

Visualizing the Effects of a Changing Distance on Data Using Continuous Embeddings

Gina Gruenhage^{1,*}

*Department of Computer Science, Technische Universität Berlin and BCCN Berlin,
Berlin, Germany*

Manfred Oppel

Department of Computer Science, Technische Universität Berlin, Berlin, Germany

Simon Barthelme

CNRS, GIPSA-lab, 11 Rue des Mathématiques, 38400 Saint-Martin-d'Hères, France

Abstract

Most Machine Learning (ML) methods, from clustering to classification, rely on a distance function to describe relationships between datapoints. For complex datasets it is hard to avoid making some arbitrary choices when defining a distance function. To compare images, one must choose a spatial scale, for signals, a temporal scale. The right scale is hard to pin down and it is preferable when results do not depend too tightly on the exact value one picked. Topological data analysis seeks to address this issue by focusing on the notion of neighbourhood instead of distance. It is shown that in some cases a simpler solution is available. It can be checked how strongly distance relationships depend on a hyperparameter using dimensionality reduction. A variant of dynamical multi-dimensional scaling (MDS) is formulated, which embeds datapoints as curves. The resulting algorithm is based on the Concave-Convex Procedure (CCCP) and provides a simple and efficient way of visualizing changes and invariances in distance patterns as a hyperparameter is varied. A variant to analyze the dependence on multiple hyperparameters is also presented. A cMDS algorithm that is straightforward to implement, use and extend is provided. To illustrate the possibilities of cMDS, cMDS is applied to several real-world data sets.

Keywords: Dimensionality Reduction, Multidimensional Scaling, Visualization, Data Exploration

*Corresponding author

Email addresses: gina.gruenhage@bccn-berlin.de (Gina Gruenhage), manfred.opper@tu-berlin.de (Manfred Oppel), simon.barthelme@gipsa-lab.fr (Simon Barthelme)

¹Full Address: Gina Gruenhage, TU Berlin, Fakultät IV, Elektrotechnik und Informatik, Sekr. MAR 4-2, Marchsstrasse 23, D-10587 Berlin, Tel: 0049-30-314-75729, Fax: 0049-30-314-24913

1. Introduction

The notion of distance is at the core of data analysis, pattern recognition and machine learning: most methods need to know how similar two datapoints are. The choice of distance metric is often a hidden assumption in algorithms. For complex data, distance or similarity are not uniquely defined. On the contrary, they can be arbitrary to some extent [1]. It is, for example, often possible to describe signals on different temporal or spatial scales, and distance functions will give a certain scale more weight than another. Each datapoint might describe several features, and there is often no unique, optimal way to weigh the features when computing a distance measure: are two individuals more alike if they have similar eye colour or hair colour, or do we think the shape of the nose matters most?

There are ways around that problem. One is to select the distance function that is best adapted to the task at hand, for example the one that gives the best performance in classification (this is effectively what is done in kernel hyperparameter selection [2]). Another is to give up on distance and rely instead on the weaker notion of neighbourhood [3].

We argue here that a third option is available. One may study how the shape of the data evolves under a change in the distance metric by representing the data in lower dimension. We suppose that a family of distance functions $d_\alpha(x, y)$ is defined by varying a hyperparameter $\alpha \in [0, 1]$, where α can represent, for example, different scales or the mixing proportion of features. Please note that α does not have to be defined on this interval, but it seems natural to start with a setting that is familiar from, e.g., convex combinations. Suppose that for a given level of α the relative distances between datapoints are well described by representing the datapoints as points on the line. As we vary α the points will move, so that each point now describes a curve. Many scenarios are possible, and we sketch them in Fig. 1. We may have full or partial *invariance*: patterns in the data that hold regardless of the value of the hyperparameter (Fig. 1A). On the other hand, the structure in the data may appear only for certain values of α (intermediate values in Fig. 1B and rather small values in C), indicating that these values are more useful than others for characterizing the data. Analyzing the evolution of structures in the data might reveal interesting dependencies, for example, declustering (Fig. 1C) or loss of information (Fig. 1D).

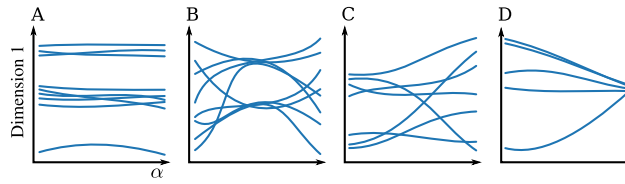


Figure 1: Sketches of different effects on the data structure that emerge when varying a hyperparameter in a distance function. The x -axis shows the hyperparameter α , the y -axis is the embedding dimension. A) Invariance: patterns hold independent of the hyperparameter. B/C) Structure emerges only for certain values of the hyperparameter. C) Declustering: clusters are lost with increasing hyperparameter. D) Information loss: Structure collapses with increasing hyperparameter.

To visualize the effects of varying the distance function we suggest to embed data into a space of smooth curves, forming what we call continuous embeddings: in continuous embeddings each datapoint is embedded as a smooth curve in \mathbb{R}^d . We will show that this approach is quite general.

Our implementation of continuous embeddings is based on multi-dimensional scaling (MDS), one of the most widely-used tools for dimensionality reduction [4, 5]. MDS builds on the pairwise relation between single data points and has an intuitive way of characterizing the structure in high-dimensional data. MDS supposes that one has distance information available, that is, we can characterize the data by a distance matrix. MDS seeks to find a set of points in a low dimensional Euclidean space, such that the Euclidean distances between points approximate the original distances. An exception is *spherical* MDS, where the embedding is constrained to a spherical manifold. MDS goes back to the 1950s, when it was first introduced as classical scaling [6]. In classical scaling, the distance matrix is transformed to a matrix of inner products from which an embedding can be computed using eigendecompositions [6, 7, 8]. Classical scaling finds a perfect embedding when the data can indeed be embedded exactly, but in all realistic cases distance matrices are not exactly Euclidean and *distance* scaling is more appropriate. Kruskal [9] introduced distance scaling by defining a cost function, *Stress*, that directly measures the error between original and embedding distances. This cost function is then optimized over the space of embedding matrices which can be done using gradient descent. Since the early work on MDS many other variants and optimization solutions have been discussed. So called non-metric variants of MDS seek to only recover the ranks of distances [10]. Ramsay [11, 12] introduces a statistical model for MDS, allowing for a maximum likelihood estimate. This approach is implemented in *Multiscale* [13]. Other MDS variants based on Stress include Sammon’s mapping [14], elastic stress [15], multidimensional unfolding [16] and local MDS [17]. Isomap [18] is also related to MDS. Here, distances are computed as geodesic distances on a manifold, which are then embedded with classical scaling. In terms of optimization one of the most popular approaches is SMACOF, a majorization method for MDS [19, 20, 21, 22].

Here, we introduce a continuous version of MDS (cMDS) by adding a smoothing penalty to the MDS cost function. Similar ideas have been used in the visualization of dynamic networks. A network is commonly represented as a graph. A 2D embedding of a static graph is often constructed using MDS or similar methods [23, 24]. In the dynamical context, where a graph is measured over time, it is important to preserve the so-called “mental map” when jumping from one timepoint to the next [25]. Early work on such *controlled stability* was done by Böhringer and Paulisch [26], North [27]. Brandes and Wagner [28] developed a more rigorous formulation of controlled stability based on regularization in a Bayesian framework. There have been three different approaches to the problem of preservation of the mental map: aggregation, anchoring and linking. In aggregation methods, the graph is aggregated into an average graph which is then visualized with a static layout algorithm [29, 30]. Anchoring methods use auxiliary edges which connect nodes to stationary reference positions [28, 31, 32]. In linking, edges are created that connect instances of a single vertex over time. The resulting graph is then visualized using standard methods [33, 34, 35]. Linking has been formulated in more rigorous ways in terms of regularized cost functions [36, 37]. Xu et al. [38] introduce an additional grouping penalty. Brandes and Mader [37] provide a good overview on dynamic graph layout. Another approach to dynamic embeddings is an extension of the

Hoff latent space model [39]. In *functional MDS* [40], individual solutions to the MDS problems are rotated in a secondary step to minimize the length of the curves. This does not allow for control of smoothness versus stress, which is true for most methods described above.

All these approaches and contributions in the field visualize temporal developments. We show here that continuous embeddings can be applied to a great variety of data, going far beyond the visualization of temporal dynamics in graphs. In particular, the continuous variable can be used to visualize artificial dynamics. This makes the method very general and is especially useful in the analysis of families of distance functions and their effects on data structure. Continuous embeddings are thus a tool for making an informed choice of the distance metric for use in further analyses.

We show how continuous embeddings can be efficiently computed using the Concave-Convex Procedure (CCCP) for optimization [41]. The resulting algorithm is a simple iterative procedure, in which the inner loops are nothing more than least squares regression with smoothing splines. We prove that the algorithm always leads to a stationary point. This goes further than other proofs [42, 43] because the cost function is nondifferentiable at certain points and doesn't share the directional derivative with the upper bound at those points. We provide an R package for cMDS which is available on github: github.com/ginagruenhage/cmdsr. We illustrate the results of cMDS with several examples. We compare cMDS to a method based on k -means to exemplify that cMDS provides a more informative way of understanding the data structure. We show that cMDS leads to novel forms of data visualization and enhances the analysis of various meta-effects in data, such as hierarchy levels in hierarchical clustering, weighting of different distance measures and consensus requirements across subjects. Furthermore, we point out that quantitative analyses on cMDS results are possible and useful. We show that cMDS is especially well-suited to dynamic and interactive contexts [44]. We provide several examples of interactive, web-based visualizations based on cMDS.

