UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

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 dataset 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 full distance matrices is prohibitive because of their quadratic memory complexity. There are algorithms that extend classical multidimensional scaling (a long-established 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 component analysis (PCA) (being the standard linear method) was first introduced (Pearson, 1901), 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 classical multidimensional scaling (classical MDS) (Torgerson, 1952; Gower, 1966), local multidimensional scaling (local MDS) (Chen and Buja, 2009), Isomap (Tenenbaum, de Silva, and Langford, 2000), t-distributed stochastic neighbor embedding (t-SNE) (Maaten and Hinton, 2008), uniform manifold approximation and projection (UMAP) (McInnes, Healy, and Melville, 2018) and autoencoders (Baldi and Hornik, 1989; Kramer, 1991). All these methods differ in how they define and maintain relationships between points, although most of them 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 nor how to measure the quality of an embedding with a validation set. The substantial time complexities of these algorithms makes k-fold cross-validation very costly as well. Finally, understanding which aspects of the data are preserved and which are distorted when reducing the dimensionality 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 would require to keep a distance-matrix of 10 billion floating-point numbers in memory. If double precision is used, then the computer in use shall have at least 80 GB of RAM.

Recently, Delicado and Pachón-García (2024) studied existing and new modifications of classical 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 arbitrary DR techniques, two of them followed more common approaches: divide-and-conquer and recursion. These consist in partitioning the distance matrix into submatrices small enough for the system to keep in main memory. Then, classical MDS is applied to every partition independently and the resulting embeddings are aligned and merged with Procustes transformations. That being said, the recursive approach suffers from low-quality embeddings because it might divide the distance matrix

into too small partitions. Therefore, we were interested in generalizing the divide-and-conquer proposal to DR methods beyond classical MDS.

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.

1.4 Structure of this master thesis

The chapters of this master's thesis are organized as follows:

- **Chapter 2: State of the art** reviews four dimensionality reduction techniques (non-classical MDS, LMDS, Isomap, and t-SNE) key to this work. It also examines the work of Delicado and Pachón-García (2024) on classical MDS for big data to provide context for the chosen divide-and-conquer approach. Finally, it discusses alternative solutions for reducing the dimensionality of big datasets, such as landmark Isomap and the `openTSNE` Python package.
- **Chapter 3: Specification and design of the solution** details the proposed divide-and-conquer framework. It explains how data is partitioned and how the resulting embeddings are merged with an efficient alignment procedure known as orthogonal Procrustes transformation.
- **Chapter 4: Development of the proposal** describes the implementation of the framework and utilized DR methods in Python, leveraging existing libraries like `concurrent.futures` and `sklearn.manifold`. Moreover, it outlines the methodology for the conducted experiments, including the datasets used (Swiss roll and MNIST) and the tuning of parameters.

- **Chapter 5: Experimentation and evaluation of the proposal** presents the results of the tests performed on divide-and-conquer DR. Specifically, it contains runtime benchmarks for Isomap and t-SNE, an analysis of computational overheads and a qualitative analysis of the embeddings produced by the divide-and-conquer versions of SMACOF, LMDS, Isomap, and t-SNE on the benchmark datasets.
- **Chapter 6: Analysis of sustainability and ethical implications** discusses the environmental benefits of the proposed algorithm in terms of reduced computational storage costs and energy consumption. Aditionally, it addresses how the method can improve the visibility of small communities within datasets by embedding more inidividuals.
- **Chapter 7: Conclusions** summarizes the key findings and outlines potential directions for future research.

2 State of the art

Many nonlinear DR techniques and big data solutions discovered in recent years have lead to our divide-and-conquer proposal. Hence, we shall first review them to understand the advances they have achieved. In Section 2.1, our goal is to review four dimensionality reduction techniques that we will later apply our divide-and-conquer framework to. Next, in Section 2.2, we will showcase in more detail the work of Delicado and Pachón-García (2024) with respect to classical MDS in big data to motivate our choosing of the divide-and-conquer approach. Finally, Sections 2.3 and 2.4 offer a discussion about three more solutions to the big data problem in DR and how they relate to our proposal: landmark Isomap (de Silva and Tenenbaum, 2002), the out-of-core dimensionality reduction framework (Reichmann, Hägele, and Weiskopf, 2024) and novel 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 procedure that minimizes metric stress using a majorization technique (Kruskal, 1964a; Kruskal, 1964b). Given the distance matrix $\mathbf{D_X} = (\delta_{ij})$ of the original high-dimensional data and its corresponding low-dimensional distance matrix $\mathbf{D_{\tilde{Y}}} = (d_{ij})$, metric stress determines how much different $\mathbf{D_X}$ and $\mathbf{D_{\tilde{Y}}}$ are by the expression

$$STRESS_M(\mathbf{D_X}, \mathbf{D_{\tilde{Y}}}) = \sqrt{\frac{\sum_{i<j} \left(\delta_{ij} - d_{ij}\right)^2}{\sum_{i<j} \delta_{ij}^2}}.$$

SMACOF (see Algorithm 1) is faster for this problem than general optimization methods, such as gradient descent, and it consists on iterative Guttman transformations (Guttman, 1968) applied to the embedded data.

2.1.2 LMDS

Local multidimensional scaling (LMDS) is a variant of non-classical multidimensional scaling that differs in how large distances are treated (Chen and Buja, 2009). Specifically, a repulsive term between distant points is added to the stress function to further separate points in the low-dimensional configuration (see Algorithm 2). Parameters τ (which must be in the unit interval) and k may be tuned with k'-cross validation thanks to the LC (Local Continuity) meta-criteria (Chen and Buja, 2009).

2.1.3 Isomap

Isomap (Tenenbaum, de Silva, and Langford, 2000) is a nonlinear technique that preserves geodesic distances between points in a manifold (see Algorithm 3). 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 classical MDS step.

The only tuning parameter of Isomap is the bandwidth, which can be interpreted as the limiting distance of nearest neighbors, ε, or the amount of nearest neighbors per point, k. However, there is no consensus on what is the best method to choose it.

Algorithm 1 SMACOF

Require: $\mathbf{D_X} = (\delta_{ij})$, the $n \times n$ matrix of observed distances; q, the embedding's dimensionality; *n_iter*, the maximum number of iterations; and ε, the convergence threshold.
Ensure: $\tilde{\mathbf{Y}}$, a configuration in a q-dimensional space.

1: Initialize at random $\tilde{\mathbf{Y}}^{(0)} \in \mathbb{R}^{n \times q}$
2: $k \leftarrow 0$
3: **repeat**
4: Compute the distance matrix of $\tilde{\mathbf{Y}}^{(k)}$, $\mathbf{D}_{\tilde{\mathbf{Y}}^{(k)}}$, with entries $d_{ij}^k = \|\tilde{y}_i^{(k)} - \tilde{y}_j^{(k)}\|$
5: Compute the metric stress: $STRESS_M(\mathbf{D_X}, \mathbf{D}_{\tilde{\mathbf{Y}}^{(k)}})$
6: Compute the Guttman transform: $\tilde{\mathbf{Y}}^{(k+1)} = n^{-1}\mathbf{B}(\tilde{\mathbf{Y}}^{(k)})\tilde{\mathbf{Y}}^{(k)}$ where $\mathbf{B}(\tilde{\mathbf{Y}}^{(k)}) = (b_{ij}^k)$:

$$b_{ij}^k = \begin{cases} -\delta_{ij}/d_{ij}^k & \text{if } i \neq j \text{ and } d_{ij}^k > 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}^k = 0 \\ -\sum_{j \neq i} b_{ij}^k & \text{if } i = j \end{cases}$$

7: $k \leftarrow k + 1$
8: **until** $k \geq n\_iter$ or $|STRESS_M(D_{\mathbf{X}}, \tilde{\mathbf{Y}}^{(k-1)}) - STRESS_M(D_{\mathbf{X}}, \tilde{\mathbf{Y}}^{(k)}| < \varepsilon$
9: **return** $\tilde{\mathbf{Y}}^{(k)}$

Algorithm 2 LMDS

Require: $\mathbf{D_X} = (\delta_{ij})$, the $n \times n$ matrix of observed distances; q, the embedding's dimensionality; k, the size of neighborhoods; and τ, the weight of the repulsive term.
Ensure: $\tilde{\mathbf{Y}}$, a configuration in a q-dimensional space.

