Evaluation of Euclidean Embeddings for Streamlines
Research project

Kateryna Konotopska
198234
kateryna.konotopska@studenti.unitn.it

Abstract

Complex data types such as documents, images, and, in our case, brain tractography are used for a lot of applications. In particular, when it comes to the data mining or machine learning domain, for applications such as clustering or nearest neighbor queries, it is important to have a vector representation of the data; one way to address this problem is to embed the original data into a vector space with a new distance function approximating the original distance. In this report we review Euclidean embeddings, which allow us to use the Euclidean distance as distance between embedded objects. We compare different variations of the Lipschitz embedding on the dMRI dataset in order to evaluate which of these embedding techniques are the most suitable for the given domain.

1 Introduction

The grey matter is the surface of the brain and is composed of neurons, the processing units of the brain. The internal part of the brain is the white matter, composed of the axons of the neurons, which connects the processing units. With diffusion magnetic resonance imaging (dMRI) and tractography techniques, it is possible to reconstruct in vivo the geometry of the pathways of the axons at the millimiter level. Such anatomical information is crucial for neuroscientists and clinicians in order to understand the structure and function of the brain. See figure 1.

White matter fiber tracts are reconstructed as a set of streamlines. The streamline is a 3D polyline which approximates the common pathway of thousands of axons from dMRI data. Typically, from the white matter of a single brain, it is possible to obtain approximately 1 million streamlines, collectively called the tractogram.

The analysis of the tractogram often requires algorithms from machine learning / pattern recognition and computational geometry, such as clustering and nearest neighbor queries. Given the large number of streamlines in a tractogram, scalability is a main concern. Usually, the most efficient algorithms require a vectorial representation of the objects they manipulate. Unfortunately, streamlines are not unidimensional vectors containing a fixed set of features, instead they are sequences of 3D points with non homogeneous lengths. For this reason, their vector representation is an interesting problem.

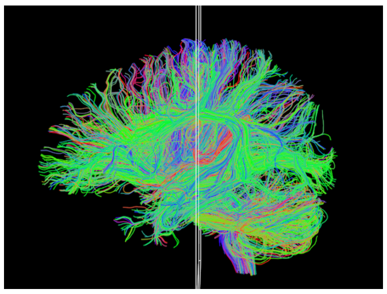
Figure 1: An example of tractogram

The general problem to transform objects into vectors is called Euclidean embedding, which is usually applied to many domains. Embedding techniques require to have a distance function between objects and they aim to approximate such distance with the Euclidean distance between the corresponding embedded objects.

In this work we compare some main Euclidean embedding techniques on entire tractograms and quantify the degree of approximation of each one, with respect to the amount of time required by the computation. A main concern in the given context is scalability, so we address algorithms that can embed one million objects on a standard computer in a short time.

To best of our knowledge, such comparison is not present in the neuroscientific literature and only in (Olivetti et al., 2012) a Euclidean embedding was studied; i.e.: the Dissimilarity Representation, which belongs to a family of Euclidean embeddings called Lipschitz embeddings. The aim of this report is to examine some other embeddings of this family.

In section 2 we start from the definitions of the evaluation metrics for embeddings, quantifying some important properties for a Euclidean embedding. It is followed by the description of methods, starting from the Lipschitz embedding, having a property of limited distortion, and by other methods such as Dissimilarity Representation, LMDS and Fastmap. Section 3 describes in details the experiment setup and santiy checks for the methods and obtained results in terms of performance variability and execution times. The discussion of the results and behavior analysis is described in section 4. Finally as a conclusion, in section 5 optimal performances with corresponding configurations are reported, in order to give the idea of which can be the most suitable algorithm for the given task.

2 Methods

In this section we briefly introduce embedding methods and evaluation metrics studied and implemented in this project. We start with some basic concept and notation and then proceed with description of the Lipschitz embedding and its important properties. Then, we introduce its approximations such as the Dissimilarity Representation, LMDS and Fastmap, which will be used in the experiments of Section 3.

2.1 Basic Concepts and Notation

From (Samet, 2006):

Metric (on a set S): a function $d : S \times S \rightarrow R$, satisfying the following conditions: non-negativity, identity, symmetry and triangle inequality.

Metric space: a set S of objects with some distance metric d, where $S \subset U$ is a dataset of N objects drawn from a universe U, and $d : U \times U \rightarrow \mathbb{R}^+$ is the original distance function on U.

