DISTANCE-BASED DIMENSIONALITY REDUCTION FOR BIG DATA

ADRIÀ CASANOVA LLOVERAS

Thesis supervisor
PEDRO DELICADO USEROS (Department of Statistics and Operations Research)

Thesis co-supervisor
CRISTIAN PACHÓN GARCIA (Department of Statistics and Operations Research)

Degree
Master's Degree in Data Science

Master's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

Dimensionality reduction aims to project a data set into a low-dimensional space. Many techniques have been proposed, most of them based on the inter-individual distance matrix. When the number of individuals is really large, the use of distance matrices is prohibitive. There are algorithms that extend MDS (a classical dimensionality reduction method based on distances) to the big data setting. In this TFM, we adapt these algorithms to any generic distance-based dimensionality reduction method.

Contents

1 Introduction, Motivation, and Objectives

1.1 Introduction

Dimensionality Reduction (DR) techniques embed high-dimensional datasets into signficantly lower-dimensional spaces while preserving the structure of the original data. Their primary goal is to tackle the common challenges of working with high-dimensional data, such as sparsity (caused by the curse of dimensionality) or large computational and storage costs.

That being said, DR is mainly used for visualization purposes. In this setting, complex high-dimensional data is projected into $\mathbb{R}^2$, making patterns and clusters more apparent.

Many DR methods have been proposed since Principal Components Analysis (PCA) (the most famous linear method) was first introduced, each with distinct approaches and objectives. In particular, non-linear techniques have turned out to be very useful thanks to their ability to preserve complex relationships. Some examples are Multidimensional Scaling (MDS), Local MDS, Isomap, t-Distributed Stochastic Neighbor Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP) and Autoencoder. All these methods differ in how they define and maintain relationships between points, although all try to preserve global structure and local neighborhoods.

Despite their utility, DR methods face a few limitations. Many algorithms require computing and/or keeping in memory pairwise distances between all data points, resulting in quadratic time complexity and memory requirements. This becomes prohibitive for large datasets with millions of points. Parameter selection presents challenges too, since there is no general consensus on the best way to tune them and k-fold cross-validation is very costly due to the substantial time complexities of these algorithms. Furthermore, information loss is inevitable during dimensionality reduction and understanding which aspects of the data are preserved and which are distorted is crucial to correctly interpret the results.

1.2 Motivation

The recent advent of large data has led to new challenges and technologies to face them. When the number of observations of a dataset is very big, distance-based DR methods become computationally prohibitive because of their quadratic time and memory complexities. For example, working with a dataset of 100,000 individuals and 10 numeric variables would require a system with about 400GB of RAM.

Recently, Delicado and Pachón-García 2024 studied existing and new modifications of MDS to tackle big datasets, demonstrating significant improvements in computational efficiency while maintaining embedding quality. Even though most of them could not be generalized to any DR techniques, two of them followed more common approaches: divide-and-conquer and recursion. However, the latter suffers from low-quality embeddings because it might divide the dataset into too small partitions. Therefore, we were interested in generalizing the divide-and-conquer proposal.

1.3 Objectives

The primary goal of this thesis is to propose a divide-and-conquer framework for any generic distance-based dimensionality reduction method that effectively decreases the method's time and memory complexities. Specifically, we aim to:

1. Review the literature to analyze the properties and applications of the most used DR techniques.

2. Develop a generalized framework for distance-based DR methods that leverages the divide-and-conquer strategy and the Procrustes transformation to reduce time and memory complexities.

3. Implement and parallelize the proposed framework for specific DR algorithms such as non-classical MDS, Local MDS, Isomap and t-SNE.

4. Empirically evaluate the performance of the adapted algorithms in terms of computational efficiency, size limitations and quality of embeddings on benchmark datasets of varying sizes.

5. Compare the results with the traditional counterpart of each tested DR method.

6. Provide guidelines and best practices for selecting and tuning the proposed DR methods based on dataset characteristics and desired properties of the embedding.

The successful completion of these objectives will contribute to making dimensionality reduction techniques accessible for very large datasets, democratizing their use across scientific and industrial applications where data scale is a present challenge.