1: Compute the symmetrized k-NN graph of $\mathbf{D_X}$, $\mathcal{N}_k$
2: Calculate $t = \frac{|\mathcal{N}_k|}{|\mathcal{N}_k^C|} \cdot \text{median}_{\mathcal{N}_k}(\delta_{ij}) \cdot \tau$
3: Minimize the modified stress function:

$$\tilde{\mathbf{Y}} = \min_{\mathbf{Y} \in \mathbb{R}^{n \times q}} \sum_{(i,j) \in \mathcal{N}_k} (\delta_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2 - t \sum_{(i,j) \notin \mathcal{N}_k} \|\mathbf{y}_i - \mathbf{y}_j\|$$

4: **return** $\tilde{\mathbf{Y}}$

Algorithm 3 Isomap

Require: $\mathbf{D_X}$, the $n \times n$ matrix of observed distances; q, the embedding's dimensionality; and ε, the limiting distance of nearest neighbors, or k, the amount of nearest neighbors per point.
Ensure: $\tilde{\mathbf{Y}}$, a configuration in a q-dimensional space.

1: Find the graph $\mathcal{N}$ given by the radius $\varepsilon > 0$ or the k-NN
2: Compute the distance matrix of $\mathcal{N}$, $\mathbf{D}_{\mathcal{N}}$
3: Apply classical MDS to $\mathbf{D}_{\mathcal{N}}$: $\tilde{\mathbf{Y}} = \text{MDS}(\mathbf{D}_{\mathcal{N}}, q)$
4: **return** $\tilde{\mathbf{Y}}$

2.1.4 t-SNE

t-distributed stochastic neighbor embedding (t-SNE) is a nonlinear dimensionality reduction technique that preserves local neighborhoods by modeling similarities between points as conditional probabilities (Maaten and Hinton, 2008). The difference between these probability distributions in high and low-dimensional spaces is then minimized (see Algorithm 4).

t-SNE focuses on retaining the local structure of the data while ensuring that every point $\mathbf{y}_i$ in the low-dimensional space will have the same number of neighbors, making it particularly effective for visualizing clusters. The use of the Student's t distribution with 1 degree of freedom (or Cauchy distribution) in the low-dimensional space addresses the *crowding problem* by allowing dissimilar points to be modeled far apart (Maaten and Hinton, 2008). The characteristic parameter of t-SNE is the perplexity, which can be interpreted as the average effective number of neighbors of the high-dimensional datapoints $\mathbf{x}_i$. Typical values are between 5 and 50.

Algorithm 4 t-SNE

Require: $\mathbf{D_X} = (\delta_{ij})$, the $n \times n$ matrix of observed distances; q, the embedding's dimensionality; $Perp$, the perplexity.

Ensure: $\tilde{\mathbf{Y}}$, a configuration in a q-dimensional space.

1: For every point $\mathbf{x}_i$, find $\sigma_i > 0$ so that the conditional probability ditribution

$$p_{j|i} = \frac{\exp(-\delta_{ij}^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\delta_{ik}^2/2\sigma_i^2)} \text{ if } i \neq j,\ p_{i|i} = 0$$

has perplexity $Perp = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$

2: Symmetrize conditional distributions: $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$

3: Consider Cauchy-distributed joint probabilities for the low-dimensional data $\mathbf{y}_i$:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{h \neq k} (1 + \|y_h - y_k\|^2)^{-1}}$$

4: Minimize the sum of Kullback-Leibler divergences between the joint distributions over all datapoints:

$$\tilde{\mathbf{Y}} = \min_{\mathbf{Y} \in \mathbb{R}^{n \times q}} \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

5: **return** $\tilde{\mathbf{Y}}$

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 datasets. 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

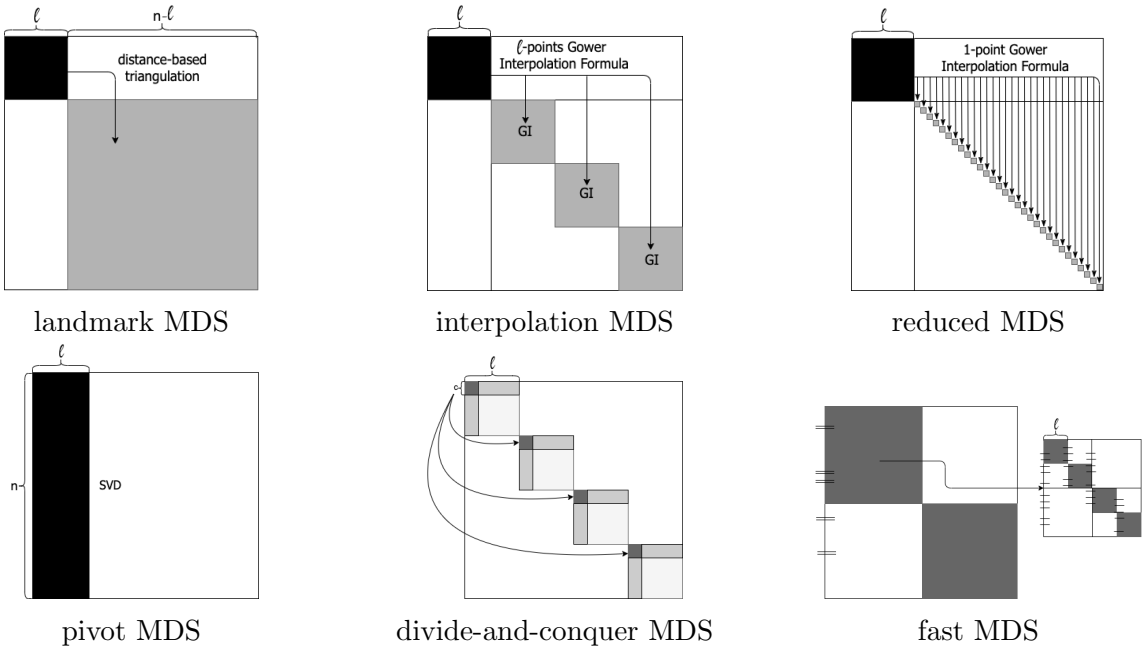

Figure 1: Schematic representation of the six MDS algorithms for big data described in Delicado and Pachón-García, 2024 (Source: original publication).

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.

2.3 Landmark Isomap and the out-of-core dimensionality reduction framework

de Silva and Tenenbaum (2002) first introduced landmark MDS as a step in landmark Isomap, a variation of Isomap that reduced the time complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2 l)$, where $l \ll n$ is the amount of landmark points. Other big data versions of MDS can be used with Isomap similarly. Nonetheless, interpolation-based and approximation-based algorithms cannot be trivially generalized to other nonlinear dimensionality reduction methods because they depend on the eigendecomposition computed in classical MDS.

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,

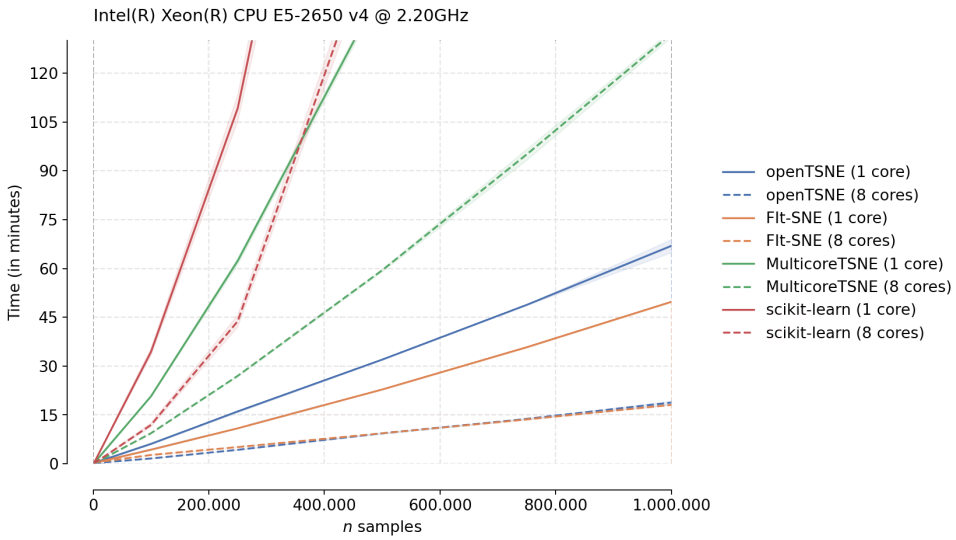

Figure 2: Benchmark of the best open-source t-SNE Python implementations by Poličar (2023) (Source: original webpage).