2. Methods

In order to present the cMDS algorithm, we introduce some notations and definitions. We describe the objective of the algorithm and present the cost function that we need to optimize. Optimization can be done in a coordinate-wise manner and we present the optimization of single coordinates via the Concave-Convex Procedure and pseudocode of the full algorithm in Section 2.2. In the subsequent section, we prove that the optimization of single coordinates is first-order optimal, i.e. it leads to stationary points of the original cost. We also show that this directly entails first order optimality of the full cost.

We start by setting the notations. The original data or objects are denoted by $\mathbf{s}_1(\alpha) \dots \mathbf{s}_N(\alpha)$, where N is the number of objects in the data. The objects are defined in an arbitrary metric space, e.g. \mathbb{R}^D . The parameter α measures a continuous dimension. We will see that objects, such as images or networks, can be endowed with a continuous dimension, for example by examining them at different scales, so that scale plays the role of the continuous parameter. At this point we would like to note that we discretize all equations from the beginning, because ultimately, in the implementation, the hyperparameter has to be discretized. It would certainly be possible to develop all the mathematics in a continuous matter. We prefer to present the mathematics in such a

way that the equations in the paper can be used as a direct reference for implementation. However, important equations, such as the cost function and the penalty are given in their continuous form as well, to improve the understanding of the problem definition. Thus, α is represented on a grid $\alpha_1 \dots \alpha_T$. We use T to denote the maximum value of α since time is the most natural framework to think about the hyperparameter. With $f(\cdot)$ we refer to function values at all grid points while $f(k)$ refers to a function value at a specific value of α . The objects are endowed with a distance function $d(x, y)$. The appropriate distance function depends on the data-space and on the nature of the problem. Given a distance measure, we can define a distance array, $\mathbf{D}^{(\cdot)} \in \mathbb{R}^{T \times N \times N}$, where the entry d_{ij}^k holds the distance between objects \mathbf{s}_i and \mathbf{s}_j at α_k . We assume that the distances between datapoints give a good summary of the patterns in the data. The goal of cMDS will be to extract these patterns.

2.1. Objective

The objective of cMDS is to retrieve curves or manifolds in \mathbb{R}^d , $d \ll D$ which we denote as $\mathbf{x}_1^{(\cdot)} \dots \mathbf{x}_N^{(\cdot)} \in \mathbb{R}^{T \times d}$, such that the evolution of distances between the curves represents the evolution of distances between the datapoints. When we talk about curves, we mean the technical differential geometry sense of the word, namely, a curve is a 1-dimensional manifold in \mathbb{R}^d . The curves are represented as a configuration array $\mathbf{X}^{(\cdot)} \in \mathbb{R}^{T \times d \times N}$ and for a given time-point α_k \mathbf{X}^k is a $d \times N$ matrix in which each column represents the coordinates of a curve at time α_k . In \mathbb{R}^d we measure the distance between two curves at α_k with the Euclidean distance. Thus, for each configuration, we have a distance array $\tilde{\mathbf{D}}^{(\cdot)} \in \mathbb{R}^{T \times N \times N}$ such that $\tilde{d}_{ij}^k = \|\mathbf{x}_i^k - \mathbf{x}_j^k\|_2$. To denote that $\tilde{\mathbf{D}}^{(\cdot)}$ is computed from $\mathbf{X}^{(\cdot)}$, we occasionally write $\tilde{\mathbf{D}}^{(\cdot)}(\mathbf{X}^{(\cdot)})$. These are the approximate distances given by our embedding, and the objective is to make the approximate distances as close as possible to the real distances. A natural expression of that objective is the following cost function, which quantifies the distortion of the embedding:

$$\mathcal{L}(\mathbf{X}^{(\cdot)}, \mathbf{D}^{(\cdot)}) = \int_0^1 \left(\|\tilde{\mathbf{D}}^\alpha(\mathbf{X}^\alpha) - \mathbf{D}^\alpha\|_F \right)^2 d\alpha, \quad (1)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This is the MDS cost function for continuous data. Its discretized form is:

$$\mathcal{L}(\mathbf{X}^{(\cdot)}, \mathbf{D}^{(\cdot)}) = \sum_{k=1}^T \left(\|\tilde{\mathbf{D}}^k(\mathbf{X}^k) - \mathbf{D}^k\|_F \right)^2. \quad (2)$$

In practice and for real datasets, MDS is a highly non-convex optimization problem with multiple local minima. Additionally, since distances are invariant to rotations, translations and symmetries, so are the MDS embeddings. Minimizing (2) is equivalent to solving MDS problems for different values of the hyperparameter individually. The individual problems are known as *Kruskal-Shephard* scaling [9]. This would result in solutions that are independent for different α_k and thus might lie in quite different local minima. However, one would expect that slight changes in the hyperparameter lead to only slight changes in the embedding. To solve this problem, we will require that each curve is continuous and smooth, which can be achieved by adding a suitable

penalty function $\Omega(\mathbf{X}^{(\cdot)})$ to the cost. The effect of the smoothing penalty can be interpreted as the goal of tracing one particular local minimum across different values of the hyperparameter. This results in the cost function

$$\mathcal{C}(\mathbf{X}^{(\cdot)}, \mathbf{D}^{(\cdot)}) = \mathcal{L}(\mathbf{X}^{(\cdot)}, \mathbf{D}^{(\cdot)}) + \lambda \Omega(\mathbf{X}^{(\cdot)}). \quad (3)$$

Classical spline penalties [45] are particularly convenient. We introduce the penalty in its continuous form:

$$\Omega(\mathbf{X}(\alpha)) = \int_0^1 \left\| \frac{\partial^2 \mathbf{x}(\alpha)}{\partial \alpha^2} \right\| d\alpha. \quad (4)$$

In practice, we work with discrete one-dimensional manifolds in \mathbb{R}^d for which the penalty reads

$$\begin{aligned} \Omega(\mathbf{X}^{(\cdot)}) &= \sum_d \text{diag} \left(\sum_i (\mathbf{x}_i^{(\cdot)})^T (\mathcal{D}^{(2)})^T \mathcal{D}^{(2)} \mathbf{x}_i^{(\cdot)} \right) \\ &= \sum_d \text{diag} \left(\sum_i (\mathbf{x}_i^{(\cdot)})^T \mathbf{M} \mathbf{x}_i^{(\cdot)} \right), \end{aligned} \quad (5)$$

where $\mathcal{D}^{(2)}$ denotes the discrete second order differential operator. The parameter λ controls how strongly the roughness of the curves is penalized. For $\lambda = 0$, we recover the classical cost function of MDS, with the extension that we have T separate MDS problems, one for each value of α . For $\lambda \rightarrow \infty$ the resulting curves are straight lines, independent of the original data. The value of λ is easy to set by visual inspection. It should simply be large enough to result in fairly smooth curves, while keeping the distortion reasonably small. A reasonable strategy for setting λ automatically is then to maximize λ , under a constraint on the quality of the embedding. This quality can be measured in various ways [46, 47]. The effect of the smoothing parameter λ is shown in a supplementary material file.

2.2. Optimization via the Concave-Convex Procedure (CCCP)

The cMDS algorithm optimizes the cost in a curve-by-curve manner. This is possible since the cost is a sum over costs per curve. Thus, we have an outer loop over curves and an inner loop that performs conditional optimization on $\mathbf{x}_i^{(\cdot)}$. That is, we assume that all curves $\mathbf{x}_j^{(\cdot)}, j \neq i$ are fixed. The inner loop is a Maximization Minimization procedure. Specifically, we use the Concave-Convex Procedure (CCCP) [41]. This way, each minimization step is a simple spline regression. In effect, the algorithm only needs to compute spline regressions with surrogate data.

We now outline the usage of CCCP for the optimization of a single curve. We expand the first term of the cost function to identify the convex and concave parts. We denote the cost of a single curve as $f(\cdot)$:

$$\begin{aligned}
f(\mathbf{x}_i^{(\cdot)}) &= \sum_{kj} \left(\|\mathbf{x}_i^k - \mathbf{x}_j^k\|_2 - d_{ij}^k \right)^2 + \lambda (\mathbf{x}_i^{(\cdot)})^T \mathbf{M} \mathbf{x}_i^{(\cdot)} \\
&= \sum_{kj} (\mathbf{x}_i^k - \mathbf{x}_j^k)^T (\mathbf{x}_i^k - \mathbf{x}_j^k) \\
&\quad - \sum_{kj} 2 d_{ij}^k \|\mathbf{x}_i^k - \mathbf{x}_j^k\| + \sum_{kj} (d_{ij}^k)^2 \\
&\quad + \lambda (\mathbf{x}_i^{(\cdot)})^T \mathbf{M} \mathbf{x}_i^{(\cdot)} \\
&= f_{\text{vex}}(\mathbf{x}_i^{(\cdot)}) + f_{\text{cave}}(\mathbf{x}_i^{(\cdot)}) + \text{const}, \tag{6}
\end{aligned}$$

where

$$f_{\text{vex}}(\mathbf{x}_i^{(\cdot)}) = \sum_{kj} (\mathbf{x}_i^k - \mathbf{x}_j^k)^T (\mathbf{x}_i^k - \mathbf{x}_j^k) + \lambda (\mathbf{x}_i^{(\cdot)})^T \mathbf{M} \mathbf{x}_i^{(\cdot)} \tag{7}$$

and

$$f_{\text{cave}}(\mathbf{x}_i^{(\cdot)}) = - \sum_{kj} 2 d_{ij}^k \|\mathbf{x}_i^k - \mathbf{x}_j^k\|. \tag{8}$$