Euclidean embedding: the injective mapping $F : S \rightarrow R^k$ with a distance metric d', where d' is a Euclidean distance metric. A Euclidean embedding is said to have low distortion if $||F(a) - F(b)||_2 \approx d(a, b)$.

2.2 Evaluation metrics

In order to evaluate the quality of a Euclidean embedding, i.e. the degree of approximation between the original distances on U and the Euclidean distance between the corresponding vectors, we describe some of the most common metrics.

2.2.1 Distortion

Distortion is defined by the product $c_1 \cdot c_2$ where c_1 and c_2 are the maximum "shrinking" and "stretching" factors of the distance function in the embedded space with respect to the distance function in the original space:

$\forall o_1, o_2 \in S$:

$$\frac{1}{c_1} \cdot d(o_1, o_2) \leq d'(F(o_1), F(o_2)) \leq c_2 \cdot d(o_1, o_2) \tag{1}$$

2.2.2 Stress

Stress is used to measure the overall deviation in distances and is defined as:

$$\frac{\sum_{o_1,o_2} (d'(F(o_1), F(o_2)) - d(o_1, o_2))^2}{\sum_{o_1,o_2} d(o_1, o_2)^2} \tag{2}$$

2.2.3 Correlation

Pearson correlation coefficient $\rho_{d,d'}$ between the original distance and the Euclidean distance of the embedded objects.

2.3 Lipschitz embedding

Let S be a set of objects from original metric space with metric function d. The Lipschitz embedding is defined in terms of the set R of subsets of S:

$$R = \{A_1, A_2, ..., A_k\} \tag{3}$$

where $A_i \subset S$ is called i-th *reference set* of the embedding. An embedding of object $x \in S$ with respect to R is defined by distances of x to each of the reference sets, where distance is defined as:

$$d_A(x) = \min_{o \in A} d(o, x) \tag{4}$$

These sets can be interpreted as axes of coordinate system of the embedding. The main idea of the Lipschitz embedding and its approximations consists in the following observation: given three objects x, o_1 and o_2, and the distances of o_1 and

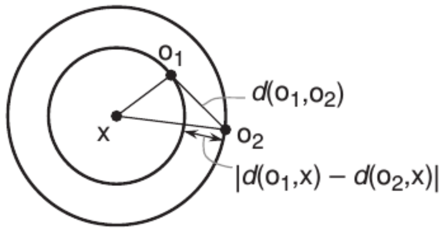

Figure 2: Image showing distance bound $|d(o_1, x) - d(o_2, x)| \leq d(o_1, o_2)$, assuming that objects x, o_1 and o_2 are points and distances are represented by line segments between them (Samet, 2006)

o_2 from x, we can have some information about distance between o_1 and o_2, see Figure 2.

From this observation it can be easily proved that $|d(o_1, A) - d(o_2, A)|$ is the lower bound on $d(o_1, o_2)$.

2.3.1 Selecting reference sets

Defining R means to set: the number of reference sets k, the size of each reference set $|A_i|$ and the policy with which we select objects for each reference set.
Objects for each reference set were chosen randomly from the dataset.
Number and sizes of reference sets were chosen according to different approaches:

- according to the definition proposed by (Linial et al., 1995): number of reference sets $k = \lfloor log_2 N \rfloor^2$, each of sizes $\#A_i = 2^j$, where $j = \lfloor (i-1)/(log_2 N + 1) \rfloor$.
  Let $F(x)$ be the Lipschitz embedding of $x \in S$ and let d' be the Euclidean distance. Given this value of k and sizes of reference sets, it is proved that F satisfies the following inequality:

$$\begin{aligned} &\frac{c}{\lfloor log_2 N \rfloor} \cdot d(o_1, o_2) \\ &\leq d'(F(o_1), F(o_2)) \\ &\leq d(o_1, o_2) \end{aligned} \tag{5}$$

  where c is a constant; i.e. the embedding has limited distortion.