MDS, t-SNE, UMAP, autoencoders) to a small subset of the data and then projects the remaining datapoints in blocks. In order to obtain the aforementioned mappings, Reichmann, Hägele, and Weiskopf (2024) gathered a different projection mechanism 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. 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.

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 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 modify the embedding's structure, our approach allows any system to tackle arbitrarily large datasets and leverage parallelization.

In order to preserve the geometry 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 $l-1$) points, we sample $c < l$ 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, $\tilde{\mathbf{Y}}_{\mathcal{C}}$. 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, $\tilde{\mathbf{Y}}_{\mathcal{C}}^{(i)}$, from those of the current partition, $\tilde{\mathbf{Y}}_i$. This way, we can compute the optimal Procustes transformation between $\tilde{\mathbf{Y}}_{\mathcal{C}}$ and $\tilde{\mathbf{Y}}_{\mathcal{C}}^{(i)}$ and apply it to the larger matrix $\tilde{\mathbf{Y}}_i$, 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 dimensionality reduction algorithm, and Section 3.1, which shows how the Procrustes transformation can be obtained.

Algorithm 5 Divide-and-conquer dimensionality reduction

Require: $\mathbf{D}_{\mathbf{X}} = (\delta_{ij})$, the $n \times n$ matrix of observed distances; $\mathcal{M}$, the DR method; l, the partition size; c, the amount of connecting points; q, the embedding's dimensionality; and arg, $\mathcal{M}$'s specific parameters.

Ensure: $\tilde{\mathbf{Y}}$, a configuration in a q-dimensional space.

if $n \leq l$ **then**
  return $\mathcal{M}(\mathbf{D}_{\mathbf{X}}, q, arg)$
end if
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
Sample c connecting points from $\mathcal{P}_1$: $\mathcal{C} \subset \mathcal{P}_1$ with $|\mathcal{C}| = c$
Apply DR method to first partition: $\tilde{\mathbf{Y}}_1 = \mathcal{M}(\mathbf{D}_{\mathbf{X}_{\mathcal{P}_1}}, q, arg)$
Extract embedding of $\mathcal{C}$: $\tilde{\mathbf{Y}}_{\mathcal{C}} = \tilde{\mathbf{Y}}_1[\mathcal{C}, :]$
for $i = 2$ to k **do**
  Stack connecting points to current partition: $\mathbf{X}_{\text{stack}} = [\mathbf{X}_{\mathcal{C}}; \mathbf{X}_{\mathcal{P}_i}]$
  Project stacked data: $\tilde{\mathbf{Y}}_{\text{stack}} = \mathcal{M}(\mathbf{D}_{\mathbf{X}_{\text{stack}}}, q, arg)$
  Separate embedding of $\mathbf{X}_{\mathcal{C}}$: $\tilde{\mathbf{Y}}_{\mathcal{C}}^{(i)} = \tilde{\mathbf{Y}}_{\text{stack}}[1:c, :]$ and $\tilde{\mathbf{Y}}_i = \tilde{\mathbf{Y}}_{\text{stack}}[(c+1):, :]$
  Align projection using Procrustes: $\tilde{\mathbf{Y}}_i = \text{Procrustes}(\tilde{\mathbf{Y}}_{\mathcal{C}}, \tilde{\mathbf{Y}}_{\mathcal{C}}^{(i)}, \tilde{\mathbf{Y}}_i)$
end for
Combine all projections: $\tilde{\mathbf{Y}}' = [\tilde{\mathbf{Y}}_1; \tilde{\mathbf{Y}}_2; \ldots; \tilde{\mathbf{Y}}_k]$
Reorder rows to match original ordering: $\tilde{\mathbf{Y}}' = \tilde{\mathbf{Y}}'[\text{order}, :]$
Apply PCA to center and rotate for maximum variance: $\tilde{\mathbf{Y}} = \text{PCA}(\tilde{\mathbf{Y}}', q)$
return $\tilde{\mathbf{Y}}$

3.1 Orthogonal Procrustes transformation

Our problem of aligning the partitions' embeddings is known in the literature as the *Procustes problem* (see, for instance, Borg and Groenen (2005)). Depending on the kind of fitting desired, different 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 and inter-individual distances, we considered best to limit the problem to rigid motions or, in other words, rotations and reflections. Now, let $\mathbf{A} \in \mathbb{R}^{c \times q}$ be the target configuration ($\tilde{\mathbf{Y}}_{\mathcal{C}}$ in Algorithm 5) and $\mathbf{B} \in \mathbb{R}^{c \times q}$ the corresponding testee ($\tilde{\mathbf{Y}}_{\mathcal{C}}^{(i)}$ 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 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.

By expanding the expression of $L(\mathbf{T})$ and applying a lower bound inequality on traces derived by Kristof (1970), a global solution to the minimization problem can be found. 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 $\mathbf{T} = \mathbf{VU}'$. 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

4.1 Python implementation of divide-and-conquer 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 DR algorithm. Initially, we aimed to develop and publish an R library because the thesis directors already had experience with the language. However, after reviewing the literature on DR for big data, we realized that many solutions were implemented in Python instead (Reichmann, Hägele, and Weiskopf, 2024). So, in order to leverage the existing coding ecosystem, we switched to Python. From that moment onward, we documented the code development on an open-source GitHub repository (https://github.com/airdac/TFM_Adria). This system allowed us to easily update and share our implementations and experiments.

With time, the Python functions and classes we have written in different modules have become structured in a directory tree, hence taking the form of a package. Even though our project has not been published in any Python package index yet, it effectively works as a library. 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 and tested four DR algorithms: SMACOF, LMDS, Isomap and t-SNE. All but LMDS are wrappers to methods coded in other efficient and parallelized Python libraries. Specifically, we used the `sklearn.manifold` module (Pedregosa et al., 2011) for Isomap and SMACOF and `openTSNE` (Poličar, 2023) for t-SNE. LMDS, on the other hand, is a less popular method and, up to our knowledge, it has no public Python implementation at the moment. However, the R library `smacofx` (Leeuw and Mair, 2009) does, so we translated it to Python with the help of the `scipy.spatial.distance` and `sklearn.neighbors` modules. This also allowed us to optimize the LMDS runtime with the `numba` just-in-time compiler (Lam, Pitrou, and Seibert, 2015; Aycock, 2003).

As shown in Section 5, our implementation does not add a significant computation overhead when compared to standard DR techniques. In fact, divide-and-conquer DR leverages parallelization and provides a significant speed up to standard DR methods. Most DR techniques have quadratic time complexity (Kruskal, 1964a; Kruskal, 1964b; Chen and Buja, 2009; Maaten and Hinton, 2008), except for Isomap, whose computation time is $\mathcal{O}(n^2 \log(n))$ (Tenenbaum, de Silva, and Langford, 2000). Meanwhile, divide-and-conquer DR applies a DR method on n/l partitions of l individuals, resulting in linear time complexity: $\mathcal{O}(nl)$ usually and $\mathcal{O}(nl \log(l))$ for Isomap. The only DR method we have not been able to accelerate has been t-SNE, given that it already was thouroughly optimized in the `openTSNE` library (Poličar, Stražar, and Zupan, 2024). Finally, since we only keep in memory the inter-individual distance matrix of the partitions being embedded at each moment, our DR framework reduces space complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(l^2)$.

4.2 Experiments' methodology and parameter tuning

Once our framework was developed and we had implemented a few DR methods to test it, we run a series of experiments on the MNIST (Cohen et al., 2017) and the Swiss roll (Tenenbaum, de Silva, and Langford, 2000) datasets. We downloaded the MNIST dataset from the NIST webpage (Standards and Technology, 2024) and generated points on the

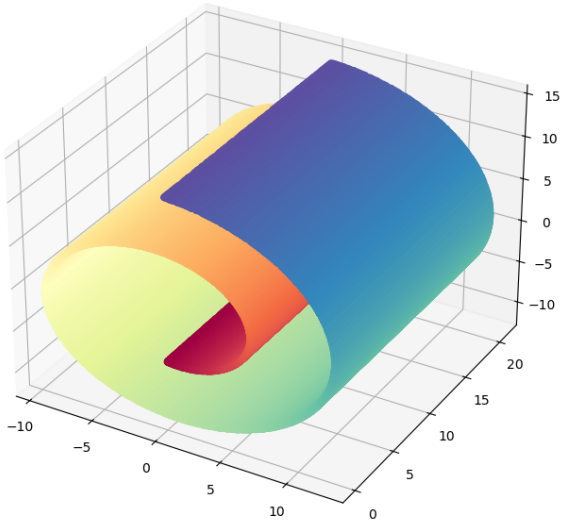

Figure 3: The Swiss roll manifold. Color represents the angle of rotation along the spiral.