In the optimization, we can omit the constant term that doesn't depend on $\mathbf{x}_i^{(\cdot)}$. The iterative CCCP algorithm $(\mathbf{x}_i^{(\cdot)})^{t-1} \mapsto (\mathbf{x}_i^{(\cdot)})^t$ is given by

$$(\mathbf{x}_i^{(\cdot)})^t = \arg \min_{\mathbf{x}_i^{(\cdot)}} u(\mathbf{x}_i^{(\cdot)}, (\mathbf{x}_i^{(\cdot)})^{t-1}), \tag{9}$$

where the convex upper bound is computed by taking the first-order Taylor expansion of the concave part of f . The concave part is, however, non-differentiable at $\mathbf{x}_i^k = \mathbf{x}_j^k$. We thus work with a modified subdifferential.

$$\begin{aligned}
&\vec{\nabla}^{SD} f_{\text{cave}}((\mathbf{x}_i^k)^{t-1}) \\
&= 2 \sum_k \sum_i d_{ij}^k \begin{cases} \frac{(\mathbf{x}_i^k)^{t-1} - \mathbf{x}_j^k}{\|(\mathbf{x}_i^k)^{t-1} - \mathbf{x}_j^k\|} & \text{if } (\mathbf{x}_i^k)^{t-1} \neq \mathbf{x}_j^k \\ \{\mathbf{u} : \|\mathbf{u}\| = 1\} & \text{if } (\mathbf{x}_i^k)^{t-1} = \mathbf{x}_j^k \end{cases}. \tag{10}
\end{aligned}$$

In the latter case, \mathbf{u} is chosen randomly. The usual definition of the subdifferential [48] would yield a less strong condition on \mathbf{u} , namely $\|\mathbf{u}\| \leq 1$. We choose the stronger variant to achieve maximum descent in each optimization step. We introduce surrogate points $\hat{\mathbf{x}}_j^{(\cdot)}$.

$$\hat{\mathbf{x}}_j^k = \mathbf{x}_j^k + d_{ij}^k \begin{cases} \frac{(\mathbf{x}_i^k)^{t-1} - \mathbf{x}_j^k}{\|(\mathbf{x}_i^k)^{t-1} - \mathbf{x}_j^k\|} & \text{if } (\mathbf{x}_i^k)^{t-1} \neq \mathbf{x}_j^k \\ \{\mathbf{u} : \|\mathbf{u}\| = 1\} & \text{if } (\mathbf{x}_i^k)^{t-1} = \mathbf{x}_j^k \end{cases}, \tag{11}$$

where, again, \mathbf{u} is chosen randomly.

The resulting upper bound for the original cost is

$$\begin{aligned}
u((\mathbf{x}_i^{(\cdot)})^t, (\mathbf{x}_i^{(\cdot)})^{t-1}) &= \sum_{kj} ((\mathbf{x}_i^k)^t - \mathbf{x}_j^k)^T ((\mathbf{x}_i^k)^t - \mathbf{x}_j^k) \\
&\quad - 2 \sum_{kj} d_{ij} \|(\mathbf{x}_i^k)^{t-1} - \mathbf{x}_j^k\| \\
&\quad - 2 \sum_{kj} ((\mathbf{x}_i^k)^t - (\mathbf{x}_i^k)^{t-1}) (\hat{\mathbf{x}}_j^k - \mathbf{x}_j^k) \\
&\quad + \lambda ((\mathbf{x}_i^{(\cdot)})^t)^T \mathbf{M} (\mathbf{x}_i^{(\cdot)})^t.
\end{aligned} \tag{12}$$

The updating step is thus a simple spline regression and can be performed analytically. Spline regression involves a matrix inversion with cost $\mathcal{O}(T^3)$, but the inverse needs to be computed only once. Once the inverse has been obtained the cost of the spline regression becomes $\mathcal{O}(T^2)$, with further savings possible using sparse matrix techniques. For simplicity of notation, we rewrite the embedding points $\mathbf{x}^{(\cdot)}$ as column vectors in $\mathbb{R}^{T \cdot d}$, $\bar{\mathbf{x}}_i = \text{vec } \mathbf{x}_i^{(\cdot)}$.

$$\bar{\mathbf{x}}_i^t = (N \cdot \mathbf{E}_{T \cdot d} + \lambda \mathbf{E}_d \otimes \mathbf{M})^{-1} \left(\sum_j \bar{\mathbf{x}}_j \right), \tag{13}$$

where \mathbf{E}_d is the identity matrix in $\mathbb{R}^{d \times d}$.

Agarwal et al. [49] introduce the same surrogate points based on geometrical considerations for metric (non-continuous) MDS. There, they are defined as projecting \mathbf{x}_i on the sphere \mathcal{S}_j with center \mathbf{x}_j and radius d_{ij} . In our formulation, they arise naturally from a natural splitting of the cost function into convex and concave parts. Additionally, using the subgradient circumvents the problem of the surrogate points being undefined for $\mathbf{x}_i = \mathbf{x}_j$, which Agarwal et al. [49] do not address.

Together with the iteration over the curves $\mathbf{x}_i^{(\cdot)}$ we have everything we need for the cMDS algorithm (see Fig. 2). The complexity of the algorithm is $\mathcal{O}(dT^2N^2)$ per iteration of the outer loop, since its function MM has complexity $\mathcal{O}(dT^2N)$ due to the matrix product in the spline regressions.

The resulting algorithm can also be interpreted as a majorization-minimization algorithm [20, 50, 51] which is true for any CCCP algorithm [42]. The majorization entails the computation of the surrogate points, while minimization is the computation of the updating step.

2.3. First-order optimality of the cMDS algorithm

The cMDS algorithm has the structure of a coordinate-descent method, with an outer loop that improves the configuration curve-by-curve, and an inner loop that optimises the objective for a given curve (using CCCP). As we will see it is difficult to make very strong statements about the convergence of a coordinate-descent algorithm on a non-convex, non-differentiable objective such as ours. The guarantee we can offer is that, if cMDS converges, then it converges to a local minimum or saddle point. We begin with a study of the inner loop, and discuss the outer loop in the next subsection.


```

function cMDS( $\mathbf{D}^{(\cdot)}, \mathbf{X}_0^{(\cdot)}, \lambda, \mathbf{M}, \delta$ )
    maxIter = 50
    k = 0
    while  $k < \text{maxIter}$  do
        for  $(i = 1 : N)$  do
             $\mathbf{x}_i^{(\cdot)} \leftarrow \text{MM}(\mathbf{D}^{(\cdot)}, \mathbf{X}^{(\cdot)}, i, \text{params})$ 
        end for
         $k \leftarrow k + 1$ 
    end while
    return  $\mathbf{X}^{(\cdot)}$ 
end function

function MM( $\mathbf{D}^{(\cdot)}, \mathbf{X}^{(\cdot)}, i, \text{params}$ )
    repeat
         $\hat{\mathbf{x}}^{(\cdot)} \leftarrow \text{MAJORIZE}((\mathbf{X}^{(\cdot)})^{t-1}, \mathbf{D}^{(\cdot)}, i, \text{params})$ 
         $(\mathbf{X}^{(\cdot)})^t \leftarrow \text{MINIMIZE}((\mathbf{X}^{(\cdot)})^{t-1}, \hat{\mathbf{x}}^{(\cdot)}, i, \text{params})$ 
    until  $\left( \frac{\frac{1}{N} \sum_i |(\mathbf{x}^{(\cdot)})^{t-1} - (\mathbf{x}_i^{(\cdot)})^t|}{\frac{1}{N} \sum_i |(\mathbf{x}_i^{(\cdot)})^t|} > \delta \right)$ 
    return  $\mathbf{x}_i^{(\cdot)}$ 
end function

```

Figure 2: The cMDS algorithm. The maximum number of iterations is set to 50. In practice, this is a sufficient number of iterations for the outer loop. The parameters passed to the MM function include, for example, N, d, T, λ , weights (if used), the penalty matrix \mathbf{M} and the error tolerance.

2.3.1. First-order optimality of the inner loop

Since CCCP has become an important tool in machine learning, there is a vast literature on convergence proofs for CCCP in general. Lanckriet and Sriperumbudur [42] analyze the convergence of CCCP, for smooth and differentiable cost functions. In this case, the authors prove global convergence, in the sense that the algorithm arrives at a stationary point, i.e. a local minimum or saddle point, from any initialization point. The cMDS cost function, however, is nonsmooth and nondifferentiable at $\mathbf{x}_i^k = \mathbf{x}_j^k$. Yen et al. [43] focus on convergence rate of CCCP and work with nonsmooth objective functions. They connect CCCP to more general block coordinate descent methods. They use results on convergence of coordinate descent on nonconvex and nonsmooth problems [52] by showing that CCCP is an instance of block coordinate descent. However, the authors constrain the class of objective functions to which their proof applies. The nonsmooth part should be convex piecewise linear, which is not the case for cMDS. Razaviyayn et al. [53] extend block coordinate descent methods to inexact block coordinate descent. This allows the authors to treat nondifferentiable and nonconvex objective functions. The authors unify convergence results for various methods such as CCCP and Expectation Maximization. We follow ideas of this paper to show that a limit point of the optimization chain of a single curve in cMDS via CCCP is a stationary, first-order optimal point of the cost.