- manual dimensions settings: different custom settings of k were tried, while dimensions of reference sets are calculated by formula proposed by Linial

Algorithm 1 Lipschitz embedding

```
    Input dataset $D$, distance function $d$, target dimension $k$, boolean $Linial1994$
    Output dataset $D'$
 1: procedure LIPSCHITZ
 2:     $N \leftarrow len(D)$
 3:     if $Linial1994 = True$ then
 4:         $k \leftarrow \lfloor log_2 N \rfloor^2$
 5:     $i \leftarrow 1$
 6:     for $i \leq k$ do
 7:         $j \leftarrow \lfloor (i-1)/(log_2 N + 1) \rfloor$
 8:         $sizeA \leftarrow 2^j$
 9:         $A[i] \leftarrow$ choose randomly sizeA objects from D
10:         $i \leftarrow i + 1$
11:     $D' \leftarrow$ compute distances from reference sets (D,R)
```

2.4 Dissimilarity Representation

The Dissimilarity Representation can be seen as a particular instance of Lipschitz embedding, with each reference set composed by only one object: $A_1 = \{o_1\}, A_2 = \{o_2\}, ..., A_k = \{o_k\}$. In this case the Euclidean embedding of an object is represented again by distances of this object from the reference sets:

$$F(x) = (d(x, o_1), d(x, o_2), ..., d(x, o_k)) \tag{6}$$

2.4.1 Selecting reference objects

The objects for reference sets were selected following policies proposed in (Olivetti et al., 2012):

- Random selection
- Farthest First Traversal (FFT)
- Subset Farthest First (SFF)

2.5 Landmark Multidimensional Scaling (LMDS)

In classical multidimensional scaling, an embedding is represented by the "projection" of the object on a set of vectors which have the largest variance possible. In order to obtain such vectors eigenvectors/eigenvalues decomposition of the distance matrix D of the entire dataset is computed, then the first k eigenvectors corresponding to the highest eigenvalues are chosen.
The bottleneck of this method is the calculation of the distance matrix D of the whole dataset, when

Algorithm 2 Dissimilarity embedding

```
**Input** dataset *D*, distance function *d*, target dimension *k*, selection policy *policy*
**Output** dataset *D'*
1: procedure DISSIMILARITY
2:     $N \leftarrow len(D)$
3:     $prototypes \leftarrow$ choose prototypes (D, d, k, policy)
4:     $i \leftarrow 1$
5:     for $i \leq N$ do
6:         $j \leftarrow 1$
7:         for $j \leq k$ do
8:             D'[i,j] $\leftarrow$ d (D[i], prototype[j])
9:             $j \leftarrow j + 1$
10:        $i \leftarrow i + 1$
```

its size is large, therefore an approximation of this method is used instead. Such approximation is called Landmark MDS. It computes eigenvectors/eigenvalues decomposition on a smaller subset of dataset objects, called *landmarks*, akin to the reference objects of Section 2.4.

In the final embedding $F(x)$ the point is represented by distances of x from the landmark points, projected on the "normalized" eigenvectors: divided by the square root of their corresponding eigenvalues.

2.5.1 Landnmarks selection policy

The Policies used for selecting the landmarks were:

- Random selection.
- Minmax policy described in (de Silva and Tenenbaum, 2004). This is similar to FFT and SFF mentioned in Section 2.4.1.

2.6 Fastmap

2.6.1 Method description

This method is based on the projection of the objects $x \in S$ onto k mutually orthogonal directions. Each of these directions is determined by two points from the dataset, called also *pivot* points. Reference lines are determined iteratively: on each iteration, given a distance d_i, two furthest points with respect to this distance are computed (or more precisely their approximation). The line determined by these points represents the *ith* reference line and it is used, together with current distance function d_i, to compute the next distance

Algorithm 3 LMDS embedding computation

```
**Input** dataset *D*, distance function *d*, target dimension *k*, landmarks number *l*, selection policy *policy*
**Output** dataset *D'*
1: procedure LMDS
2:     $N \leftarrow len(D)$
3:     $landmarks \leftarrow$ choose landmarks (D, d, k, l, policy)
4:     Dl $\leftarrow$ compute squared distance matrix (landmarks)
5:     $\Lambda, U \leftarrow$ compute eigenvalues/eigenvectors decomposition (Dl)
6:     $M^{\#} \leftarrow \Lambda^{-1/2}U^T$
7:     $i \leftarrow 1$
8:     for $i \leq N$ do
9:         $j \leftarrow 1$
10:        for $j \leq l$ do
11:            d'[i,j] $\leftarrow d(D[i], landmark[j])^2$
12:            $j \leftarrow j + 1$
13:        $i \leftarrow i + 1$
14:    $\mu \leftarrow$ column mean (Dl)
15:    $D' \leftarrow \frac{1}{2}M^{\#}(\mu - d')$
```

function d_{i+1}, which represents the distance of the projections of points on the hyperplane perpendicular to the *ith* reference line.