Swiss roll manifold with the `sklearn.datasets.make_swiss_roll` function (Pedregosa et al., 2011). All these experiments were executed, logged, plotted and discussed on Jupyter notebooks (Kluyver et al., 2016) with the help of the Python packages `numpy` (Harris et al., 2020), `pandas` (McKinney, 2010), `matplotlib.pyplot` (Hunter, 2007), `os`, `time`, `sys`, `shutil`, `logging` and `warnings`. Furthermore, results were serialized with the `pickle` module in final tests.

The MNIST dataset consisted of 345,035 pictures of hand-drawn digits represented in $\mathbb{R}^{784}$. Each dimension would correspond to the intensity of a pixel in grayscale and there were 28×28 of them. See Figure 4 to observe a few pictures in the dataset. On the other hand, the Swiss roll dataset was sampled from a bidimensional object in $\mathbb{R}^3$ (see Figure 3) shaped like a sheet of paper curved into a spiral along its longest axis. In order to measure the time complexity of divide-and-conquer DR, we randomly generated datasets with sizes ranging between 10^3 and 10^8 points. A good embedding of the Swiss roll into $\mathbb{R}^2$ should intuitively be a rectangle, as if the Swiss roll were unfolded. In MNIST, we would like to obtain 10 clearly distinct clusters corresponding to each digit present in the dataset.

The general experimental methodology we used consisted in a series of steps. Given that working with big datasets is costly, we randomly sampled a subset of 1000 points and embedded them into $\mathbb{R}^2$ with a traditional DR technique. To distinguish it from its divide-and-conquer version, we called it a *bare* technique. Then, before increasing the size of the dataset, we tuned the parameters of the method by applying it to the same dataset with different value combinations. In the tuning process, we took into account the runtime and embedding quality for each value combination. Our goal was to find parameter values that provided a good trade-off between speed and preserving the structure of the data. In Section 5, we enter into more detail on the qualitative assesment of each dataset's embeddings.

Once parameters were tuned with a small enough subset of the data for our system to handle, we applied divide-and-conquer DR to larger datasets with partitions of $l = 1000$ points. In comparison, when we applied the bare techniques to big datasets, execution would crash because of a lack of main memory. Hence, as shown in Section 5, we managed

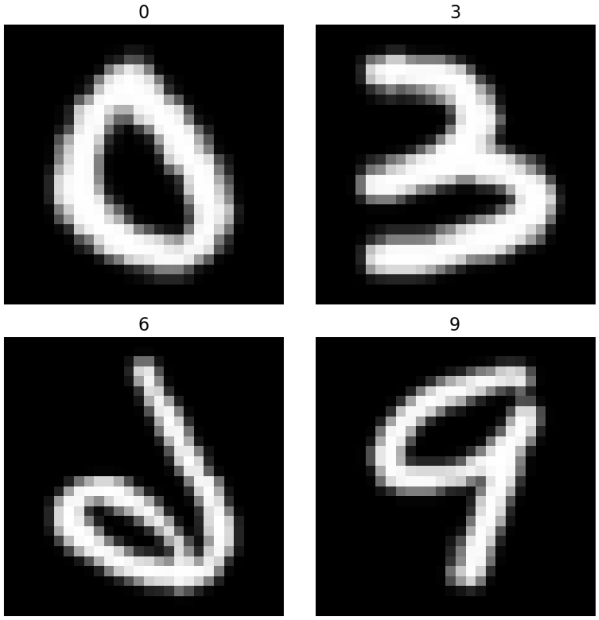

Figure 4: Four sampled images of the MNIST dataset.

to extend the functionality of DR techniques to arbitrarily large datasets.

5 Experimentation and evaluation of the proposal

The goal of this Section is to present the experiments we have conducted on the divide-and-conquer DR algorithm to assess its quality and performance. Following the methodology described in Section 4.2, we were able to corroborate the space and time complexity of the procedure and understand better its embedding mechanism. We chose the Swiss roll and MNIST datasets for the tests because of their popularity, which makes it easier to compare our method with others in the literature, and because of how well Isomap and t-SNE embedded them. That allowed us to go even further and successfully unfold a 10^8 points Swiss roll with divide-and-conquer Isomap. On the other hand, the classification task in the MNIST dataset proved more complicated to our algorithm, which was slower and separated digits worse than `openTSNE`'s implementation of t-SNE.

The experimentation and evaluation process also provided us some insights on the bare SMACOF, LMDS, Isomap and t-SNE techniques. For example, we realized that LMDS, a nonlinear method intended to overcome the limitations of the SMACOF algorithm, was not able to unfold the Swiss roll dataset. Even after having tuned the k and τ parameters (see Algorithm 2), its embedding was porous and irregular instead of uniform and rectangular (see Figure 13). Further research on this problem could lead to the development of a variation of LMDS that reduces the dimensionality of the Swiss roll better.

Additionally, we will describe the computer system we performed the tests on. Even though our main development computer was a Macbook Pro (14-inc, Nov 2023) with 16 GB of RAM and the Apple M3 chip, we noticed that the `concurrent.futures` module, which parallelized the execution of Algorithm 5, did not work in Mac computers with ARM chipsets. Therefore, we ended up using a Windows system. Specifically, our PC was an Asus ROG G513QM-HF026 laptop with the AMD Ryzen 7 5800H CPU, 16 GB of DDR4-3200MHz RAM, an SSD M.2 NVMe PCIe 3.0 and an NVIDIA RTX 3060 GPU.

Concluding, in Section 5.1, we will present the benchmarks performed on Isomap and t-SNE. In Section 5.2, we measure the dividing, embedding and merging steps of divide-and-conquer DR separately to detect possible computation overheads. Finally, in Sections 5.3, 5.4, 5.5 and 5.6, we go through a qualitative analysis of the divide-and-conquer DR algorithm on the previously discussed datasets and dimensionality-reduction methods.

5.1 Runtime benchmarks of divide-and-conquer Isomap and divide-and-conquer t-SNE

Isomap was the first DR method we implemented into our divide-and-conquer framework, so it also was the first method we benchmarked. Later, we measured the runtime of divide-and-conquer t-SNE as well, since t-SNE is the most popular DR method nowadays. We tested divide-and-conquer Isomap on the Swiss roll dataset with different parameter combinations and with parallel and serial computation. However, in all tests, the number of connecting points for the Procrustes transformation was 100. We chose this number because, based on the thesis directors' experience on big data MDS (Delicado and Pachón-García, 2024), it guaranteed good links between partitions' embeddings and efficient computations when $1,000 \leq l \leq 10,000$. Figure 5 shows the average runtimes of 20 experiments with different sets of parameters and dataset sizes. Parameters were previously tuned to ensure embeddings would preserve the structure of the data. As described in Section 4.2, we applied bare Isomap to Swiss rolls of 1,000, 3,162 and 10,000 points,

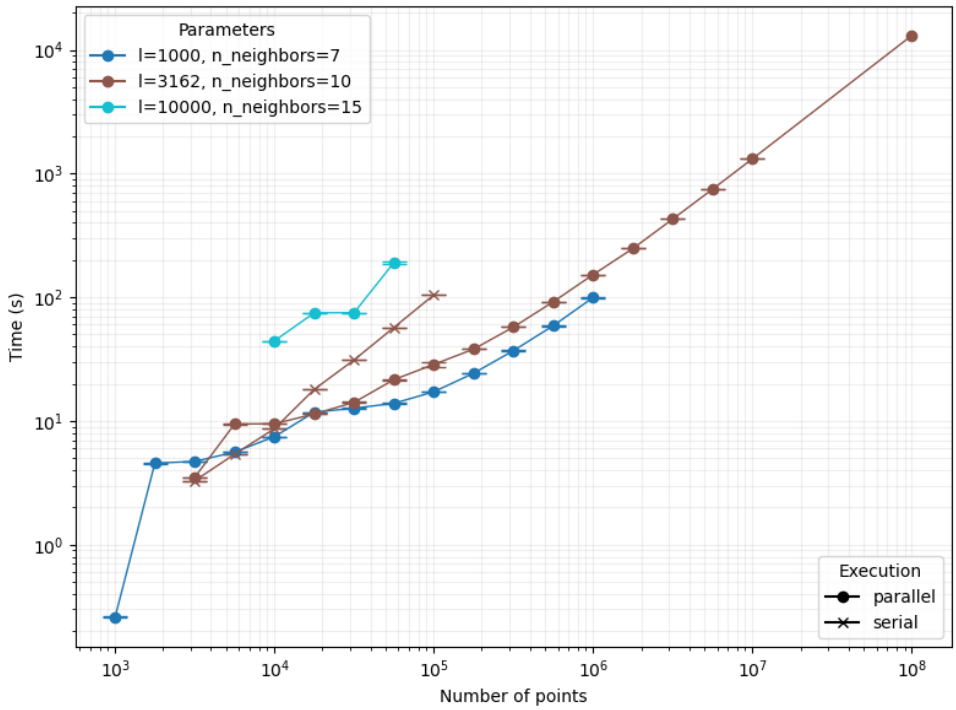

Figure 5: Runtime (s) of divide-and-conquer Isomap averaged over 20 experiments. Tests were performed on datasets generated on the Swiss roll manifold with sizes ranging from 10^3 to 10^8. Data was embedded into $\mathbb{R}^2$ with different parameter combinations and $c = 100$. Runtime in parallel and serial execution is also compared.