2 State of the Art

In this section, we aim to review four dimensionality reduction techniques that we will later apply our divide-and-conquer framework to. Next, we will showcase in more detail the work of Delicado and Pachón-García with respect to MDS in big data (Delicado and Pachón-García 2024) to motivate our choosing of the divide-and-conquer approach. Finally, we will discuss three more appreciable solutions to the big data problem in DR and how they relate to our proposal: Landmark Isomap (Silva and J. Tenenbaum 2002), the Out-of-Core Dimensionality Reduction Framework (Reichmann, Hägele, and Weiskopf 2024) and formidable Python t-SNE implementations.

2.1 A few Dimensionality Reduction Techniques

2.1.1 Non-classical MDS. The SMACOF algorithm

SMACOF (Scaling by MAjorizing a COmplicated Function) is a multidimensional scaling algorithm that minimizes metric stress using a majorization technique (Ingwer Borg and P. J. F. Groenen 1997). Also known as the Guttman Transform, this technique is more powerful for this problem than general optimization methods, such as gradient descent.

Algorithm 1 SMACOF

```
Require: $D_{\mathcal{X}} = (\delta_{ij})$, the matrix of observed distances; $q$, the embedding's dimensionality; $n\_iter$, the maximum number of iterations; and $\epsilon$, the convergence threshold.
Ensure: $\tilde{\mathcal{Y}}$, a configuration in a $q$-dimensional space.
1: Initialize $\tilde{\mathcal{Y}}^{(0)} \in \mathbb{R}^{n \times q}$
2: $k \leftarrow 0$
3: repeat
4:     Compute distance matrix of $\tilde{\mathcal{Y}}^{(k)}$: $d_{ij}(\tilde{\mathcal{Y}}^{(k)}) = \|x_i - x_j\|$
5:     Compute the Metric STRESS: $STRESS_M(D_{\mathcal{X}}, \tilde{\mathcal{Y}}^{(k)}) = \sqrt{\frac{\sum_{i<j}(\delta_{ij}-d_{ij})^2}{\sum_{i<j}\delta_{ij}^2}}$
6:     Compute the Guttman Transform: $\tilde{\mathcal{Y}}^{(k+1)} = n^{-1}B(\tilde{\mathcal{Y}}^{(k)})\tilde{\mathcal{Y}}^{(k)}$ where $B(\tilde{\mathcal{Y}}^{(k)}) = (b_{ij})$:
           $$b_{ij} = \begin{cases} -\delta_{ij}/d_{ij}(\tilde{\mathcal{Y}}^{(k)}) & \text{if } i \neq j \text{ and } d_{ij}(\tilde{\mathcal{Y}}^{(k)}) > 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\tilde{\mathcal{Y}}^{(k)}) = 0 \\ -\sum_{j \neq i} b_{ij} & \text{if } i = j \end{cases}$$
7:     $k \leftarrow k + 1$
8: until $k \geq n\_iter$ or $|STRESS_M(D_{\mathcal{X}}, \tilde{\mathcal{Y}}^{(k-1)}) - STRESS_M(D_{\mathcal{X}}, \tilde{\mathcal{Y}}^{(k)}| < \epsilon$
9: return $\tilde{\mathcal{Y}}^{(k)}$
```

2.1.2 Local MDS

Local MDS (Chen and Buja 2009) is a variant of non-classical multidimensional scaling that differs in how large distances are treated. Specifically, a repulsive term between distant points is added to the stress function to further separate points in the low-dimensional configuration.

Parameters τ (which must be in the unit interval) and k may be tuned with k'-cross validation thanks to the LCMC (Local Continuity Meta-Criteria). (*POSSIBLE ANNEX*)

Algorithm 2 Local MDS

```
Require: $D_{\mathcal{X}}$, the matrix of observed distances; $q$, the embedding's dimensionality; $k$,
    the size of neighborhoods; and $\tau$, the weight of the repulsive term.
Ensure: $\tilde{\mathcal{Y}}$, a configuration in a $q$-dimensional space.
1: Compute the symmetrized k-NN graph of $D_{\mathcal{X}}$, $\mathcal{N}$
2: Calculate $t = \frac{|\mathcal{N}|}{|\mathcal{N}^C|} \cdot \text{median}_{\mathcal{N}}\left(D_{i,j}\right) \cdot \tau$
3: Minimize
       $$\sum_{(i,j)\in N} \left(D_{i,j} - \|\mathbf{y}_i - \mathbf{y}_j\|\right)^2 - t \sum_{(i,j)\notin N} \|\mathbf{y}_i - \mathbf{y}_j\|$$
4: return The solution to the optimization problem, $\tilde{\mathcal{Y}}$.
```

2.1.3 Isomap

Isomap (J. B. Tenenbaum, Silva, and Langford 2000) is a nonlinear technique that preserves geodesic distances between points in a manifold. The key insight of Isomap is that large distances between objects are estimated from the shorter ones by the shortest path length. Then, shorter and estimated-larger distances have the same importance in a final MDS step.

