiral for different scaling

For finite and discrete data a normalisation problem appears. That is linked to the fact that the computed graph (MST, $K-$nearest neighbors...) used before computing curvilinear distance depends strongly on the scale on the different axes.

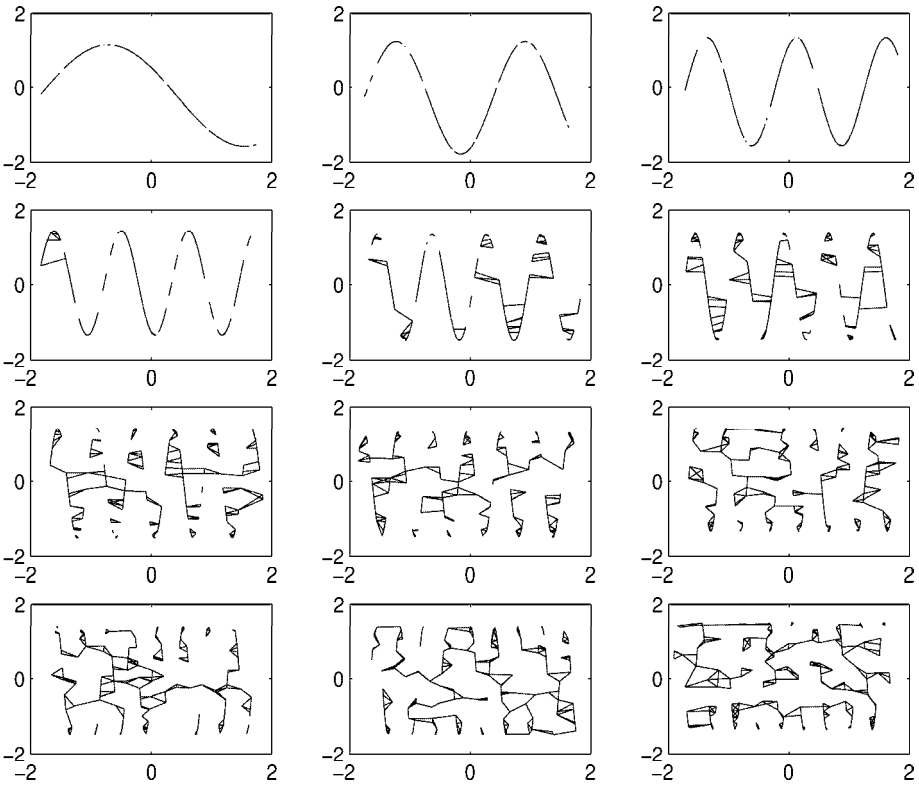

Fig. 3: $3-$ nearest neighbors graph for classically normalized data for $Y = sin(\omega X)$, $X \in \{5, 10, ...55, 60\}$

2 Graph-Based Normalisation

2.1 Principle

Let us first observe that when sets are non-linear the use of "classical" normalisation that studies the global dispersion around a central indicator (usually the barycenter) won't be useful. Non-linear sets study needs local indicators. Our algorithm is based on the principle of making neighborhoods spherical and of ray 1 (in average).

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 ($\in \Re^d$).

X^j the j^{th} column vector of X ($\in \Re^N$).

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 definition leads us to find neighborhoods (the neighborhood of a point X_i is defined by all the points X_j such that $G^X(X_i, X, j) = 1$).

2.2.1 Exhaustive version

To make neighborhoods spherical we compute all the edge vectors $\vec{y} = X_i\vec{X}_j$ 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.

To make the neighborhoods spherical we only apply PCA analysis on Y that gives a P isometric matrix.

$G^X = G^{PX}$ because P is an isometric transformation.

Applying $X := PX$ rotates the edges of G^X so that they are $Y = PY$ with a diagonal covariance matrix.

To make the average ray equal to 1, we define 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 finally we apply $X^j = X^j / w_j$ to normalize edge's size to 1.

This transformation changes everything in the graph organisation so described steps have to be iterated. Observation of convergence (stability) is given by the indication of :

The P isometric transformation (we characterize it by the maximum rotation angle).

The w_j weights we expect to be equal to 1.

The following graphs illustrate results of the algorithm. For each example, the the graph for classically normalized data is first presented, then the PCA cumulated inertia, the maximum of rotation, the weight of axis and, finally, the MST for graph-normalized data.