2.6.2 Selecting pivot objects

It is assumed that projecting objects onto the line with big *spread* (distance between two furthest objects laying on that line) gives us more information from the projected values.

Also it is assumed that distance d of the original space satisfies the triangle inequality, otherwise some properties of the embedding are not guaranteed.[1]

On each iteration we compute the approximation of two furthest objects r and s and compute the projection of each object $a \in S$ onto this line. From the triangle inequality the value of such projection on *ith* iteration is given by:

$$x_a^i = \frac{d_i(r,a)^2 - d_i(s,a)^2 + d_i(r,s)^2}{2 \cdot d_i(r,s)} \tag{7}$$

The distance function at each iteration is defined in the following way:

[1]One of them is pruning property: it is satisfied if and only if $d'(F(a), F(b)) \leq d(a,b) \forall a, b \in S$ [definition from (Samet, 2006)]

1. $d_1(a,b) = d(a,b)$
2. $d_2(a,b)^2 = d_1(a,b)^2 - (x_a^1 - x_b^1)^2$
3. $d_i(a,b)^2 = d_{i-1}(a,b)^2 - (x_a^{i-1} - x_b^{i-1})^2 = d(a,b)^2 - d_e(F_{i-1}(a), F_{i-1}(b))^2$

where F_i is the embedding with first i coordinates and d_e is the Euclidean distance. Examples and complete description can be found in (Faloutsos and Lin, 1995)

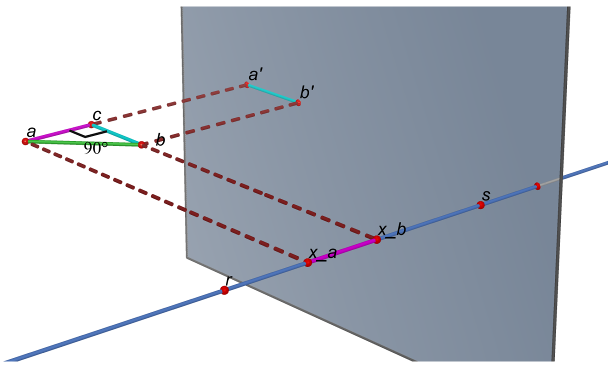

Figure 3: Image showing that $d_2(a,b)^2 = d_1(a',b')^2 = d_1(a,b)^2 - (x_a^1 - x_b^1)^2$

Algorithm 4 Fastmap embedding computation

Input dataset *D*, distance function *d*, target dimension *k*

Output dataset *D'*

```
1: procedure FASTMAP
2:     $N \leftarrow$ len(D)
3:     $i \leftarrow 1$
4:     for $i \leq N$ do
5:         $j \leftarrow 1$
6:         for $j \leq k$ do
7:             $r_j, s_j \leftarrow$ choose pivot points (D, D', d)
8:             $D'_{ij} \leftarrow$ compute projection ($D_i$, D', d, r, s)
9:             $j \leftarrow j + 1$
10:        $i \leftarrow i + 1$
```

3 Experiments

3.1 Dataset

We compare the results of the embedding methods on simulated n-dimensional vectors in Euclidean space and on the dataset of interest: real tractographies reconstructed from dMRI recordings of the human brain.

3.1.1 Simulated data

For the sanity check of the algorithms 100.000 50-dimensional points were randomly generated. Values of each dimension are float values (64 bits) ranging from 0.0 to 1.0, drawn from the uniform probability distribution.

3.1.2 dMRI dataset

The experiments have been carried on the dMRI (diffusion-weighted magnetic resonance imaging) recordings datasets of three different subjects from the publicly available Human Connectome Project dMRI dataset. For each subject tractograms are composed of 400-500 thousands of streamlines. Each streamline is a sequence of 3D coordinates of varying length: from 12 to ~500 points.
The distance function used in this case is the *mam* distance, defined as follows:

$$d_{mam}(s_a, s_b) = \frac{d_m(s_a,s_b) + d_m(s_b,s_a)}{2} \quad (8)$$

where

$$d_m(s_a, s_b) = \frac{1}{|s_a|} \sum_{x_i \in s_a} min_{x_j \in s_b} ||x_i - x_j||_2 \quad (9)$$

s_a and s_b are streamlines a and b.