Algorithm 3 Isomap

```
Require: $D_{\mathcal{X}}$, the matrix of observed distances; $q$, the embedding's dimensionality; and
    $\epsilon$ or $k$, the bandwidth.
Ensure: $\tilde{\mathcal{Y}}$, a configuration in a $q$-dimensional space.
1: Find the $\epsilon$-NN or k-NN graph of $\mathcal{X}$, $G$.
2: Compute the distance matrix of $G$, $D_G$.
3: Embed $D_G$ to a $q$-dimensional space with MDS.
4: return The output configuration of MDS.
```

The only tuning parameter of Isomap is the bandwidth (ϵ or k), but there is no consensus on what is the best method to choose it.

2.1.4 t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding) (Maaten and Hinton 2008) is a nonlinear dimensionality reduction technique that preserves local neighborhoods by modeling similarities between points as conditional probabilities. The difference between these probability distributions in high and low-dimensional spaces is then minimized.

t-SNE focuses on retaining the local structure of the data while ensuring that every point y_i in the low-dimensional space will have the same number of neighbors, making it particularly effective for visualizing clusters. The use of the Student t-distribution in the low-dimensional space addresses the *crowding problem* by allowing dissimilar points to be modeled far apart (Maaten and Hinton 2008).

Perplexity is interpreted as the average effective number of neighbors of the high-dimensional datapoints x_i and typical values are between 5 and 50.

```
Algorithm 4 t-SNE
Require: $\mathcal{X} \in \mathbb{R}^{n \times p}$, the high-dimensional configuration; $q$, the embedding's dimensionality; perplexity $Perp$.
Ensure: $\tilde{\mathcal{Y}}$, a configuration in a $q$-dimensional space.
1: For every datapoint $i$, find $\sigma_i$ so that the conditional probability ditribution
       $$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$
   has perplexity $2^{-\sum_j p_j \log_2 p_j} = Perp$
2: Symmetrize conditional distributions: $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$ if $i \neq j$, $p_{ii} = 0$
3: Consider Student t-distributed joint probabilities for the low-dimensional data $y_i$:
       $$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{h \neq k}(1 + \|y_h - y_k\|^2)^{-1}}$$
4: Minimize the sum of Kullback-Leibler divergences between the joint distributions over all datapoints:
       $$C(\tilde{\mathcal{Y}}) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$
5: return The solution to the optimization problem, $\tilde{\mathcal{Y}}$.
```

2.2 Multidimensional Scaling for Big Data

Delicado and Pachón-García 2024 compared four existing versions of MDS with two newly proposed (Divide-and-conquer MDS and Interpolation MDS) to handle large data. As can be seen in Figure 1, these can be grouped into four categories:

- **Interpolation-based**: Landmark MDS, Interpolation MDS and Reduced MDS apply classical multidimensional scaling to a subset of $l \ll n$ points and then interpolate the projection of the remaining data. They differ in how the interpolation is computed: Landmark MDS uses distance-based triangulation; Interpolation MDS, the l-points Gower Interpolation Formula; and Reduced MDS, the 1-point Gower Interpolation Formula.

- **Approximation-based**: Pivot MDS approximates the SVD of the full inner product matrix with the SVD of the inner product matrix between a subset of $l \ll n$ points and all the points in the dataset.

- **Divide-and-conquer**: In Divide-and-conquer MDS, the dataset is randomly partitioned into subsets of up to $l \ll n$ points into which MDS is independently applied. Then, the resulting embeddings are aligned with Procrustes transformations.

- **Recursive**: Fast MDS is similar in spirit to Divide-and-conquer MDS, but it partitions the data recursively.