For readability we denote $x = \mathbf{x}_i^{(\cdot)}$ in the following. It is easy to see that, by optimizing single curves via minimization of an upper bound, we monotonically decrease the original cost.

$$f(x^1) \geq f(x^2) \geq f(x^3) \geq \dots \quad (14)$$

However, from this it is not yet clear (assuming convergence of the algorithm) that the resulting configuration is a stationary point of the original cost, since the algorithm could produce a non-increasing sequence that does not tend to an optimal point. Therefore, we need to show that the limit point z of the inner loop is first-order optimal, i.e. $\vec{\nabla} f(x)|_{x=z} = 0$. Then, we prove that this entails first-order optimality of the inner loop. Strictly speaking, this only holds with probability one, with respect to the randomisation step in (10), which chooses the direction of the next step when an iterate ends on one of the other data points. We will first show that with probability one, such a point will never be a fixed point of the algorithm. For simplicity, we consider this for the standard MDS cost function without the penalty term. The updating step is then

$$\mathbf{x}_i^t = \frac{1}{N} \sum_j \hat{\mathbf{x}}_j. \quad (15)$$

Now we assume, without loss of generality, that the current iterate $\mathbf{x}_i^{t-1} = \mathbf{x}_N$. Using the definition of the auxiliary points, we have:

$$\mathbf{x}_i^t = \frac{1}{N} \left(\sum_j \mathbf{x}_j + \sum_{j=1}^{N-1} d_{ij} \frac{\mathbf{x}_N - \mathbf{x}_j}{\|\mathbf{x}_N - \mathbf{x}_j\|} + d_{iN} \mathbf{u} \right). \quad (16)$$

If this point was a fixed point of the algorithm, i.e. $\mathbf{x}_i^t = \mathbf{x}_i^{t-1}$, there would be only one single direction \mathbf{u} which fulfills the resulting linear equation (16). But this event has probability zero when \mathbf{u} is chosen randomly.

We thus know that for any fixed point, with probability one, $u(\cdot, z)$ and $f(\cdot)$ are differentiable at $x = z$. Hence, with probability one, for any fixed point z of the algorithm the following condition on the derivative holds:

$$\vec{\nabla} u(x, z) \Big|_{x=z} = \vec{\nabla} f(z). \quad (17)$$

Theorem 1. *Every limit point of the iterates generated by (13) is first-order optimal.*

Proof. Let us assume that a subsequence x^{t_j} exists which converges to a limit point z .

$$u(x^{t_{j+1}}, x^{t_{j+1}}) = f(x^{t_{j+1}}) \leq u(x^{t_{j+1}}, x^{t_j}) \leq u(x, x^{t_j}) \quad \forall x.$$

The equality follows from the fact that $u(y, y) = f(y)$. The first bound follows from $u(x, y) \geq f(x)$ and the last inequality is due to the optimality of $x^{t_{j+1}}$. We now perform the limit $j \rightarrow \infty$ and arrive at

$$u(z, z) \leq u(x, z) \quad \forall x.$$

Thus $u(\cdot, z)$ has a local minimum at $x = z$. This implies that

$$\vec{\nabla} u(x, z) \Big|_{x=z} = 0,$$

since $u(\cdot, z)$ is a convex differentiable function. Combining this with (17) we obtain

$$\vec{\nabla} f(z) = 0.$$

□

2.3.2. Convergence of the outer loop

The outer loop has the structure of a (block)-coordinate descent algorithm (see [54], for a review). Surprisingly, there are few results in the literature that could be used to guarantee convergence of coordinate-descent on a non-differentiable, non-convex objective function. There are some examples to the contrary: Powell [55] gives an example of a non-convex differentiable objective that leads coordinate descent to loop forever between suboptimal points.

In our particular case, however, we have established that points of non-differentiability cannot be fixed points of the inner loop, and that the set of fixed points of the inner loop are first-order optimal (meaning the gradient of the cost with respect to curve x_i is 0). A global fixed point would therefore imply that all conditional gradients are null, meaning that the fixed point is first-order optimal. Further, since the algorithm produces a strictly non-increasing sequence of cost values, it can only converge to a local minimum or saddle-point (if it converges at all).

This proof does not exclude the possibility of a limit cycle, meaning that there could be several limit points with the same cost which the algorithm visits in turns. However, in our implementation, we determine convergence based on the stationarity of the configuration and not of the cost. Thus, if the inner loop converges, it converges to a unique configuration which is a stationary point. If the algorithm did indeed jump between

several limit points, it would not converge and stop after a certain number of iterations. However, this has never happened in our experience.

Beyond the fact that the algorithm does converge to an appropriate point, it would be interesting to prove some results on the speed at which it converges. Unfortunately, local convergence is very difficult to study in our case since most techniques rely on the assumption that the cost function becomes quadratic in a neighbourhood of an optimum. Due to the invariances inherent in the MDS cost function, it is not clear at all that a local quadratic model is in any way appropriate.

2.4. cMDS with more than one hyperparameter

In certain cases it might be interesting to look at the effects of varying more than one hyperparameter. An extension of cMDS is thus the use of two or more hyperparameters, α and β . This allows to consider, for example, time plus an additional hyperparameter such as scaling or weighting. Having multiple hyperparameters effectively only changes the penalty matrix used in the spline regressions: instead of a penalty matrix appropriate for parametric curves in $\mathbb{R} \rightarrow \mathbb{R}^d$, we need a penalty appropriate for curves in $\mathbb{R}^m \rightarrow \mathbb{R}^d$, where m is the number of parameters.

As an example, take the case where we have two hyperparameters, $m = 2$. Suppose we have a grid $\alpha_1 \dots \alpha_{T_\alpha} \times \beta_1 \dots \beta_{T_\beta}$. Then, our configuration matrix is $\mathbf{X}^{(\cdot)(\cdot)} \in \mathbb{R}^{T_\alpha \cdot T_\beta \times d \times N}$. If $T_\alpha \neq T_\beta$, we have two matrices for the discrete second order differential operator, \mathbf{M}_α and \mathbf{M}_β , of appropriate dimensionality. Thus, we get the following penalty matrix for two hyperparameters:

$$\mathbf{M} = \mathbf{M}_\beta \otimes \mathbf{E}_{T_\alpha} + \mathbf{E}_{T_\beta} \otimes \mathbf{M}_\alpha, \quad (18)$$

defining a separable penalty across the two hyperparameters. For more on penalty matrices, see for example [45].

The results can be visualised by selecting certain slices of the grid. In the case of 1-dimensional embeddings, selecting a certain value for one hyperparameter leads to the visualization of curves that depend on the other hyperparameter. In case of 2-dimensional embeddings, one can select a certain value of one hyperparameter and then analyze the corresponding, possibly animated, scatterplot that varies with the second one.

2.5. Initialization

The cMDS algorithm needs to be seeded with a starting configuration. The optimization works with a random initialization, but performance can be improved with a more structured approach.

One possibility is to perform classical scaling [9, 6] or other standard MDS methods such as SMACOF [20, 56] for each value of α . However, the separate solutions might be difficult for cMDS to "glue" together and thus form a poor initialization. A more robust variant is initialization with an aggregated solution. To obtain this solution we average each curve $\mathbf{x}_i^{(\cdot)}$ over all values of α and then perform classical scaling on $\bar{\mathbf{X}}$. If some patterns are very strong for only a certain range of α , they might influence the entire embedding via the aggregated initialization. Thus, one should consider which kind of initialization is suitable for the data at hand.

2.6. Embedding dimension

We would like to shortly comment on the choice of the embedding dimension. Since we aim for visualization, one only needs to choose between $d = 1$ or $d = 2$. One way to choose is to look at the so-called Shepard plots. In such a plot, embedding distances are plotted against original distances. For an ideal embedding, all the points in such a plot would fall on the diagonal. Thus, the spread from the diagonal can be used to judge whether the embedding is meaningful or not. If the Shepard plot looks reasonable for $d = 1$, one should choose this dimension. If there is a significant improvement for $d = 2$, one should choose the higher dimension.

2.7. Variants of cMDS

There exists multiple variants of the standard MDS problem. All of them can be implemented in cMDS. Most variants are defined over weights $\mathbf{W}^{(\cdot)}$ in the cost function:

$$\begin{aligned} \mathcal{C}(\mathbf{X}^{(\cdot)}, \mathbf{D}^{(\cdot)}) &= \sum_{k=1}^T \|\mathbf{W}^k \circ (\tilde{\mathbf{D}}^k - \mathbf{D}^k)\|_F^2 \\ &\quad + \lambda \sum_i (\mathbf{x}_i^{(\cdot)})^T (\mathcal{D}^{(2)})^T \mathcal{D}^{(2)} \mathbf{x}_i^{(\cdot)}. \end{aligned}$$

Sammon's mapping [14] can be implemented by setting $w_{ij}^k = (d_{ij}^k)^{-1}$. In *elastic stress*, on the other hand, we have weights $w_{ij}^k = (d_{ij}^k)^{-2}$ [15]. This is equivalent to a Kamada-Kawai layout in graph visualization [23]. *Multidimensional unfolding* is useful when the data separates into groups [16]. Then, the weights corresponding to between-group distances are set to zero. Local MDS (LMDS) [17] is a slightly more complex variant. Here, the weights are set depending on local neighborhoods: If object \mathbf{s}_j^k is a k -nearest neighbor of \mathbf{s}_i^k (which we denote as $j \in \mathcal{N}_i$), then the corresponding term in the cost gets weight $w_{ij} = 1$ and the original distance d_{ij} is used. In all other cases the weight w is set to a small value (not dependent on i, j) and $d_{ij} = D_\infty$, where D_∞ is a large constant. This leads to a focus on local neighborhood structure and adds a repulsive force to avoid a 'crumbling together' of the embedding. The corresponding cMDS cost function is

$$\begin{aligned} \mathcal{C}(\mathbf{X}^{(\cdot)}, \mathbf{D}^{(\cdot)}) &= \sum_{k=1}^T \sum_{i=1}^N \sum_{j \in \mathcal{N}_i} \left(\|\mathbf{x}_i^{(\cdot)} - \mathbf{x}_j^{(\cdot)}\|_2 - d_{ij}^k \right)^2 \\ &\quad + \sum_{k=1}^T \sum_{i=1}^N \sum_{j \notin \mathcal{N}_i} w^k \left(\|\mathbf{x}_i^{(\cdot)} - \mathbf{x}_j^{(\cdot)}\|_2 - D_\infty^k \right)^2 \\ &\quad + \lambda \sum_i (\mathbf{x}_i^{(\cdot)})^T (\mathcal{D}^{(2)})^T \mathcal{D}^{(2)} \mathbf{x}_i^{(\cdot)}. \end{aligned}$$

LMDS introduces an additional hyperparameter via w , that needs to be optimized for the data at hand.