3.2 Experiment setup

Each embedding was tried on the whole dataset of each of the three subjects. Evaluation metrics of the embeddings were calculated on smaller subsets of the dataset, each of 10.000 randomly extracted streamlines. For each subject, the same evaluation subset was used to evaluate different embeddings. We tried each embedding method 100 times without fixing seeds of randomness in our algorithms, in order to see the mean and the standard deviation of the results of each method.
Embedding sizes of all the methods had the same set of values. Additionally in case of LMDS embedding different numbers of landmarks were tried.
For the dMRI dataset we used as a baseline the resampling method, in which a fixed number of 3D coordinates is used to represent each streamline; they are results of the downsampling or upsampling process. Each triple of coordinates are then concatenated in a single vector, the Euclidean distance is used to evaluate distance between resampled in this way streamlines.

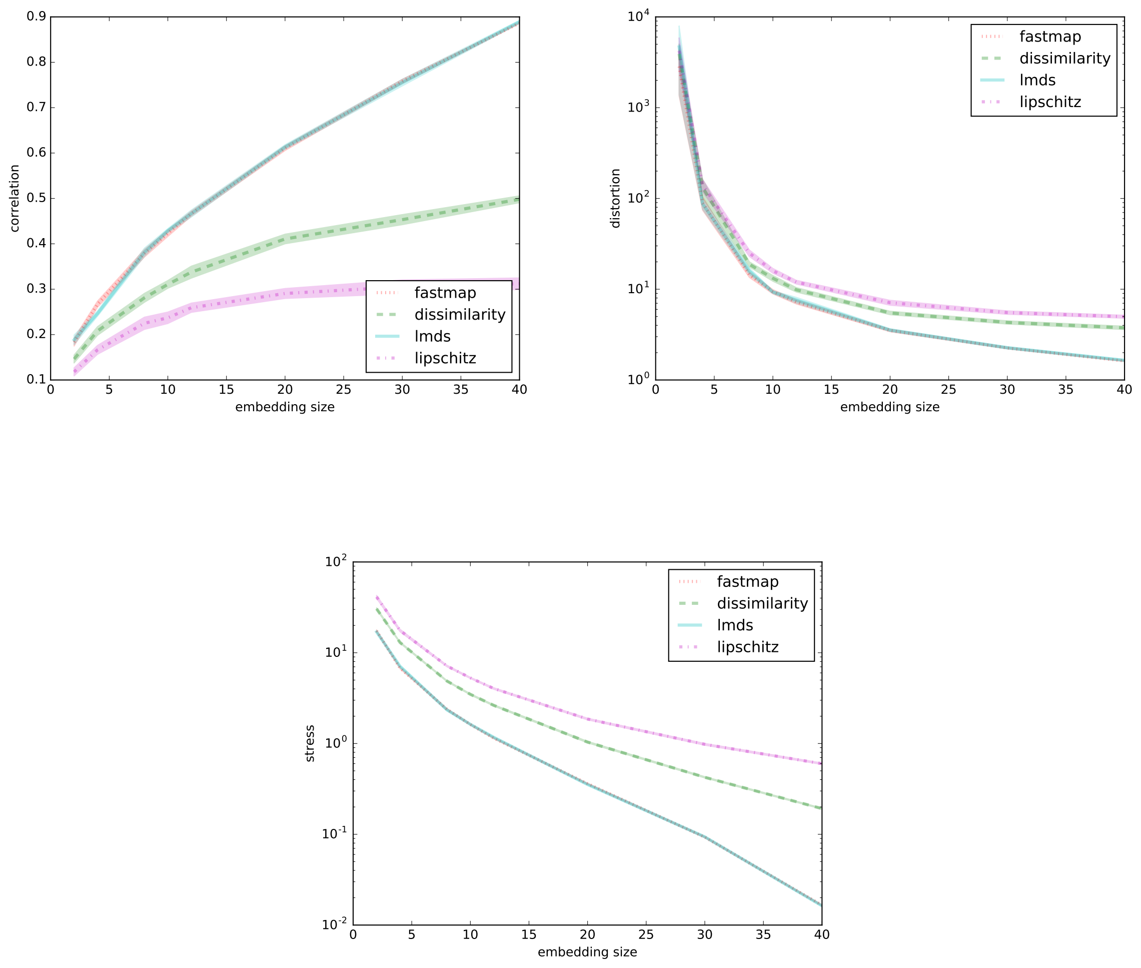

Figure 4: Results of evaluation of embeddings on randomly generated data, number of landmarks in LMDS is set to 60. The results of fastmap & LMDS overlap in all graphs

.