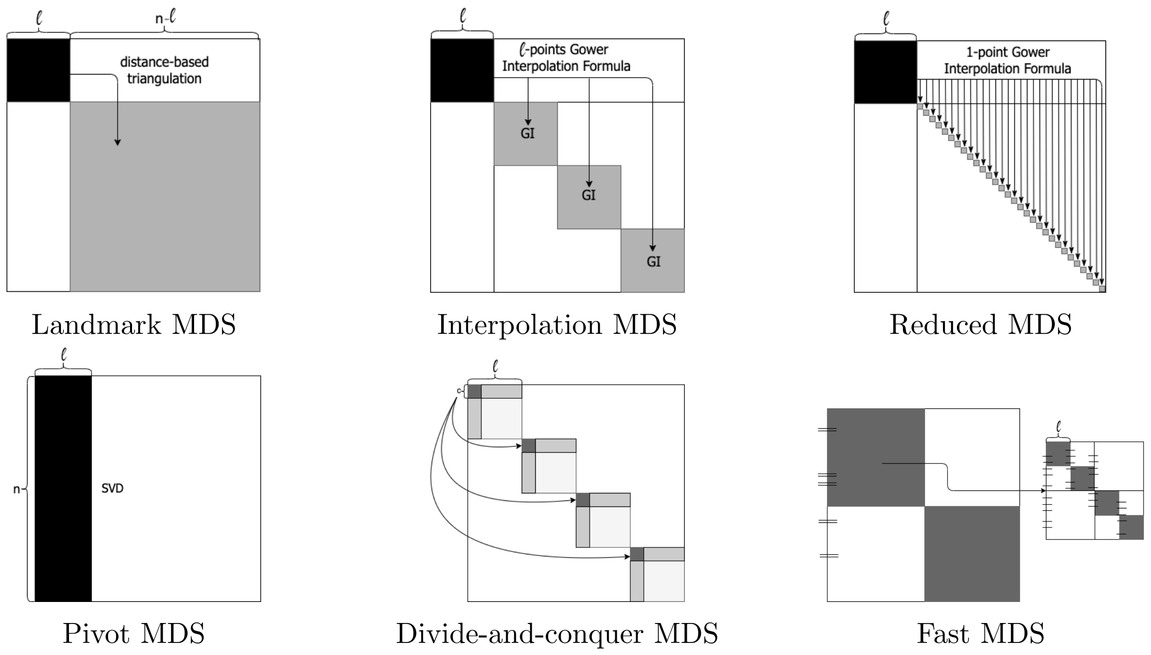
ℓ
n-ℓ
distance-based triangulation
ℓ-points Gower Interpolation Formula
GI
GI
GI
1-point Gower Interpolation Formula
Landmark MDS
Interpolation MDS
Reduced MDS
n
SVD
Pivot MDS
Divide-and-conquer MDS
Fast MDS

Figure 1: Schematic representation of the six MDS algorithms for big data described in Delicado and Pachón-García 2024. (Source: original publication.)

2.3 Landmark Isomap and the Out-of-Core Dimensionality Reduction Framework

Silva and J. Tenenbaum 2002 first introduced Landmark MDS in 2002 by applying it to Isomap (L-Isomap). This way, they reduced the time complexity of both classical multidimensional scaling and Isomap from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 l)$, where $l \ll n$ is the amount of landmark points.

Note that, similarly to Landmark MDS, other big data versions of MDS can be used with Isomap. Nonetheless, interpolation-based and approximation-based algorithms cannot be trivially generalized to nonlinear dimensionality reduction methods.

Later, Reichmann, Hägele, and Weiskopf 2024 proposed the Out-of-Core Dimensionality Reduction Framework. Similar to Interpolation MDS, this algorithm applies a DR method that produces a mapping between high- and low-dimensional spaces (i.e. PCA, MDS, t-SNE, UMAP, Autoencoder) to a small subset of the data and then projects the remaining datapoints in blocks. In order to obtain the aforementioned mappings, Reichmann et. al. gathered different projection mechanisms for every method. PCA and autoencoders learn a parametric mapping between the original data and the embedding, so projecting new points is straightforward. For non-classical MDS, stress is minimized for a single point while keeping others fixed, a process known as *single scaling* in the literature (Basalaj 1999). A similar strategy is used for t-SNE (Zhang et al. 2021) and UMAP (McInnes, Healy, and Melville 2018), which leverage the k-NN of the projecting point to initialize the optimizer.