It is also possible to use ISOMAP [18] to visualize changes in manifold structure that are induced by a change in metric. To this end, one constructs geodesic distances based on different distances $d_\alpha(x, y)$ and subsequently applies cMDS.

2.8. R package **cmds**

To make cMDS accessible we provide an R package, **cmds**, on github ([here](#)). The heart of the package is the function **cmds**. Functions for plotting and summarizing results are also available. To show the usability of cMDS we display exemplary function calls here:

```
ComputeCmds(kDistances, kDim = 2, kLambda = 5, kWeights = "sammon")
ComputeCmds(kDistances, kDim = 1, kLambda = 3, kInit = "smacof")
```

where DL is a list of distance matrices of length T , k is the embedding dimension and λ is λ , the regularization parameter. W can be set to use different variants of MDS, such as *Sammon's mapping*, *Kamada-Kawai* and *unfolding*. Different initializations are also available. There is also a built-in plotting function. For example, the Shepard plot can be plotted like this:

```
PlotCmds(ComputeCmds(kDistances), kShepard = TRUE)
```

Further examples can be found in the package documentation.

2.9. Distance families

We would like to discuss some general properties of distance families.

Weighted distances. A very common case is that data can be described using different features or sets of features. In that case, we can build two distance matrices \mathbf{D}_1 and \mathbf{D}_2 on one feature respectively. Then, we can construct a weighted metric using a convex combination of the two. Using a convex combination ensures that each resulting matrix is again a distance matrix.

$$\mathbf{D}^{(\cdot)} = \sqrt{\alpha \mathbf{D}_1^2 + (1 - \alpha) \mathbf{D}_2^2}. \quad (19)$$

We present examples of weighted distance in Section 3.

Changing inherent dimensionality. An interesting issue is a change in intrinsic dimensionality in the distance function. In classical scaling, low dimensional embeddings are projections of higher dimensional embeddings. For example, a 2d embedding is the projection of the 3d embedding. In reverse, this means that embedding high dimensional data in lower dimensions leads to larger distortion. In continuous embeddings, the dimensions are not stacked, as in the classical case. However, it is still true that a larger difference in dimensionality between original and embedded data leads to larger distortion. We illustrate this by mixing a distance matrix based on low dimensional data ($d = 2$) with one based on high dimensional data ($d = 12$). We embed this data in $d = 2$. Plotting the distortion for each value of α , as defined in (2), shows that, as expected, embedding the 2d data in $d = 2$ yields zero distortion. The distortion increases as the high dimensional data gets more weight in the mixture.

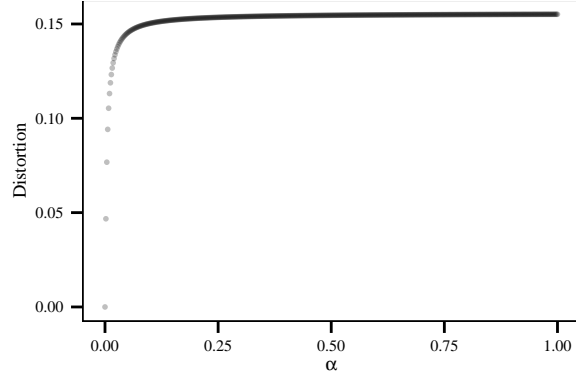


Figure 3: Change of inherent dimensionality: $\alpha = 0$ represents a distance matrix based on two dimensional data, $\alpha = 1$ is based on 12-dimensional data. The mixture was embedded in $d = 2$. It is clearly visible that, as the influence of the high-dimensional data increases, the distortion also increases.

3. Examples

In the following examples we show how cMDS can be used to analyze the effect of hyperparameters of distance functions on the data. The first example is a toy example, where we illustrate how cMDS can visualize cluster structure in data and how it compares to a more basic method based on k -means. In the second example, we use a weighted metric: distances evolve according to the relative weight given to some of the dimensions. In the third example, we work with brain connectivity data. We vary the distance function between networks according to different thresholding rules. In the fourth example, we derive a multiscale representation of data using hierarchical clustering and visualize changes in distance over scale.

3.1. Comparison of cMDS to clustering

We compare cMDS to a rudimentary method of tracking the influence of changing distances on the data structure.

An approach based on traditional methods is to perform clustering on the original data for multiple instances of the distance family $d_\alpha(x, y)$. To track whether clusters in the data emerge independently of the distance measure, we use the set of cluster indices c_i of a specific level of α for all levels of α and compute the corresponding quality of the clustering result, for example, by comparing within and between cluster variance. By good clustering quality, we mean data that has well separated clusters. Low quality on the other hand refers to strongly overlapping clusters. If the quality breaks down, this should be visible in the cMDS result on first glance.

We demonstrate this with an artificial example. Suppose we draw five random cluster centers in \mathbb{R}^5 and then let those cluster centers linearly collapse to zero, such that for $\alpha = 0$ we have well separated cluster centers and at $\alpha = 1$ all cluster centers are at the origin. In our example, we draw the cluster centers uniformly from a 5-dimensional hypercube with coordinate-wise limits of $[-10, 10]$ and sample 15 points from a linear

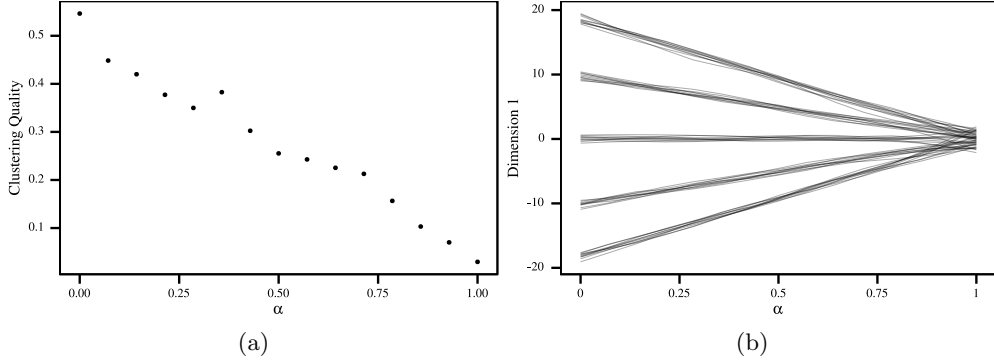


Figure 4: (a) Clustering quality based on k -means clustering. We performed clustering at $\alpha = 0$ and used the resulting indices at all levels of α . One can see that the quality decreases, thus indicating that the cluster structure that is present in the beginning is not invariant. However, one cannot know what happens with the data exactly. (b) cMDS on the toy example. $\alpha = 0$ represents the distance matrix based on data in five clusters in \mathbb{R}^5 . $\alpha = 1$ represents a random distance matrix. The results provide at a glance that clusters dissolve at larger values of α . This information is difficult to obtain based on k -means clustering alone.

approximation between the centers and the origin. We then simulate Gaussian data at all values of α , using the respective cluster centers and simulating 10 points per cluster. We use a variance of 0.3 to ensure that the clusters are well separated in the beginning. We then compute the corresponding distance matrices for each level of α . Now, let us assume we don't know anything about the data. We compute a k -means clustering for $\alpha = 0$. We use the resulting cluster indices for all levels of α and compute cluster quality. We use the following quality measure:

$$q = \left(\sum_i \frac{d_{max}(x_{c_i})}{d_{min}(c_i)} \right)^{-1}$$

i.e. for each cluster, we compute the maximum distance between two points in this cluster and the minimum distance from this cluster's center to another cluster center. The inverse of the sum of those values gives our quality measure. Low quality (strong overlap) of the clustering leads to low values of q . We run k -means and compute q ten times to get an average measure for cluster quality. The results are shown in Fig. 4a. We see that the quality of the clustering decreases with α . From this we can conclude only that the clustering in the beginning is not present at the end. What we do not know is what the data actually looks like. Are there no clusters in the end or maybe different clusters than in the beginning? Let us now look at the cMDS embedding of the distance matrices (Fig. 4b). Here we see that the clustering structure slowly degenerates with increasing α and that no other clusters arise for large α . Thus, we gain a lot more insight about the data without having to run k -means many times and computing average cluster quality measures.