3.3 Results

3.3.1 Simulated data

Since the distance metric of the original space is the Euclidean distance, the execution time of all methods are similar (considering that Euclidean distance computation is pretty fast): they range between 1 and 3 seconds, only Lipschitz embeddings presented higher computation times: from 1 to 6 seconds. Therefore it is not interesting in this case the analysis of the running time. 10 runs were tried to estimate the standard deviation of the methods.
The results are reported in figure 4.

3.3.2 dMRI dataset

Results among different subjects did not present significant difference, therefore the result of only one subject is presented. In figure 5 evaluation of correlation, stress, distortion and execution time are reported.

4 Discussion

4.1 Simulated data

All algorithms behaved as expected on the simulated dataset: the LMDS and fastmap methods obtained the best results in terms of correlation, stress and distortion, while dissimilarity and lipschitz have worse results. The computation times of all methods are similar, except the case of Lip-

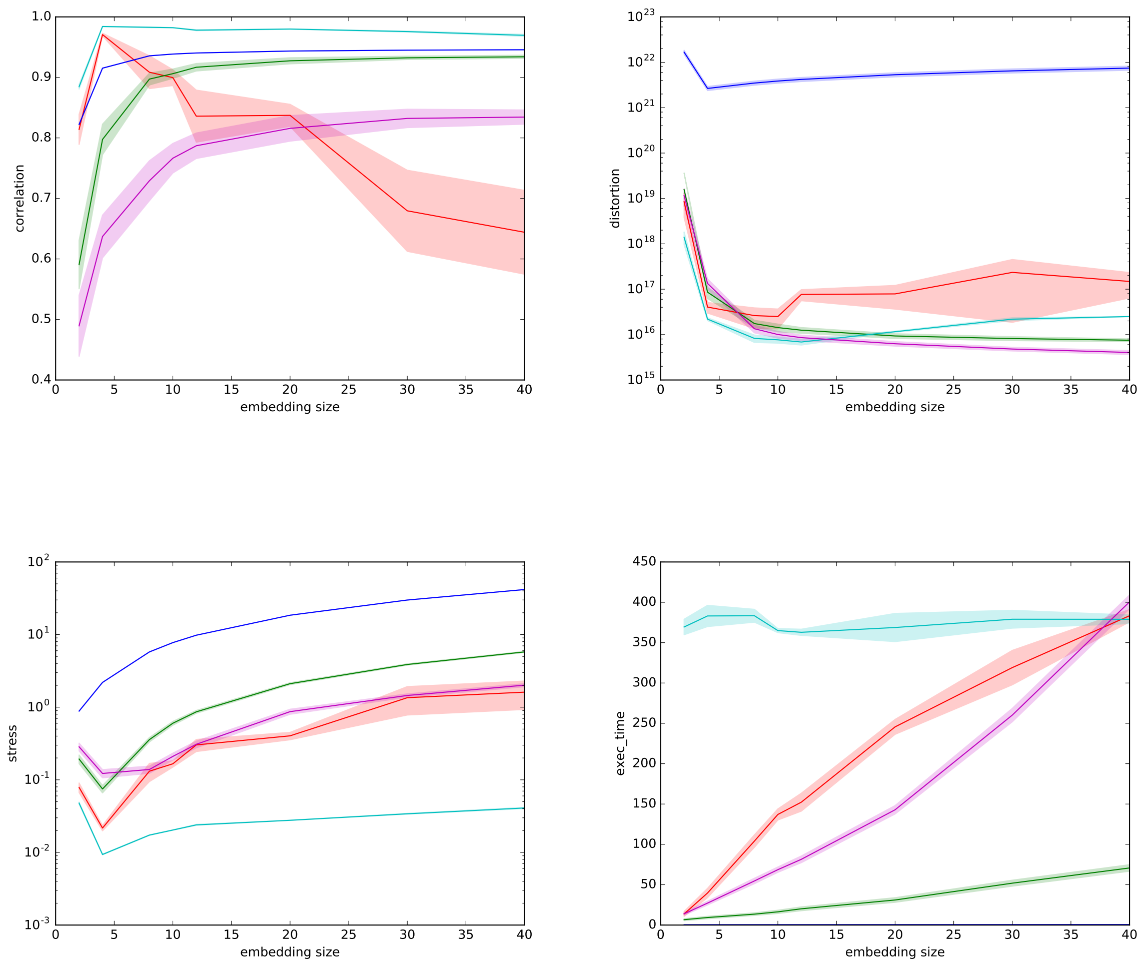

Figure 5: Results of evaluation of embeddings on single subject: Fastmap, Dissimilarity, LMDS, Lipschitz, Resampling; number of landmarks in LMDS is set to 150

schitz embedding, in which it was significantly slower (almost doubled with respect to the other methods).

4.2 dMRI dataset

Before discussing the results obtained from experiments on the dMRI data lets recall that in this case the original distance *mam* is not a metric and the prunning property is not guaranteed.
Each embedding method is analyzed with the increasing size of embeddings (k).
As we can observe from the figure 5, all methods lead to the increment of stress measure values. It is because the values of the embedded distances are prevalently bigger then the original ones, since the prunning property is not guaranteed. The lowest stress values are presented by the LMDS algorithm since it applies a kind of normalization of the embedding coordinates.
LMDS algorithm presented the best correlation and stress values, while it is the slowest method in terms of computation time. (See table 1)
Resampling method seems to have a good correlation and its execution is very fast, but it introduces a big distortion with comparison to the other methods.
Both fastmap and LMDS algorithms at some point present a loss of quality as more dimensions are added. The similar behavior of fastmap embedding can be observed also in (Vassilis Athitsos and Kollios, 2004). A possible explanation may be the fact that projections in non-Euclidean spaces in-