2.4 openTSNE, an outstanding implementation of t-SNE

Maybe because of its popularity, t-SNE has been optimized for large data environments in Python through iterative implementations. P. G. Poličar, Stražar, and Zupan 2024 considered many of them and published `openTSNE`, a package that includes several t-SNE extensions "*to address scalability issues and the quality of the resulting visualizations*". Furthermore, they compared their proposal with those of other popular packages in serial and parallel configurations. In Figure 2 it can be seen that, thanks to these advances, the challenges of big data have already been solved in the specific case of t-SNE.

(*POSSIBLE ANNEX*)

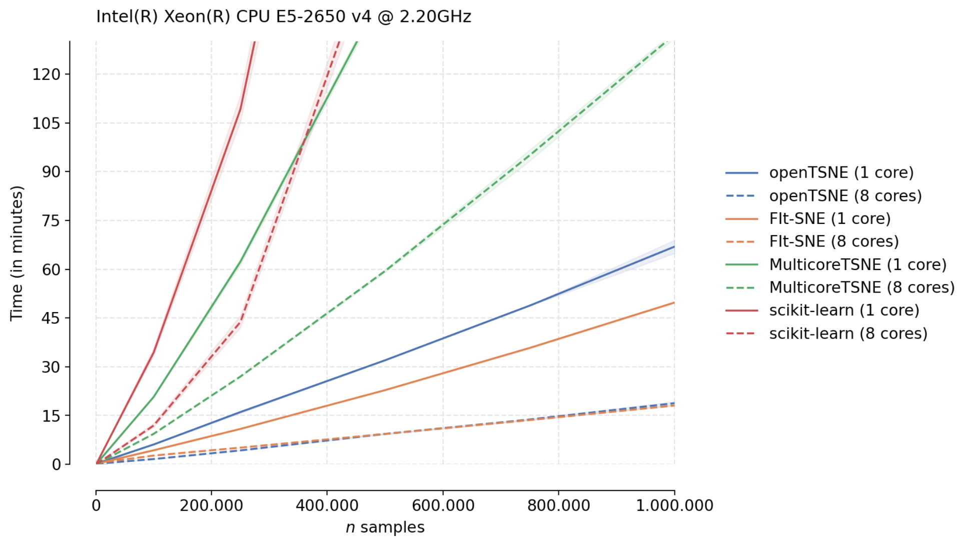

Figure 2: Benchmark of the best open-source t-SNE Python implementations by P. Poličar 2023. (Source: original webpage.)

3 Specification and Design of the Solution

Our solution to the problem of applying dimensionality reduction techniques to large datasets consists in dividing the data into partitions small enough for the computer to process and then merge their embeddings. As stated earlier, we follow the same approach as in Delicado and Pachón-García 2024. By means of an efficient alignment procedure named *Procustes transformation*, we manage to orient every embedding to a specific alignment, with the goal that the final embedding will be coherent. Therefore, even though partitioning the data might induce errors in the embedding's structure, our approach allows any system to tackle arbitrarily large datasets and leverage parallelization.

In order to preserve the topology of the partition's embedding, we narrow Procustes transformations to rigid motions; that is, rotations and reflections. Moreover, we diminish the overhead computation of merging the embeddings by computing the transformation matrix with only a few datapoints. Specifically, if partitions have l (or $l-1$) observations, we sample $c \ll l$ points from the first partition and stack them temporarily to every other partition. Later, we embed the first partition and extract the result of the c sampled points, $\tilde{\mathcal{Y}}_{\mathcal{C}}$. Afterward, for every remaining partition, we project the stacked data with the chosen DR method and separate the c points corresponding to the first partition, $\tilde{\mathcal{Y}}_{\mathcal{C}}^{(i)}$, from those of the current partition, $\tilde{\mathcal{Y}}_i$. This way, we can compute the optimal Procustes transformation between $\tilde{\mathcal{Y}}_{\mathcal{C}}$ and $\tilde{\mathcal{Y}}_{\mathcal{C}}^{(i)}$ and apply it to the larger matrix $\tilde{\mathcal{Y}}_i$, all without needing to process two full partitions.

For a more detailed specification of our solution, see Algorithm 5, which depicts the Divide-and-conquer for Dimensionality Reduction algorithm, and section 3.1, which shows how the Procrustes transformation can be obtained.