following a logarithmic sequence. In fact, the leftest point in every graph corresponds to tests where $l = n$, so only one partition was identified and data was not divided nor merged. This explains the shorter runtimes in these tests. Notice that we increased the number of neighbors k for larger values of l because partitions were denser.

After some experimentation, we realized that if $l = 3162$ and $k = 10$, the embedding was nearly perfect no matter the amount of individuals (see Figure 6). Meanwhile, when $l = 10,000$ and $k = 15$, quality was similar and time was significantly larger, so we used $l = 3162$ and $k = 10$ for the largest datasets. This way, we managed to embed 10^8 three-dimensional points into the Euclidean plane in about 3 hours. Hence, we showed that divide-and-conquer Isomap is capable of handling arbitrarily large datasets on a bare computer while maintaining the quality of the embedding.

Regarding parallelization, we can observe in Figure 5 that it effectively reduces the time complexity of divide-and-conquer DR, although its overhead slows down the algorithm when $n \leq 10^4$. Overall, results show that divide-and-conquer Isomap is linear in time with respect to n.

Afterward, we tested divide-and-conquer t-SNE on the same datasets (see Figure 7). However, t-SNE performed notably slower than Isomap on the Swiss roll, so we only run its divide-and-conquer variation with one parameter combination, $c = 100$, $l = 1,000$, $Perp = 30$ and $n\_iter = 250$, where n_iter is the number of iterations carried out to minimize the Kullback-Leibler divergence (Kullback and Leibler, 1951).

Even though divide-and-conquer t-SNE is about two orders of magnitude slower than divide-and-conquer Isomap, time complexity is linear as well, proving the expected results. The quality of the embedding, on the other hand, is very low. See Figure 8 to observe

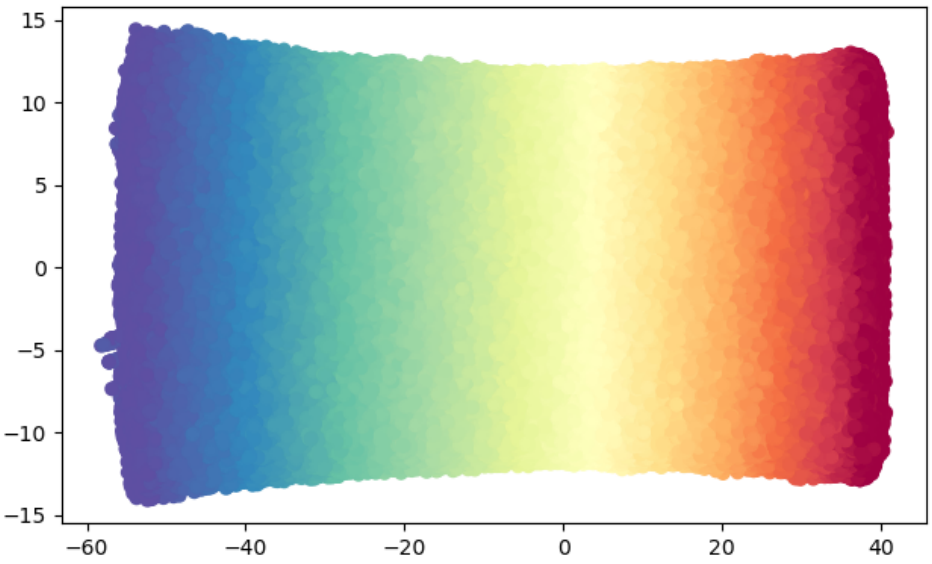

Figure 6: Bidimensional embedding of a 10^8 points Swiss roll dataset computed by divide-and-conquer Isomap with $k = 10$, $l = 3,162$ and $c = 100$. Color represents the angle of rotation along the Swiss roll spiral.

that the structure of the data is broken into separate parts and the spiral shape is not unfolded. The curved shapes in the embedding suggest that this problem might be caused by t-SNE itself, which does not comprehend the intrinsic dimensionality of the data.

5.2 Analysis of possible overheads in divide-and-conquer DR

One reasonable question to formulate about the divide-and-conquer approach is whether splitting the data and merging the resulting embeddings constitue a significant computational cost in comparison to reducing each partition's dimensionality. Theoretically, fractionating the data should be rapid. As for the merging of partitions' embeddings, in Section 3 we depicted how we align and overlap them with Procustes transformations between each embedding and one in specific (the first one). We intentionally find a rigid transformation with a random subset of only $c < l$ points to accelerate its computation, and then multiply the full partition's embedding with the $q \times q$ matrix of the transformation. Therefore, the overhead of merging embeddings should be insignificant.

Table 1 shows the results of an experiment where each part of the divide-and-conquer DR algorithm was independently timed. We uniformly sampled 5,000 points from the MNIST dataset and embedded them into the Euclidean plane with divide-and-conquer SMACOF. The arguments used were $l = 1000$, $c = 100$, $n\_iter = 300$, $\varepsilon = 0.001$. Even though it being a single experiment, we selected commonly used data and results were very clear. Hence, the following conclusions can be extrapolated to other settings. As it was expected, neither partitioning the dataset nor aligning the partial embeddings entail a noteworthy overhead in divide-and-conquer DR. Indeed, the prior and the latter were about 5,506 times and 50,710 times swifter than embedding all partitions with SMACOF, respectively.

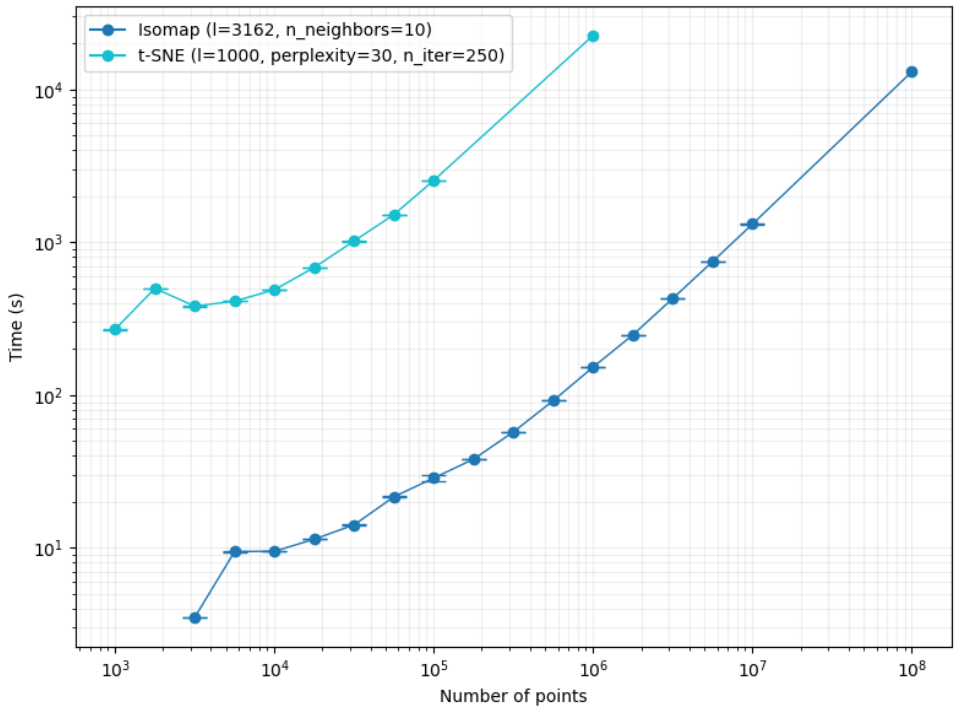

Figure 7: Runtime (s) of divide-and-conquer Isomap and divide-and-conquer t-SNE averaged over 20 experiments. Tests were performed on datasets generated on the Swiss roll manifold with sizes ranging from 10^3 to 10^8. Data was embedded into $\mathbb{R}^2$ with different parameter combinations and $c = 100$.