troduce a loss of information because of the lack of pruning property. In the case of fastmap the projection operation is done recursively, and it may add further information loss after each iteration. So for these methods there is a kind of trade-off between addition of new dimensions and distortion introduced by adding them.
Since the errors in fastmap may be propagated during each iteration, the choice of the hyperplane in each step may determine the quality of each subsequent iteration, and since the choice in this case is a random process it is reasonable to observe a bigger standard deviation of this method.

5 Conclusions

The LMDS and fastmap methods obtained the best results for small embedding sizes. The LMDS method is the slowest one, nevertheless its approximations with lower number of landmarks obtained quite good results (in case of small embedding sizes).
The highest correlation scores with corresponding execution times of each method are summarized in table 1.

Algorithm	Correlation	Exec. time	k
LMDS, 30 landmarks	0.982	66.1s	4
LMDS, 50 landmarks	0.983	117.8s	4
LMDS, 100 landmarks	0.985	240.7s	4
LMDS, 150 landmarks	0.985	398.8s	4
Fastmap	0.976	42.1s	4
Dissimilarity	0.946	53.9s	40+
Resampling	0.947	0.6s	40+
Lipschitz	0.890	254.3s	40+

Table 1: Best results of each method (k is the embedding size)

6 GitHub project

The project was implemented on top of numerical libraries and neuroscientific libraries for the Python language. The libraries of Euclidean embeddings can be found at the following link:
https://github.com/emanuele/euclidean_embeddings

The experiment set:
https://github.com/katerynak/embeddings

References

Christos Faloutsos and King-Ip Lin. 1995. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD Rec.*, 24(2):163–174.

Nathan Linial, Eran London, and Yuri Rabinovich. 1995. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245.

E. Olivetti, T. B. Nguyen, and E. Garyfallidis. 2012. The approximation of the dissimilarity projection. In *2012 Second International Workshop on Pattern Recognition in NeuroImaging*, pages 85–88.

H. Samet. 2006. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann series in computer graphics and geometric modeling. Elsevier/Morgan Kaufmann.

Vin de Silva and Joshua B. Tenenbaum. 2004. Sparse multidimensional scaling using landmark points.

Stan Sclaroff Vassilis Athitsos, Joni Alon and George Kollios. 2004. Learning euclidean embeddings for indexing and classification.