Algorithm 5 Divide-and-Conquer for Dimensionality Reduction

```
Require: $\mathcal{X} \in \mathbb{R}^{n \times p}$, the high-dimensional data; $\mathcal{M}$, the DR method; $l$, the partition
    size; $c$, the amount of connecting points; $q$, the embedding's dimensionality; and $arg$,
    the $\mathcal{M}$'s specific parameters.
Ensure: $\tilde{\mathcal{Y}}$, a configuration in a $q$-dimensional space.
 1: if $n \leq l$ then
 2:     return $\mathcal{M}(\mathcal{X}, q, arg)$
 3: end if
 4: Partition data into $k$ subsets: $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$ where $|\mathcal{P}_i| \leq l$ for all $i$
 5: Sample $c$ connecting points from $\mathcal{P}_1$: $\mathcal{C} \subset \mathcal{P}_1$ with $|\mathcal{C}| = c$
 6: Apply DR method to first partition: $\tilde{\mathcal{Y}}_1 = \mathcal{M}(\mathcal{X}_{\mathcal{P}_1}, q, arg)$
 7: Extract embedding of $\mathcal{C}$: $\tilde{\mathcal{Y}}_{\mathcal{C}} = \tilde{\mathcal{Y}}_1[\mathcal{C}, :]$
 8: for $i = 2$ to $k$ do
 9:     Stack connecting points to current partition: $\mathcal{X}_{\text{stack}} = [\mathcal{X}_{\mathcal{C}}; \mathcal{X}_{\mathcal{P}_i}]$
10:     Project stacked data: $\tilde{\mathcal{Y}}_{\text{stack}} = \mathcal{M}(\mathcal{X}_{\text{stack}}, q, arg)$
11:     Separate embedding of $\mathcal{X}_{\mathcal{C}}$: $\tilde{\mathcal{Y}}_{\mathcal{C}}^{(i)} = \tilde{\mathcal{Y}}_{\text{stack}}[1 : c, :]$ and $\tilde{\mathcal{Y}}_i = \tilde{\mathcal{Y}}_{\text{stack}}[(c+1) :, :]$
12:     Align projection using Procrustes: $\tilde{\mathcal{Y}}_i = \text{Procrustes}(\tilde{\mathcal{Y}}_{\mathcal{C}}, \tilde{\mathcal{Y}}_{\mathcal{C}}^{(i)}, \tilde{\mathcal{Y}}_i)$
13: end for
14: Combine all projections: $\tilde{\mathcal{Y}}' = [\tilde{\mathcal{Y}}_1; \tilde{\mathcal{Y}}_2; \ldots; \tilde{\mathcal{Y}}_k]$
15: Reorder rows to match original ordering: $\tilde{\mathcal{Y}}' = \tilde{\mathcal{Y}}'[\text{order}, :]$
16: Apply PCA to center and rotate for maximum variance: $\tilde{\mathcal{Y}} = \text{PCA}(\tilde{\mathcal{Y}}', q)$
17: return $\tilde{\mathcal{Y}}$
```

3.1 Orthogonal Procrustes Transformation

Our problem of aligning the partitions' embeddings is known in the literature as the *Procustes Problem* (I. Borg and P. Groenen 2005). Depending on the kind of fitting desired, many solutions can be found. For example, orthogonal transformations consist of rotations and reflections, but one may also desire dilations and shifts. In fact, the transformation could be any linear distortion.

That being said, in order to preserve the structure of every partitions' embedding, we considered best to narrow the problem down to rigid motions, or in other words, rotations and reflections. Now, let $\mathbf{A} \in \mathbb{R}^{c \times q}$ be the target configuration ($\tilde{\mathcal{Y}}_{\mathcal{C}}$ in Algorithm 5) and $\mathbf{B} \in \mathbb{R}^{c \times q}$ the corresponding testee ($\tilde{\mathcal{Y}}_{\mathcal{C}}^{(i)}$ in Algorithm 5). We wish to fit $\mathbf{B}$ to $\mathbf{A}$ by rigid motions. That is, we want to find the best orthogonal matrix $\mathbf{T}$ such that $\mathbf{A} \simeq \mathbf{BT}$.