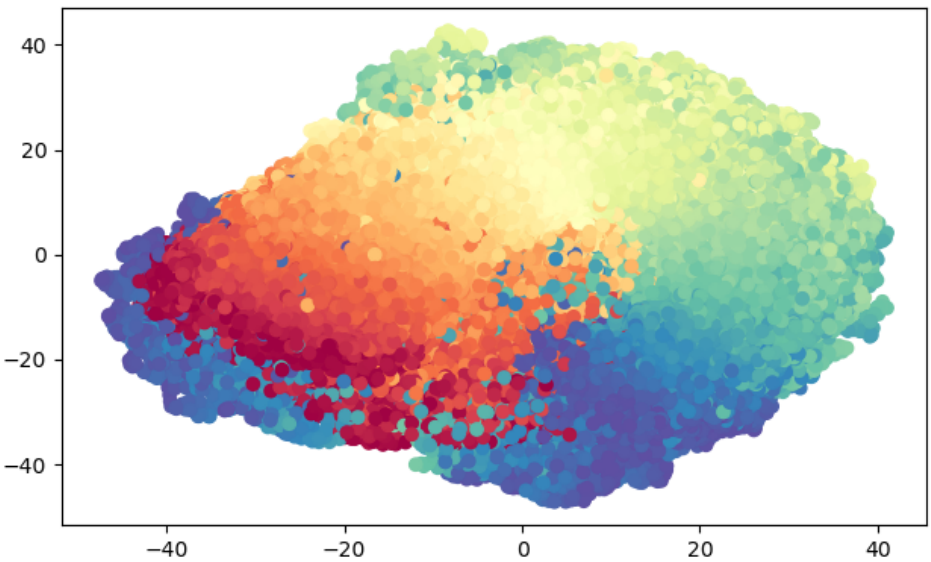

Figure 8: Bidimensional embedding of a 10^6 points Swiss roll dataset computed by divide-and-conquer t-SNE with $l = 1,000$, $c = 100$, $Perp = 30$ and $n\_iter = 250$. Color represents the angle of rotation along the Swiss roll spiral.

Table 1: Runtime (s) of each step of divide-and-conquer SMACOF on a 5000-point random subset of MNIST. The arguments used were $l = 1000$, $c = 100$, $n\_iter = 300$, $\varepsilon = 0.001$.

Operation	Divide	Embed	Merge
Duration (s)	6.88×10^{-3}	37.89	7.47×10^{-4}

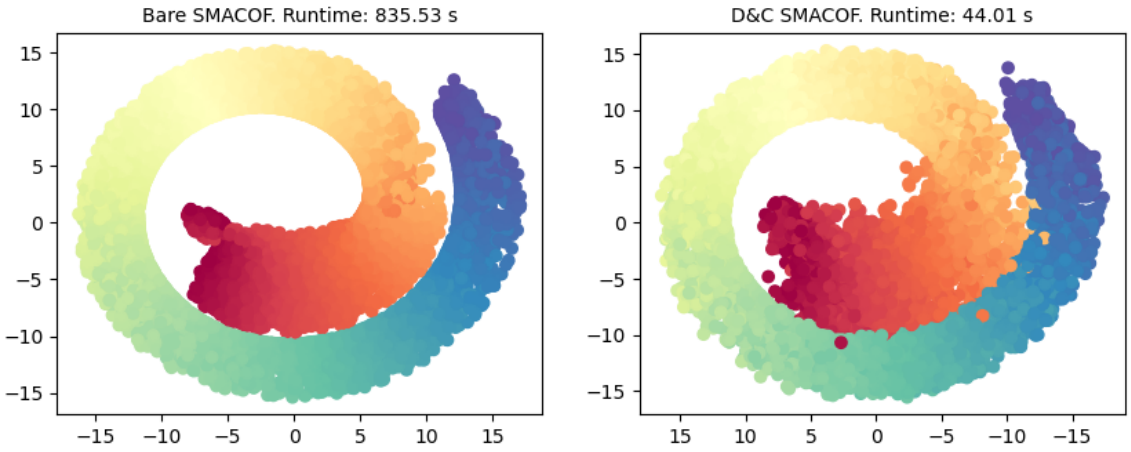

Figure 9: Comparison of the bidimensional embeddings of a 7,500 points Swiss roll dataset by bare (left) and divide-and-conquer (right) SMACOF. The arguments used were $n\_iter = 300$, $\varepsilon = 0.001$ and in divide-and-conquer there also were $l = 1000$ and $c = 100$. Color represents the angle of rotation along the Swiss roll spiral.

5.3 Divide-and-conquer SMACOF

5.3.1 Swiss roll

In this experiment, we applied bare and divide-and-conquer SMACOF to Swiss rolls of different sizes. The largest dataset the bare method could handle ended up having 7,500 individuals and taking about 14 minutes to calculate. We show in Figure 9 the results of the test.

The spiraled shape of both embeddings confirm that SMACOF cannot identify the intrinsic bidimensionality of the Swiss roll. Moreover, SMACOF condenses more the outer part of the manifold than the inner one and presents rugged edges. Regarding the qualitative differences between bare and divide-and-conquer versions, there is not much to be said. Divide-and-conquer SMACOF properly aligns the partial configurations and obtains a similar embedding than bare SMACOF.

To conclude, even though the SMACOF algorithm is not capable of properly embedding the Swiss roll, we can see a clear advantage in using it with the divide-and-conquer framework on other datasets. Indeed, divide-and-conquer SMACOF is about 19 times faster than the bare method while returning a similar low-dimensional configuration.

5.3.2 MNIST

The goal of embedding the MNIST dataset into an Euclidean plane usually is to classify and identify all digits represented in the high-dimensional data. In that sense, this is a common use case of dimensionality reduction: extracting the underlying structure of the data. With the SMACOF algorithm, we sampled 5,000 images from MNIST and tuned

the n_iter and ε parameters as well as l and c to separate as well as possible the low-dimensional configurations of all digits. In other words, we calibrated all parameters to achieve the best possible visual classification of numbers.

Figure 10 shows the result of bare and divide-and-conquer SMACOF. Except for a small cluster of eights, they are very similar and both commit the same problems and achievements. Most digits are clearly separated, except for 4, 7 and 9, which have similar shapes and therefore this behavior could be expected. Additionally, 5 falls between 2 and 8, since some hand-drawn fives resemble upside-down twos and others might look like eights. The former misclassification might denote a difficulty in comprehending the vertical orientation of pictures in SMACOF.

In order to better understand the differences between embeddings, we compared in Figure 11 each coordinate of both configurations against each other. The plots resemble a straight line with $x = y$ equation. There is a cloud of points around the plot and the line of the vertical coordinate is thicker than that of the horizontal one, but overall, and coinciding with Figure 10, the embeddings of bare and divide-and-conquer SMACOF are very similar.

Finally, we computed the Pearson correlation between bare and divide-and-conquer SMACOF in the projected dimensions. In the first dimension it was 0.900 and in the second one it was 0.879. Values close to 1 corroborate a significant resemblance between both embeddings. This evidence indicates that the fastest method should be applied in similar datasets. Meanwhile, our measurements conclude that the divide-and-conquer variation performed about 6 times faster (see Figure 10) than the traditional one, so it would be more preferable in related applications.

The main goal of divide-and-conquer DR is to reduce the dimensionality of big datasets without keeping their full distance-matrices in main memory. In order to assess this feature, we performed one more experiment on divide-and-conquer SMACOF in the MNIST dataset. Specifically, we embedded it all with the parameter values tuned in the previous experiment and obtained the bidimensional configuration depicted in Figure 12. On the other hand, the execution of bare SMACOF crashed due to an absence of main memory. We can observe in Figure 12 that the embedding of the whole dataset took 49 minutes to compute and is alike to that of the 5,000 points sampled subset. Even though the clusters of seven's and nine's images are a bit distinct, which could be due to a misrepresentation in the sampled data, we may notice that divide-and-conquer SMACOF is consistent among related high-dimensional data configurations.