3.2. Economic and demographic descriptors of EU countries

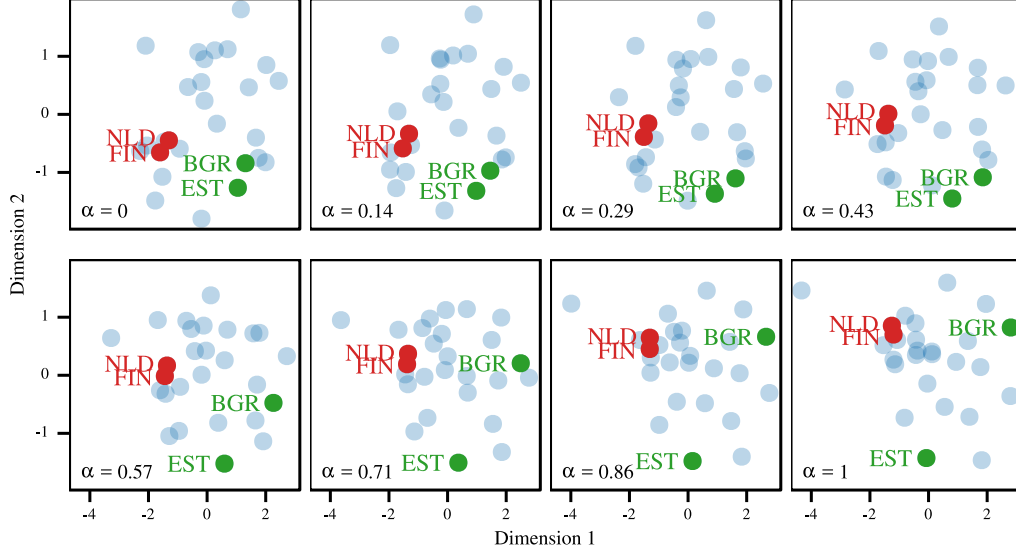


Figure 5: Effects of changes in a weighted metric. Economic and demographic distance measures are weighted according to $\mathbf{D}(\alpha) = \sqrt{\alpha \cdot \mathbf{D}_E^2 + (1 - \alpha) \cdot \mathbf{D}_D^2}$. Thus, the first panel represents an embedding that is solely based on the demographic metric, while the last one is based on the economic one. In between are samples of different weights, denoted by α . The red labels depict the neighborhood relation between the Netherlands (NLD) and Finland (FIN) which is invariant with respect to α , while the green ones, Bulgaria (BGR) and Estonia (EST), are an example of a strong dependency on α .

For this example we use economic and demographic descriptors of the 27 EU countries. The data are publicly available from the **gapminder** website. As economic variables we use income per capita (2008), CO₂ emissions per capita (2008) and number of granted patents per capita (2002). Demographic variables are total fertility rate, life expectancy at birth and the fraction of urban population. We scale all variables logarithmically. We then build two distance matrices, one solely based on economic variables, \mathbf{D}_E , the other on demographic variables, \mathbf{D}_D . Now, we put different weights on both variable groups and have a continuous range with one extreme only considering demographic variables and the other extreme only considering economic ones:

$$\mathbf{D}(\alpha) = \sqrt{\alpha \cdot \mathbf{D}_E^2 + (1 - \alpha) \cdot \mathbf{D}_D^2}$$

where α is between 0 and 1. We sample this continuum at N different values of α and thus get a distance array in $R^{N \times 27 \times 27}$. For this example, a 1D embedding is not sufficient to capture relevant trends in the data. Thus, we give snapshots of the 2D results in Fig. 5. In this case, an interactive presentation of the results is much easier to read and thus an advantageous choice. An interactive visualization is viewable at <http://tinyurl.com/cMDS-demo>. We also implemented an interactive web application

with the R package **shiny**, which is online at <http://ginagruenhage.shinyapps.io/EU-App>. The 2D embedding shows that the changes in weighting have significantly different effects on the individual neighborhood relations. For example, some demographically very similar countries start diverging when economic variables are taken into account and end up far apart under the economic distance metric (e.g. Bulgaria and Estonia). Other countries stay similar, independent of the weighting of different distances (e.g. Finland and the Netherlands). These patterns are lost when deciding a distance measure a priori. Using cMDS to visualize the effects of the hyperparameter makes them easier to discover and understand. In the web application, one can also toggle quantitative analyses of the cMDS output, such as a vector graphic showing local stability of various countries. It is also possible to color countries according to their respective penalty, to judge overall stability.

3.3. Diffusion Tensor Imaging

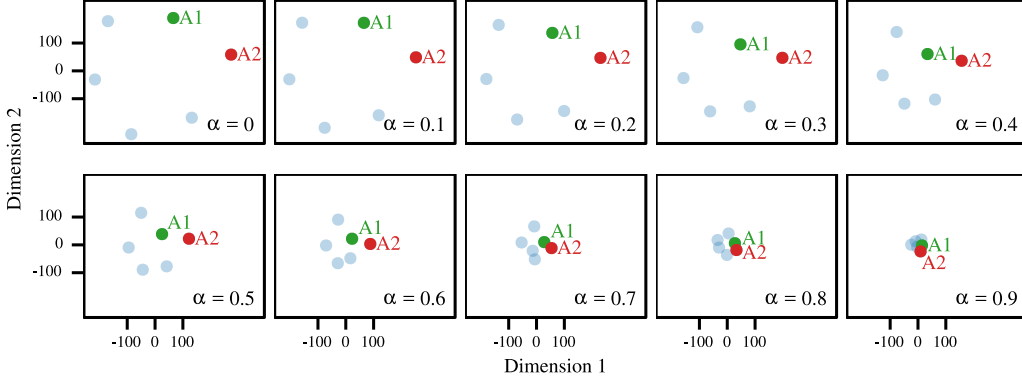


Figure 6: Embedding of thresholded networks acquired by DTI and tractography [57]. The hyperparameter α corresponds to the quantile of weakest connections that is ignored for inter-subject comparisons. Measurements are compared based on the resulting binary connectivity matrix using the Hamming distance between graphs. The analysis shows that the two measurements for subject A, A1 and A2, differ about as much as other pairwise comparisons. This is interesting because it highlights the relatively low reliability of DTI tractography.

Brain regions are linked by white matter tracts, forming a network called the Connectome [58]. Diffusion Tensor Imaging (DTI) is a form of magnetic resonance imaging that can be used to find connections between brain regions (using tractography, [59]). Here we use data obtained by Hagmann et al. [57], available here. DTI produces noisy results and it is difficult to compare individual subjects. Consequently, connectivity must often be averaged over individual subjects. There is by necessity some arbitrariness in the averaging and we show here how cMDS can be used to visually compare different subjects and to visualize changes in network structure as the averaging criterion is varied. In the original data, the brain is segmented into 998 regions of interests that cover about 1.5 cm^2 each and belong to one of 66 anatomical regions (33 per hemisphere). Here, we

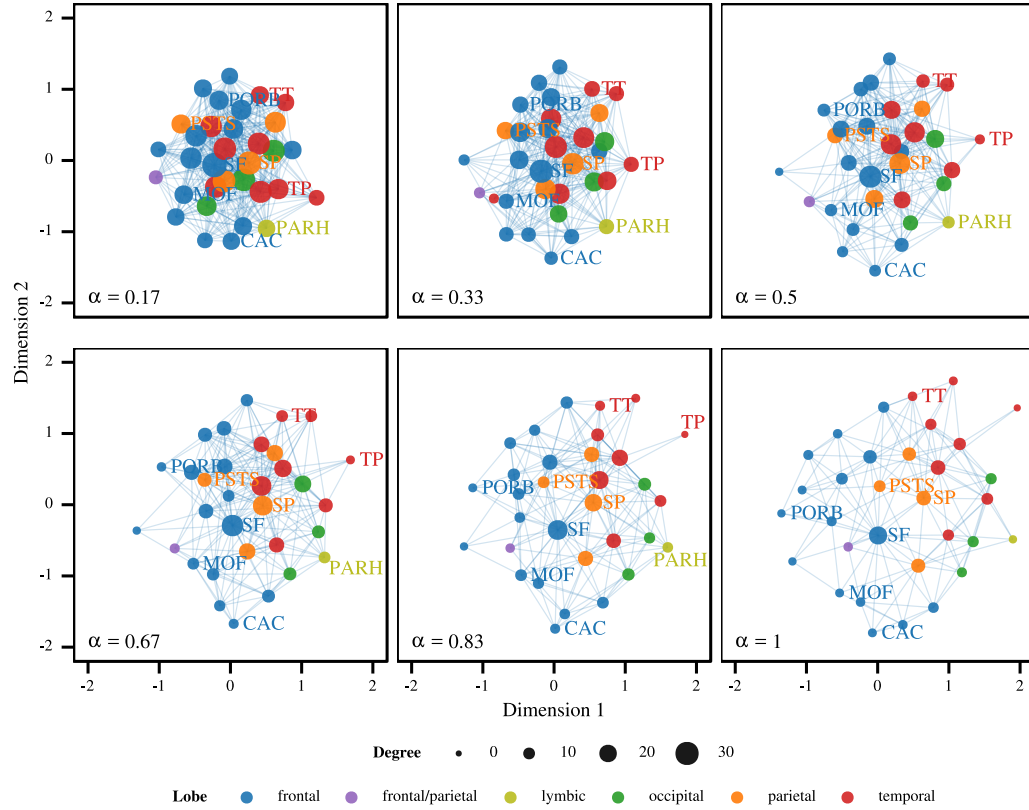


Figure 7: Embedding of regional networks acquired by DTI and tractography [57]. The different panels correspond to different threshold levels α when building consensus networks. The superior frontal cortex (SF) and superior parietal cortex (SP) are stable regions in the core of the network, while the parahippocampal cortex (PARH), the caudal anterior cingulate (CAC), and the transverse temporal cortex (TT) are stable peripheral regions. Some regions are less stable, e.g. the pars orbitalis (PORB), the temporal pole (TP), the medial orbitofrontal cortex (MOF) and the postcentral gyrus (PSTS).

do not work with the full network, but rather with regionally aggregated data for the left hemisphere. That is, the data are adjacency matrices in $\mathbb{R}^{33 \times 33}$, where $A_{ij}^{(s)}$ is the connection strength between regions i and j for subject s . Connections are measured for five different subjects, with two separate measurements for subject A. To compare different subjects and to evaluate whether strong connections are more stable across subjects than weak ones, we perform a thresholding analysis. Our hyperparameter is the quantile of weak connections that we ignore in the comparisons. Thus, $\alpha = 0.1$ corresponds to setting the weakest 10% of connections to zero. We then binarize the adjacency matrix, $B^{(s)} > A^{(s)}$, such that we have $B_{ij}^{(s)} \in \{0, 1\}$ and compute the Hamming distance between the graphs corresponding to different measurements. We use embeddings in \mathbb{R}^2 to represent the data (Fig. 6). Surprisingly, the approximate distances between the two measurements of subject A, A1 and A2, are of the same order of magnitude compared to the distances between distinct subjects. This is true across thresholding levels. This is a curious finding, since these two measurements are averaged in Hagmann et al. [57] before inter-subject averaging is performed. Our findings suggest that these two measurements are not easily distinguishable from other pair comparisons. According to Hagmann et al. [57], the correlation between the regional connectivity of A1 and A2 is $r^2 = 0.78$, compared to $r^2 = 0.65$ between distinct subjects. The cMDS visualization suggests that A1 and A2 are not notably more similar than other pairwise comparisons.