To measure the $\simeq$ relation, we may use the sum-of-squares criterion L. Then, the transformation $\mathbf{T}$ should be chosen to minimize L. Expressed in matrix notation, our problem is

$$\min_{\mathbf{T} \in \mathrm{O}(q)} L(\mathbf{T}) = \min_{\mathbf{T} \in \mathrm{O}(q)} \mathrm{tr}(\mathbf{A} - \mathbf{BT})(\mathbf{A} - \mathbf{BT})',$$

where $\mathrm{O}(q)$ is the orthogonal group in dimension q.

(*POSSIBLE ANNEX*)

By expanding the expression of $L(\mathbf{T})$ and applying a lower bound inequality on traces derived by Kristof 1970, I. Borg and P. Groenen 2005 found a global solution to the minimization problem. Let $\mathbf{U\Sigma V}'$ be the singular value decomposition of $\mathbf{A}'\mathbf{B}$, where $\mathbf{U}'\mathbf{U} = \mathbf{I}, \mathbf{V}'\mathbf{V} = \mathbf{I}$, and $\mathbf{\Sigma}$ is the diagonal matrix with the singular values. Then, $L(\mathbf{T})$ is minimal if

$$\mathbf{VU}'.$$

Therefore, the Procrustes procedure we used would be as follows:

Algorithm 6 Procrustes Procedure

```
Require: $\mathbf{A} \in \mathbb{R}^{c \times q}$, the target matrix; $\mathbf{B} \in \mathbb{R}^{c \times q}$, the testee matrix; and $\mathbf{C} \in \mathbb{R}^{m \times q}$, the
    matrix to transform.
Ensure: $\mathbf{C}'$, the matrix $\mathbf{C}$ after alignment.
 1: Multiply $\mathbf{M} = \mathbf{A}'\mathbf{B}$
 2: Compute singular value decomposition: $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}' = \mathrm{SVD}(\mathbf{M})$
 3: Construct orthogonal matrix: $\mathbf{T} = \mathbf{VU}'$
 4: Align $\mathbf{C}$: $\mathbf{C}' = \mathbf{CT}$
 5: return $\mathbf{C}'$
```

4 Development of the Proposal

- DR methods implementations: packages used, problems found during development, tuning of parameters, experiments' methodology

4.1 Python implementation of Divide-and-conquer for DR

Given that R and Python are the standard programming languages in the Data Science field, we chose them as appropriate to implement the divide-and-conquer for DR algorithm. Initially, we aimed to depelop and publish an R library because the thesis directors already had experience with it. However, after reviewing the literature on DR for Big Data Reichmann, Hägele, and Weiskopf 2024, we realized that many solutions were implemented in Python instead. So, in order to leverage the existing coding ecosystem, we switched to Python.

Our code, then, as well as the whole thesis, was documented on an open-source GitHub repository (https://github.com/airdac/TFM_Adria). This system will also allow us to update and share our implementations and experiments easily.

With time, python modules have been structured in a directory tree as a library. Therefore, even though our project has not been published in any Python package index, it effectively works as a library with specific classes and methods. The main function is `divide_conquer`, which implements Algorithm 5 in parallel through the `concurrent.futures` module. `divide_conquer` also depends on private methods and requires a `DRMethod` object as one of its arguments. This class, inherited from `enum.Enum`, lists the supported DR methods in our package, which can be called through the `get_method_function` method.

We have implemented four DR techniques: SMACOF, Local MDS, Isomap and t-SNE. All but Local MDS are wrappers to methods in other Python libraries, which are efficient and parallelized. Specifically, we make us of the `sklearn.manifold` module Pedregosa et al. 2011 for Isomap and SMACOF and `openTSNE` P. Poličar 2023 for t-SNE. Local MDS, on the other hand, is a less popular method and has no public Python implementation at the moment. However, the R library `stops`

Structure of the Python module I've written. Classes and methods.

GitHub repository.

4.2 Python Packages Used

- **In general and for experiments**:
  - numpy
  - pandas
  - matplotlib.pyplot
  - os
  - time
  - sys
  - logging
  - warnings

  - typing
  - shutil

- **In D&C**:
  - concurrent.futures for parallelization

- **For DR methods**:
  - enum
  - scipy.spatial.distance
  - sklearn.manifold.Isomap, sklearn.manifold.smacof
  - openTSNE.TSNE
  - numba, sklearn.neighbors and stops (from R translated to Python) for Local MDS

4.3 Experiment's methodology and tuning of parameters

Experiments' methodology.