5.4 Divide-and-conquer LMDS

5.4.1 Swiss roll

When we applied bare LMDS to the Swiss roll dataset, we did not expect the embedding we obtained. Figure 13 shows the best configuration in $\mathbb{R}^2$ in the tuning process we run. As it can be seen, points tend to cluster in filaments and leave gaps between them. Moreover, the top and bottom sides of the configuration are bent inwards, effectively bringing together points with the same angle of rotation in the manifold's spiral, something that does not happen at the left or right edges. Given that LMDS is a modification of the SMACOF algorithm that address nonlinearities in the data, we expected it would properly unfold the spiral of the Swiss roll. However, while it does understand the intrinsic dimensionality of the manifold better than SMACOF (see Section 5.3.1), it does not preserve the uniformity of the data nor its global shape.

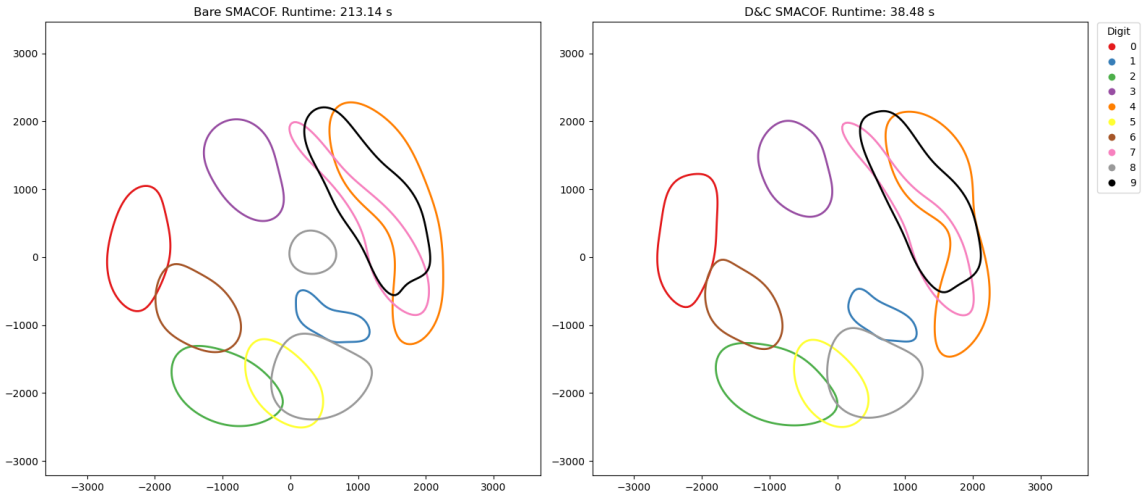

Figure 10: Kernel density estimation of the bidimensional embeddings of a 5,000 points subset of MNIST by bare (left) and divide-and-conquer (right) SMACOF. The arguments we used were $n\_iter = 300$, $\varepsilon = 0.001$ and in divide-and-conquer there also were $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

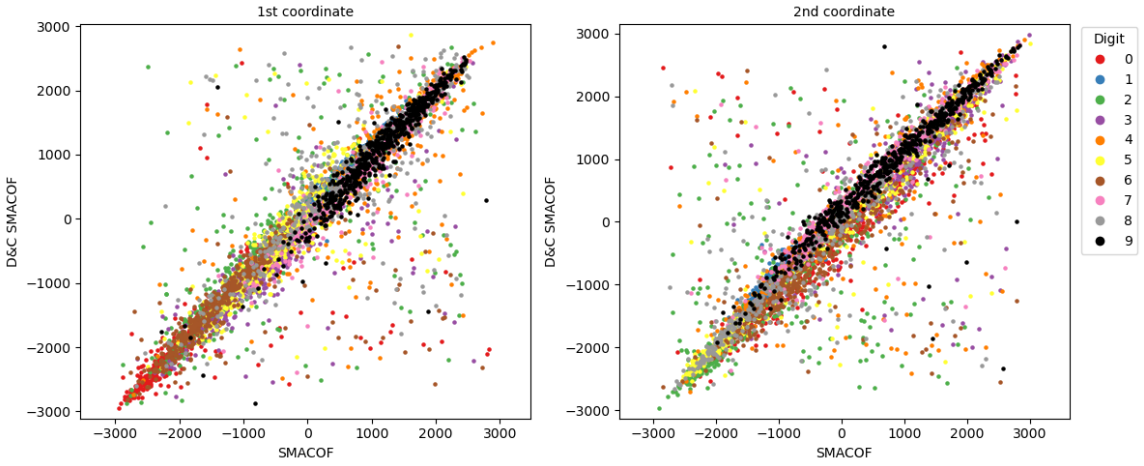

Figure 11: Scatter plots of first (left) and second (right) coordinates in the embeddings represented in Figure 10. SMACOF is compared against divide-and-conquer SMACOF.

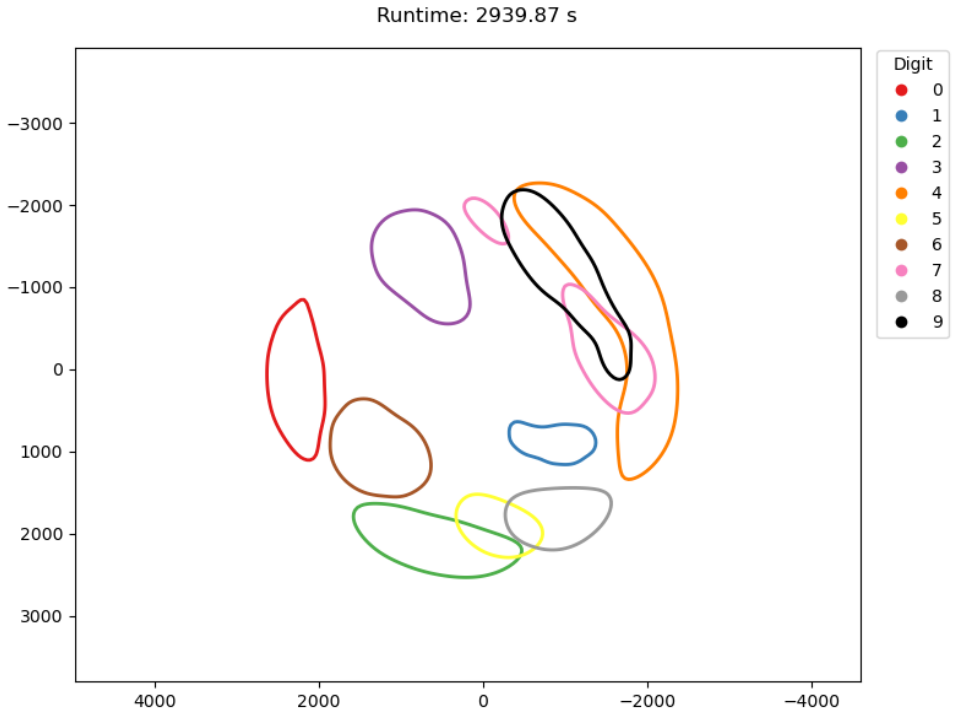

Figure 12: Kernel density estimation of the bidimensional embeddings of the whole MNIST dataset by divide-and-conquer SMACOF. The arguments used were $n\_iter = 300$, $\varepsilon = 0.001$, $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

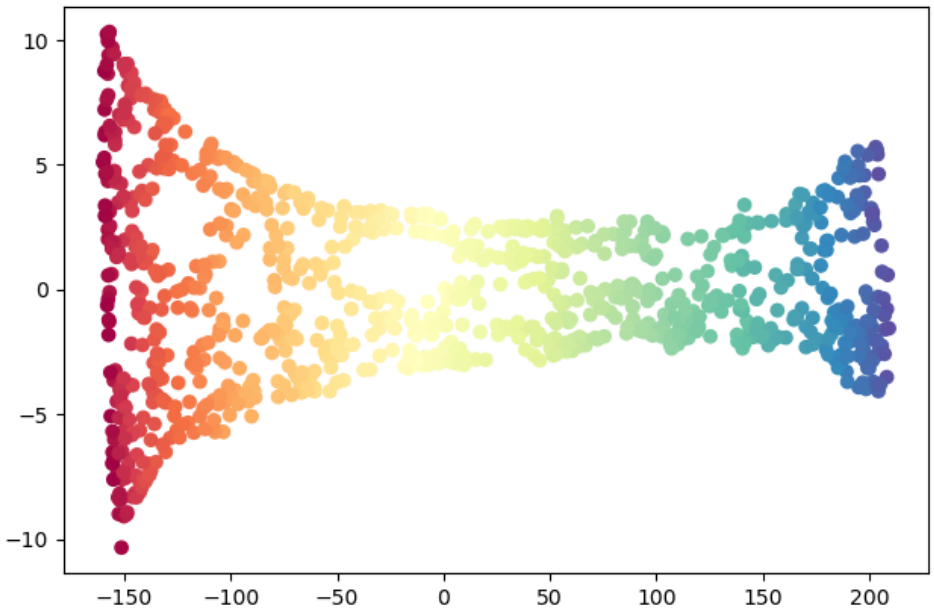

Figure 13: Bidimensional embedding of a 1,000 points Swiss roll dataset computed by LMDS with $k = 10$ and $\tau = 0.1$. Color represents the angle of rotation along the Swiss roll spiral.