Because of the relatively high noise in the data some form of averaging (over subjects) may be useful to perform structural analyses on the regional network. Due to our findings in the network thresholding analysis we treat measurements A1 and A2 separately. One approach to perform averaging is to produce consensus networks out of the individual networks, with the rule that a link is introduced in the 50% consensus network if and only if it is present in at least 50% of the individual networks. Thus, we look at the average adjacency matrix \bar{B} , where $\bar{B}_{ij} = (1/N) \sum_s B_{ij}^{(s)}$ and $B^{(s)} = A^{(s)} > 0$. We then threshold it at different levels to produce different consensus networks. We use the threshold level α as the continuous parameter in cMDS, such that $\bar{B}_{ij}(\alpha) = \bar{B}_{ij} > \alpha$. If we take the shortest-path distance on the graph that is defined by $\bar{A}(\alpha)$ as the distance measure between two regions, then changing the threshold level is exactly the same as changing the distance measure, as some links will start disappearing with higher threshold values. We can therefore apply cMDS to visualize the changes in network structure as the averaging rule is changed. For this example, we implement a weighted version of the algorithm, to mirror the standard Kamada-Kawai layout methods [23]. The results are shown in Fig. 7 for regions in the left hemisphere. We also present these results interactively with a web application using the R package **shiny** which is viewable at <http://ginagruenhage.shinyapps.io/DTI-App>. A first (and unsurprising) result is that the network density decreases significantly with the threshold level. That is, as we start requiring higher levels of consistency among the subjects, a lot of connections are rejected. We would like to note, that in this case, integer distances are embedded in a continuous space. However, aspects such centrality are recovered in the visualization: regions with dense connections and high values of centrality and betweenness are placed at the center of the configuration.

Results show that some regions are very stable: for example, the superior frontal cortex (SF) and the superior parietal cortex remain at the core of the network, while the parahippocampal cortex (PARH), the caudal anterior cingulate (CAC) and the temporal

cortex (TT) are examples of stable regions at the periphery of the network. What is even more interesting is that some regions are rather unstable and change their role in the network. The postcentral gyrus (PSTS) starts out in the periphery and then moves to the center. Other regions move from the core to the periphery, e.g. the pars orbitalis (PORB), the temporal pole (TP) and the medial orbitofrontal cortex (MOF). Since core-periphery relationships are central to the interpretation of connectome data, it is crucial to know which regions can be reliably called peripheral and others central [57]. cMDS provides this information at a glance.

3.4. Hierarchical clustering

We mentioned in the introduction that distance is often computed relative to a certain scale. For spatial or temporal data, scale corresponds to a concrete spatial or temporal window, but there are other ways to obtain a multiscale representation. Hierarchical clustering [60, 61] is such a technique. We focus here on agglomerative clustering, where the algorithm starts with each observation in one cluster. At each level, the algorithm merges the two closest clusters until only one cluster remains. Thus, there are $N - 1$ levels in the hierarchy, which gives a view of the data going from the roughest to the most detailed level. cMDS provides an interesting visualization of the results.

We first define a distance function at each level of the hierarchy. For each level, we build the distance matrix for the N datapoints as follows: if two datapoints are in the same cluster we assign a very small positive distance. Specifically, we used the absolute value of samples from a Gaussian with zero mean and standard deviation of 0.005. If two datapoints are not in the same cluster we compute the euclidean distance between the centers of their respective clusters. Here, we use the publicly available **USArrests** dataset from the R **datasets** package. It contains data on murder arrests, assault arrests, rape arrests and urban population for the different US states in 1973. We picked this dataset because it is used as an example for the R **hclust** function. We use cMDS to embed these data (Fig. 8). The result is a tree structure with the property that, at each level of the tree, distances between branches are representative of distances between clusters. This enables an immediate understanding of the hierarchical clustering results. We also developed an interactive visualization based on a 2D embedding, which better captures the potential of cMDS for such applications. We invite readers to have a look at these results (online at <http://tinyurl.com/cMDS-demo>).

4. Discussion

We introduced an easy to implement and flexible version of continuous (or dynamic) multi-dimensional scaling, namely cMDS. We showed that cMDS provides a fast and informative way of understanding the data structure by presenting four examples. In a toy example, we compared the results to a method based on k -means which gave only an approximate idea of what was going on in the data, while cMDS yielded a very concise representation of the data. With the second example on EU data we showed the effects of changing a weighted metric, putting different weights on two feature classes. Visualization revealed countries whose neighborhood relations are invariant to the change in weights as well as countries whose relative position strongly depends on the weights. In a brain connectivity example, where averaging over subjects is not straightforward,

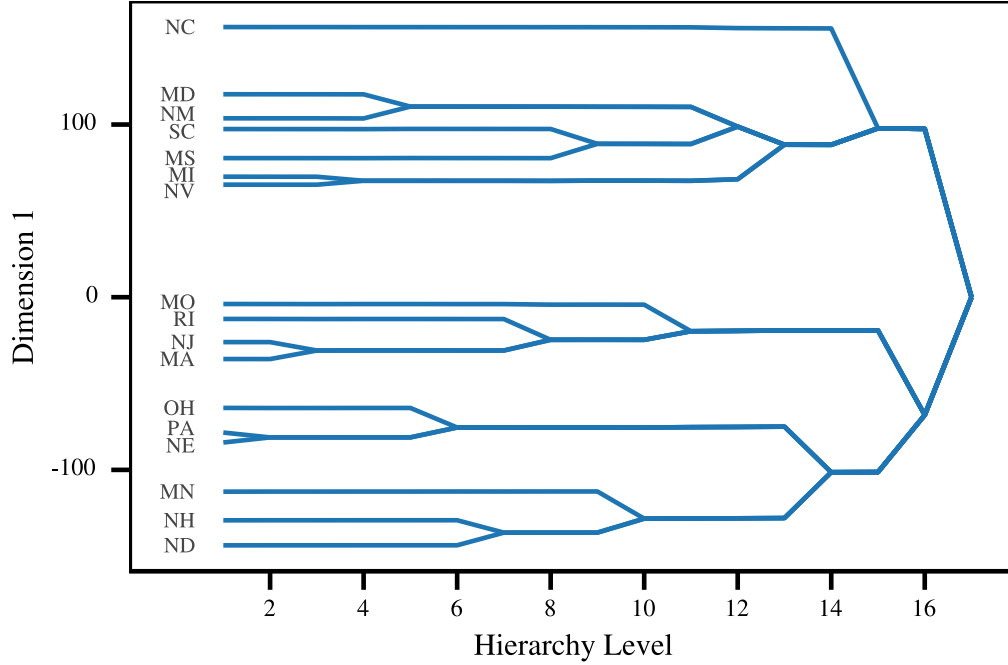


Figure 8: Embedding of an hierarchical clustering results for the **USArrests** data set which is publicly available as part of the R **datasets** package. We selected a sample of 17 states. We ran hierarchical clustering using the centroid of clusters for agglomeration. Then, we built the distance matrix for each level: two datapoints are assigned a small positive distance if they are in the same cluster and the distance between cluster centers otherwise. The cMDS result is a visualization of the tree structure. At each level, the distances between branches represents the distances between cluster centers.

we found a way to compare different subjects giving more and more weight to strong connections. We found that cMDS suggests two measurements of the same subjects to be as different as measurements of two different subjects. In a second analysis we found that the shape of the network changes according to the averaging rule. With cMDS, identifying stable and unstable regions turned out to be straightforward. Finally, we showed how hierarchical clustering can be thought of as a multiscale representation of data, and we used cMDS to visualize how the structure of the data changes across scale.

We only considered small datasets here, for which cMDS is very fast (with a runtime of a few seconds at most). Like other MDS methods, it doesn't scale well with n . It hasn't been proved to be better than $\mathcal{O}(n^3)$, though it might be, which depends on how many runs of the outer loop are needed. Practically, the runtime remains reasonable with a few hundred datapoints, and straightforward extensions for larger datasets are possible. One idea is to embed a subset of landmark points, as in landmark MDS [62]. Another is to use sparse weighting matrices, tying each datapoint to a random subset of neighbours.

By visual inspection, cMDS immediately reveals qualitative structures in the neigh-

borhood dynamics for various datasets. Furthermore, quantitative analyses are possible. For example, performing clustering on cMDS output can yield results that are robust to changes in the distance measure. We leave these extensions to future work.

5. Conclusion

With cMDS, we address a fundamental problem in pattern recognition and machine learning: the initial choice of a distance metric. This is a hidden assumption in various methods. We argue that this choice should be addressed explicitly. Our suggestion is to use continuous MDS techniques, which visualize the dynamics that (so far) arbitrary choices in distance functions introduce in data. cMDS can deal with numerous sources of arbitrariness in the distance metric, examples of which are varying scale or weighting. We show that interesting and important dynamics, such as invariance and declustering, are readily revealed by cMDS. Finally, we provide a cMDS algorithm that is straightforward to implement, use and extend.

Acknowledgments

This work was partially supported by the Deutsche Forschungsgemeinschaft (GRK1589/1).