Tuning of parameters. A few cases.

5 Experimentation and Evaluation of the Proposal

- Swiss Roll.
- MNIST.
- Time complexity measurements (including MDS).
- Evaluation and comparison with state of the art.

6 Analysis of Sustainability and Ethical Implications

DESCRIPTION OF THIS SECTION FROM THE REGULATION:

It must include an analysis of the impact of the following gender-related technical aspects:

- issues related to data management and analysis
- issues related to equity, where possible biases are identified and assessed both in the data and in the processes carried out in relation to data management and analysis
- actions carried out to eliminate or mitigate such biases

ACTUAL CONTENT:

- D&C can be used in normal computers, while traditional DR methods require supercomputers to work on big datasets. Hence, we will reduce computing emissions.
- Maybe (it needs testing) DR methods could emphasize biases in the data. This happens because when projecting only a few coordinates, small clusters could be left behind in the remaining not projected coordinates and do not show in the final embedding. Hence, in theory, small communities could become invisible.

7 Conclusions

References

Basalaj, Wojciech (1999). “Incremental multidimensional scaling method for database visualization”. In: *Visual Data Exploration and Analysis VI*. Ed. by Robert F. Erbacher, Philip C. Chen, and Craig M. Wittenbrink. Vol. 3643. International Society for Optics and Photonics. SPIE, pp. 149–158. DOI: 10.1117/12.342830.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer. ISBN: 978-0-387-25150-9.

Borg, Ingwer and Patrick J. F. Groenen (1997). *Modern Multidimensional Scaling: Theory and Applications*. Springer. ISBN: 978-0-387-94845-4.

Chen, Lisha and Andreas Buja (2009). “Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Drawing, and Proximity Analysis”. In: *Journal of the American Statistical Association* 104.485, pp. 209–219. DOI: 10.1198/jasa.2009.0111.

Delicado, Pedro and Cristian Pachón-García (2024). “Multidimensional scaling for big data”. In: *Advances in Data Analysis and Classification*. DOI: 10.1007/s11634-024-00591-9.

Kristof, Walter (1970). “A theorem on the trace of certain matrix products and some applications”. In: *Journal of Mathematical Psychology* 7.3, pp. 515–530. DOI: 10.1016/0022-2496(70)90037-4.

Maaten, Laurens van der and Geoffrey Hinton (2008). “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605.

McInnes, L., J. Healy, and J. Melville (2018). “UMAP. Uniform Manifold Approximation and Projection for Dimension Reduction”. ArXiv e-prints. URL: https://doi.org/10.48550/arXiv.1802.03426.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. (2011). “Scikit-learn: Machine learning in Python”. In: *Journal of machine learning research* 12.Oct, pp. 2825–2830.

Poličar, Pavlin (2023). *openTSNE: Extensible, parallel implementations of t-SNE*. Version 1.0.2. URL: https://opentsne.readthedocs.io/en/stable/benchmarks.html.

Poličar, Pavlin G., Martin Stražar, and Blaž Zupan (2024). “openTSNE: A Modular Python Library for t-SNE Dimensionality Reduction and Embedding”. In: *Journal of Statistical Software* 109.3, pp. 1–30. DOI: 10.18637/jss.v109.i03.

Reichmann, Luca, David Hägele, and Daniel Weiskopf (2024). “Out-of-Core Dimensionality Reduction for Large Data via Out-of-Sample Extensions”. In: *2024 IEEE 14th Symposium on Large Data Analysis and Visualization (LDAV)*. Institute of Electrical and Electronics Engineers, pp. 43–53. ISBN: 979-8-3315-1692-5. DOI: 10.1109/LDAV64567.2024.00008.

Silva, Vin and Joshua Tenenbaum (2002). “Global Versus Local Methods in Nonlinear Dimensionality Reduction”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press.

Tenenbaum, Joshua B., Vin de Silva, and John C. Langford (2000). “A Global Geometric Framework for Nonlinear Dimensionality Reduction”. In: *Science* 290.5500, pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

Zhang, Haili, Pu Wang, Xuejin Gao, Yongsheng Qi, and Huihui Gao (2021). “Out-of-sample data visualization using bi-kernel t-SNE”. In: *Information Visualization* 20.1, pp. 20–34. DOI: 10.1177/1473871620978209.

8 Annexes