The first examples are sinusoïdal sets with different frequencies.

The second ones are sinusoïdal sets with gaussian perturbation and $\pi/4$ rotation. $X^2 = sin(\omega X^1) + \sigma\varepsilon$ with $\varepsilon \rightarrow N(0,1)$ and $X = QX$ Q $\pi/4$ rotation matrix.

We observe quite encouraging results even if, for too big frequencies, we can't reconstruct data organisation. This saturation effect occures quicker when data are rotated.

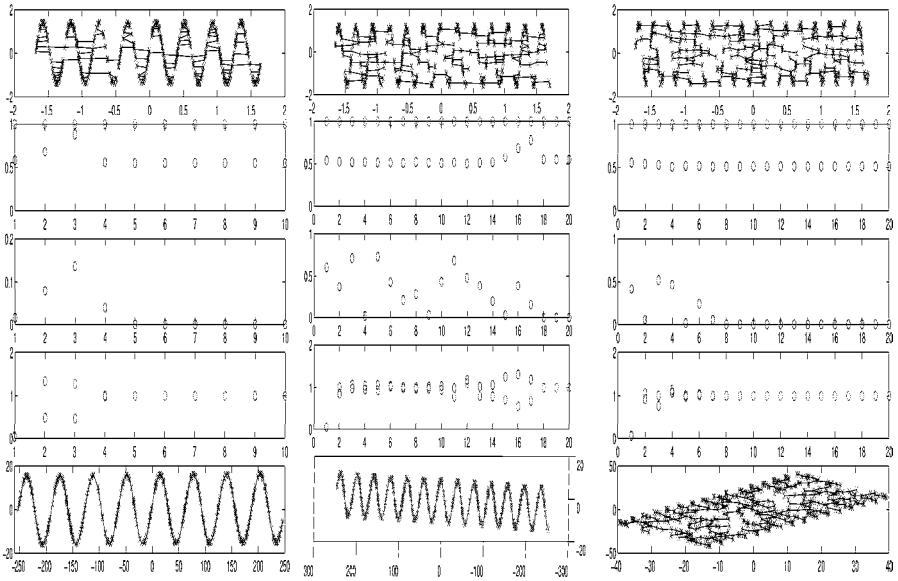

Fig. 4: 500 points with $X^1 = unifrnd(0,1)$ and $X^2 = sin(\omega X^1)$ with $\omega \in \{50, 80, 100\}$

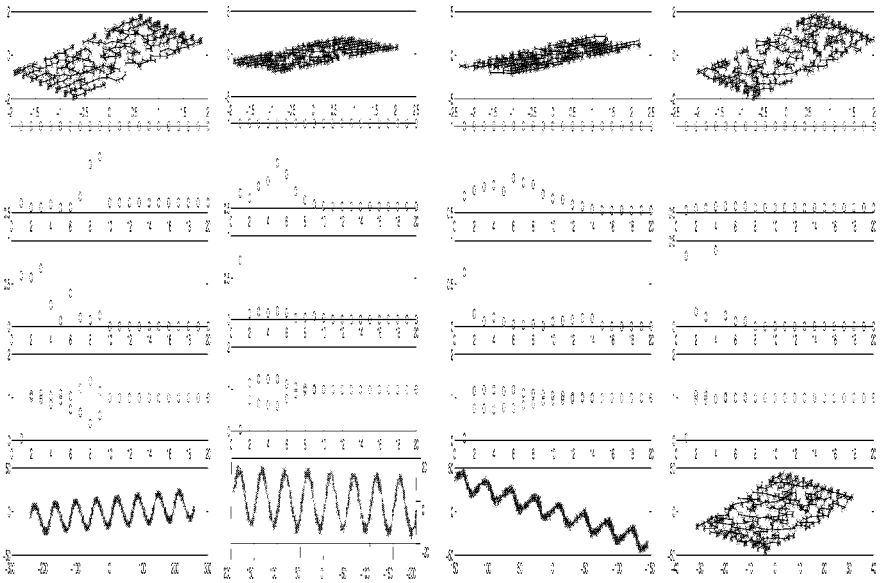

Fig. 5: Rotated examples : (1) $\omega = 50, \sigma = 0$,(2) $\omega = 50, \sigma = 0.1$, (3) $\omega = 50, \sigma = 0.2$,(4) $\omega = 70, \sigma = 0$