References

References

1. Carlsson G. Topology and data. *Bulletin of the American Mathematical Society* 2009;46(2):255–308.
2. Schölkopf B, Smola AJ. Learning with kernels: support vector machines, regularization, optimization and beyond. the MIT Press; 2002.
3. Lum PY, Singh G, Lehman A, Ishkanov T, Vejdemo-Johansson M, Alagappan M, Carlsson J, Carlsson G. Extracting insights from the shape of complex data using topology. *Scientific reports* 2013;3.
4. Buja A, Swayne DF. Visualization methodology for multidimensional scaling. *Journal of Classification* 2002;19(1):7–43.
5. Buja A, Swayne DF, Littman ML, Dean N, Hofmann H, Chen L. Data Visualization With Multidimensional Scaling. *Journal of Computational and Graphical Statistics* 2008;17(2):444–72.
6. Torgerson W. Multidimensional scaling: I. Theory and method. *Psychometrika* 1952;17(4):401–19.
7. Torgerson WS. Theory and methods of scaling. Wiley; 1958.
8. Gower JC. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika* 1966;53(3-4):325–38.
9. Kruskal JB. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 1964;29(1):1–27.
10. Shepard RN. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika* 1962;27(2):125–40.
11. Ramsay JO. Maximum likelihood estimation in multidimensional scaling. *Psychometrika* 1977;42(2):241–66.
12. Ramsay JO. Confidence regions for multidimensional scaling analysis. *Psychometrika* 1978;43(2):145–60.
13. Ramsay JO. Multiscale: Four Programs for Multidimensional Scaling by the Method of Maximum Likelihood.[user's Guide]. National Educational Resources; 1978.
14. Sammon JW. A nonlinear mapping for data structure analysis. *Computers, IEEE Transactions on* 1969;100(5):401–9.
15. McGee VE. The multidimensional analysis of 'elastic' distances. *British Journal of Mathematical and Statistical Psychology* 1966;19(2):181–96.
16. Borg I, Groenen PJF. Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics). Second ed.; Springer; 2005.
17. Chen L, Buja A. Local multidimensional scaling for nonlinear dimension reduction, graph drawing, and proximity analysis. *Journal of the American Statistical Association* 2009;104(485):209–19.
18. Tenenbaum JB, de Silva V, Langford JC. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 2000;290(5500).
19. Guttman L. A general nonmetric technique for finding the smallest coordinate space for a configuration of points. *Psychometrika* 1968;33(4):469–506.
20. De Leeuw J. Applications of Convex Analysis to Multidimensional Scaling. In: Barra JR, Brodeau F, Romier G, van Cutsem B, eds. *Recent developments in statistics*. Amsterdam, The Netherlands: North-Holland; 1977:133–45.
21. De Leeuw J, Heiser WJ. Convergence of correction matrix algorithms for multidimensional scaling. *Geometric representations of relational data* 1977;:735–52.
22. De Leeuw J. Convergence of the majorization method for multidimensional scaling. *Journal of Classification* 1988;5(2):163–80.
23. Kamada T, Kawai S. An algorithm for drawing general undirected graphs. *Information processing letters* 1989;31(1):7–15.
24. Gansner E, Koren Y, North S. Graph Drawing by Stress Majorization. In: Pach J, ed. *Graph Drawing*; vol. 3383 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2005:239–50.
25. Misue K, Eades P, Lai W, Sugiyama K. Layout Adjustment and the Mental Map. *Journal of Visual Languages & Computing* 1995;6(2):183–210.
26. Böhringer KF, Paulisch FN. Using constraints to achieve stability in automatic graph layout algorithms. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '90; New York, NY, USA: ACM; 1990:43–51.

27. North S. Incremental layout in DynaDAG. In: Brandenburg F, ed. *Graph Drawing*; vol. 1027 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 1996:409–18.
28. Brandes U, Wagner D. A bayesian paradigm for dynamic graph layout. In: DiBattista G, ed. *Graph Drawing*; vol. 1353 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 1997:236–47.
29. Brandes U, Corman SR. Visual Unrolling of Network Evolution and the Analysis of Dynamic Discourse†. *Information Visualization* 2003;2(1):40–50.
30. Moody J, McFarland D, BenderdeMoll S. Dynamic Network Visualization. *American Journal of Sociology* 2005;110(4):1206–41.
31. Diehl S, Görg C. Graphs, They Are Changing. In: Goodrich M, Kobourov S, eds. *Graph Drawing*; vol. 2528 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2002:23–31.
32. Frishman Y, Tal A. Online Dynamic Graph Drawing. *Visualization and Computer Graphics, IEEE Transactions on* 2008;14(4):727–40.
33. Erten C, Harding PJ, Kobourov SG, Wampler K, Yee G. Exploring the computing literature using temporal graph visualization. In: *Visualization and Data Analysis 2004. Edited by Erbacher, Robert F.; Chen, Philip C.; Roberts, Jonathan C.; Gröhn, Matti T.; Börner, Katy. Proceedings of the SPIE, Volume 5295, pp. 45-56 (2004).*; vol. 5295. 2004:45–56.
34. Erten C, Kobourov S, Le V, Navabi A. Simultaneous Graph Drawing: Layout Algorithms and Visualization Schemes. In: Liotta G, ed. *Graph Drawing*; vol. 2912 of *Lecture Notes in Computer Science*; chap. 41. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004:437–49.
35. Dwyer T, Hong SH, Koschützki D, Schreiber F, Xu K. Visual analysis of network centralities. In: *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60*. APVis '06; Darlinghurst, Australia, Australia: Australian Computer Society, Inc.; 2006:189–97.
36. Baur M, Schank T. Dynamic graph drawing in visone. Citeseer; 2008.
37. Brandes U, Mader M. A Quantitative Comparison of Stress-Minimization Approaches for Offline Dynamic Graph Drawing. In: Kreveld M, Speckmann B, eds. *Graph Drawing*; vol. 7034 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2012:99–110.
38. Xu K, Kliger M, Hero A. A regularized graph layout framework for dynamic network visualization. *Data Mining and Knowledge Discovery* 2013;27(1):84–116.
39. Sarkar P, Moore AW. Dynamic social network analysis using latent space models. *SIGKDD Explor Newsl* 2005;7(2):31–40.
40. Mizuta M. Multidimensional scaling for dissimilarity functions with continuous argument(functional data analysis). *Journal of the Japanese Society of Computational Statistics* 2003;15(2):327–33.
41. Yuille AL, Rangarajan A. The Concave-convex Procedure. *Neural Comput* 2003;15(4):915–36.
42. Lanckriet GR, Sriperumbudur BK. On the Convergence of the Concave-Convex Procedure. In: Bengio Y, Schuurmans D, Lafferty JD, Williams CKI, Culotta A, eds. *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc.; 2009:1759–67.
43. Yen IE, Peng N, Wang P, Lin S. On convergence rate of concave-convex procedure. In: *Proceedings of the NIPS 2012 Optimization Workshop*. 2012:.
44. Cook D, Swayne DF. Interactive and dynamic graphics for data analysis: with R and GGobi. Springer; 2007.
45. Ramsay JO, Silverman BW. Functional Data Analysis. New York: Springer Series in Statistics; 1997.
46. Kaski S, Peltonen J. Dimensionality Reduction for Data Visualization [Applications Corner]. *Signal Processing Magazine, IEEE* 2011;28(2):100–4.
47. Mokbel B, Lueks W, Gisbrecht A, Hammer B. Visualizing the quality of dimensionality reduction. *Neurocomputing* 2013;112:109–23.
48. Rockafellar RT. Convex analysis (princeton mathematical series). *Princeton University Press* 1970;46:49.
49. Agarwal A, Phillips JM, Venkatasubramanian S. Universal multi-dimensional scaling. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM; 2010:1149–58.
50. Leeuw J. Block-relaxation Algorithms in Statistics. In: Bock HH, Lenski W, Richter M, eds. *Information Systems and Data Analysis*. Studies in Classification, Data Analysis, and Knowledge Organization; Springer Berlin Heidelberg; 1994:308–24.
51. Hunter DR, Lange K. A tutorial on MM algorithms. *The American Statistician* 2004;58(1):30–7.
52. Tseng P, Yun S. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming* 2009;117(1-2):387–423.
53. Razaviyayn M, Hong M, Luo ZQ. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization* 2013;23(2):1126–53.

54. Wright S. Coordinate descent algorithms. *Mathematical Programming* 2015;151(1):3–34.
55. Powell MJ. On search directions for minimization algorithms. *Mathematical Programming* 1973;4(1):193–201.
56. De Leeuw J, Patrick M. Multidimensional scaling using majorization: SMACOF in R. *Journal of Statistical Software* 2009;31(3):1–30.
57. Hagmann P, Cammoun L, Gigandet X, Meuli R, Honey CJ, Wedeen VJ, Sporns O. Mapping the Structural Core of Human Cerebral Cortex. *PLoS Biol* 2008;6(7):e159+.
58. Sporns O, Tononi G, Kötter R. The human connectome: A structural description of the human brain. *PLoS computational biology* 2005;1(4):e42+.
59. Hagmann P, Thiran JP, Jonasson L, Vandergheynst P, Clarke S, Maeder P, Meuli R. DTI mapping of human brain connectivity: statistical fibre tracking and virtual dissection. *NeuroImage* 2003;19(3):545–54.
60. Kaufman L, Rousseeuw PJ. Finding groups in data: An introduction to cluster analysis. Wiley; 1990.
61. Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning. Springer; 2009.
62. Silva VD, Tenenbaum JB. Global versus local methods in nonlinear dimensionality reduction. *Advances in neural information processing systems* 2003;15:705–12.