2.2.2 Stochastic version for the algorithm

As the running time may be long because of the computation of different graphs at each step of the algorithm we propose here a stochastic version for huge data sets (practically 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) select $N' < N$ 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 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 $3-D$ sinusoïde with $N = 1000$, $N' = 500$ and $it = 50$ with $MST-$based normalisation and representation of the $8-$Nearest Neighbor graph

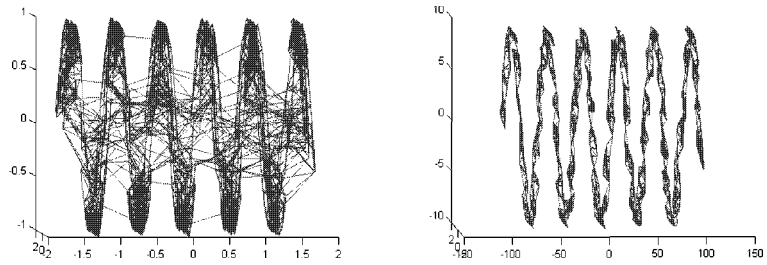

Fig. 6: Results for $3-D$ sinusoïde

3 Impact on the geodesic distance

In this section we want to observe the impact of our normalization on the geodesic distance. As the theoritical study is in progress we can only give empirical results based on examples. We, here, simulate sinusoïdal sets $X^1 \hookrightarrow U[0,1]$ and $X^2 = sin(\omega X^1)$ so the "true" geodesic distance $d_c(X_i, X_j)$ only depends of $|X_i^1 - X_j^1|$.

We randomize 100 points.

For example we observe $8-$nearest neighbors graph of Y and the points $d_c(\hat{Y}_i, Y_j$ v.s. $|X_i^1 - X_j^1|$.

were Y is, first, X normalized by division by standard deviation, then X MST-based normalized. And $\hat{d}_c(Y_i, Y_j)$ is the Didjkstra's geodesic approximation with a $8-$nearest neighbors graph.

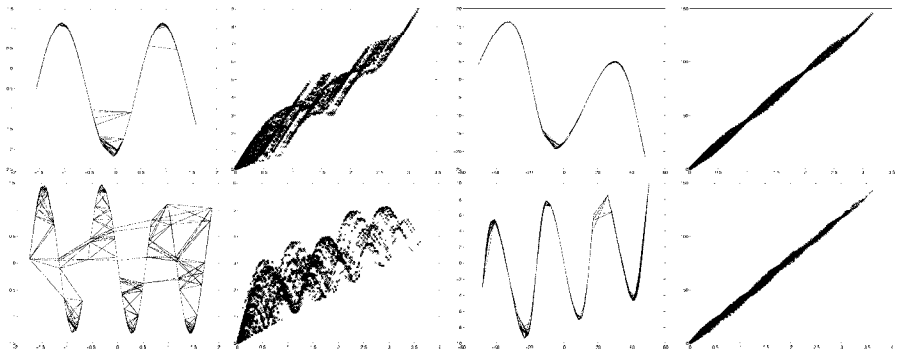

Fig. 7: $8-$nearest neighbors graph and plot of $d_c(\hat{Y}_i, Y_j$ v.s. $|X_i^1 - X_j^1|$ for classical normalization and MST-based normalization 1 $\omega = 10$ and 2 $\omega = 20$

As there is only 100 points (Dijkstra's algorithm is very heavy) we can only test small frequencies but results for geodesic computation are quite promising.

4 Conclusion and further work

We can observe that the proposed algorithm might be useful for preliminary treatment before applying non-linear data analysis methods such as those that use geodesic distance. We would now like to prove it and to determinate which graph choice is the more relevant to be used in the algorithm. Empirically MST and $1-$nearest neighbors seems to be better but we would like to know why.

References

[1] E.W. Dijkstra, A note on two problemes in connection with graphs, *Mathematics*, (1):269-271, 1951.

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

[3] J.A. Lee, A. Lendasse and M. Verleysen, Curvilinear Distance analysis versus isomap In M. Verleysen, editor, *proceedings of the* 8^{th} *European Symposium on Artificial Neural Networks* (ESANN 2000), d-side pub., pages 13-20, April, Bruges (Belgium), 2000.