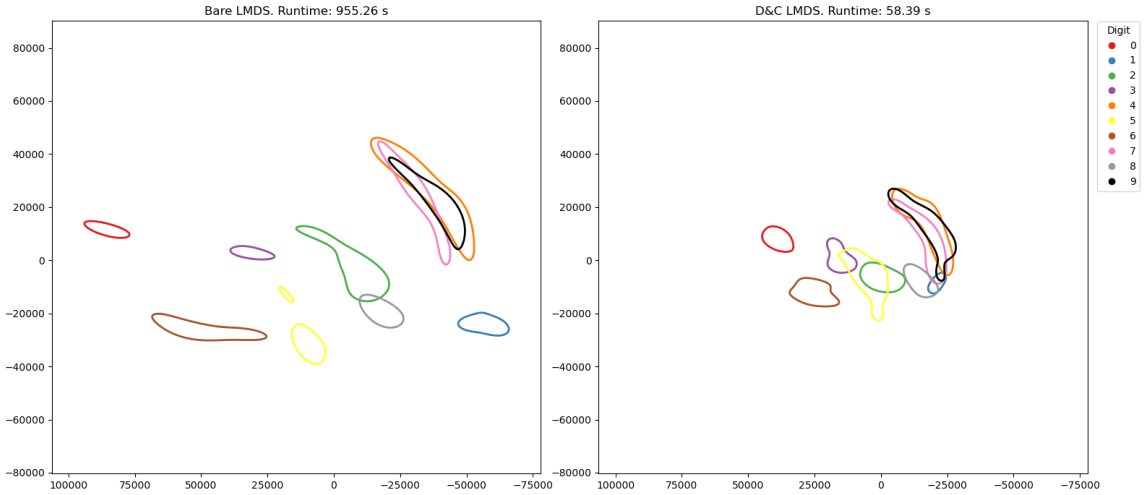

Figure 14: Kernel density estimation of the bidimensional embeddings of a 5,000 points subset of MNIST by bare (left) and divide-and-conquer (right) LMDS. The arguments used were $k = 10$, $\tau = 1$ and in divide-and-conquer there also were $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

5.4.2 MNIST

In order to experiment with LMDS and the MNIST dataset, we considered the same 5,000 points subset of the data used in Section 5.3.2 and tuned the parameters of the bare and divide-and-conquer techniques. In Figure 14, both planar optimal configurations are represented. Bare LMDS widely separates most digits, except for the commonly confused 4, 7 and 9. On the other hand, divide-and-conquer LMDS condenses the data into a narrower region of the plane, so the clusters of digits inevitably overlap. Apart from this problematic behavior, relative positions of digits' clusters are similar in both embeddings.

When we observe the dimensional correlation between bare's and divide-and-conquer's outcomes, we may notice they are very high: 0.949 in the first dimension and 0.821 in the second one. These values could be explained by the fact that the digits' clusters in both embeddings have similar relative positions.

Given the poorer embedding quality of divide-and-conquer LMDS, it might not be worth applying it to small datasets for classification. That being said, our framework achieves a non-contemptible 1,536% acceleration to the computing time of LMDS. Furthermore, Figure 15 shows that, contrary to bare LMDS, it can also tackle the whole MNIST dataset in a conventional computer and a reasonable time (about 35 minutes).

5.5 Divide-and-conquer Isomap

We discussed the embedding of divide-and-conquer Isomap for Swiss rolls of different sizes in Section 5.1. As Spiwokv (2007) first discovered, Isomap flawlessly reduces the dimensionality of the Swiss roll manifold. Moreover, divide-and-conquer Isomap achieves the same embedding quality in big datasets when $k = 10$, $l = 3,162$ and $c = 100$.

5.5.1 MNIST

Figure 16 shows that the big success of Isomap on the Swiss roll is not repeated on the MNIST dataset. When it comes to classification, bare and divide-and-conquer Isomap,

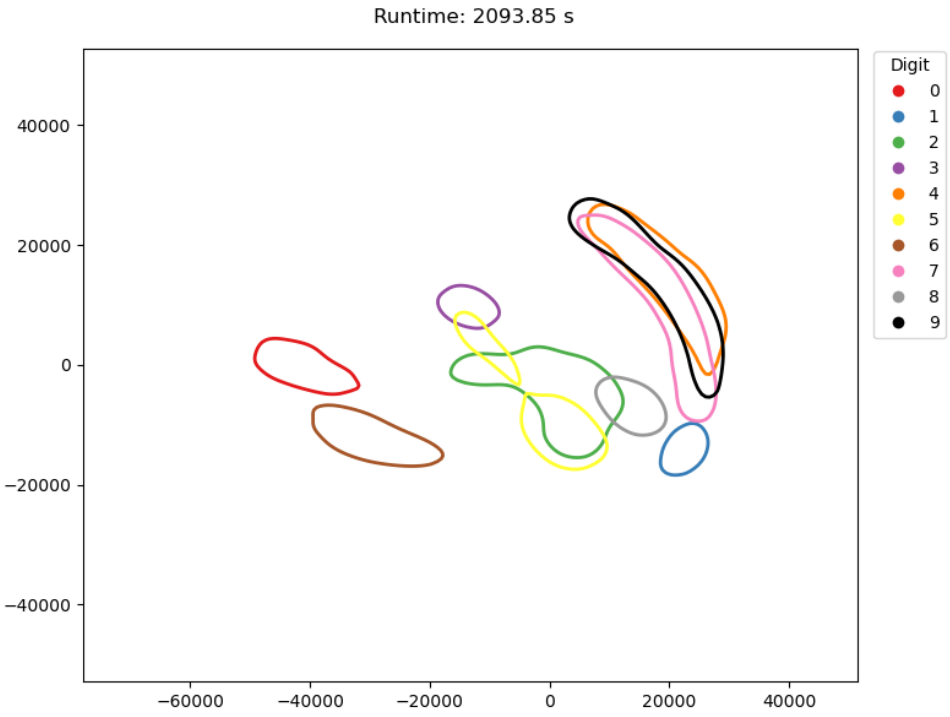

Figure 15: Kernel density estimation of the bidimensional embeddings of the whole MNIST dataset by divide-and-conquer LMDS. The arguments used were $k = 10$, $\tau = 1$, $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

overlap many clusters, thus showing a poor understanding of the intrinsic structure of the data. As usual, digits 4, 7 and 9 are very often confused. On bare Isomap, we also observe intersections between: 0' and 6'; 1' and 2'; 5' and 6'; and 2' and 8' clusters. Divide-and-conquer Isomap enlarges most intersections and also confuses the clusters of twos and sevens, although it separates zeroes and sixes.

When we look at the Pearson correlation between each method across all dimensions, we observe they do not convey the previously observed qualitative differences. In particular, the correlation between the first dimension is 0.920 and between the second one is 0.862. Finally, note that, even though we are using a very fast implementation of Isomap that surpasses SMACOF and LMDS in performance, the divide-and-conquer approach makes it 4 times faster.

As we did with SMACOF and LMDS, we considered previously tuned parameters of divide-and-conquer Isomap and embedded the whole MNIST dataset. Figure 17 plots the kernel density estimation of the outcome. Overall, even if clusters are shaped differently than in the 5,000 points configuration (see Figure 16), it essentially confuses digits the same way. Runtime is remarkably fast, anyway, since divide-and-conquer Isomap embedded 345,035 pictures into $\mathbb{R}^2$ in only two minutes.

5.6 Divide-and-conquer t-SNE

Since t-SNE is especially well suited for visualization tasks, this method is known to greatly group digits in MNIST (Maaten and Hinton, 2008). On the other hand, in Section 5.1 we presented its embedding of the Swiss roll dataset. From Figure 8, we concluded

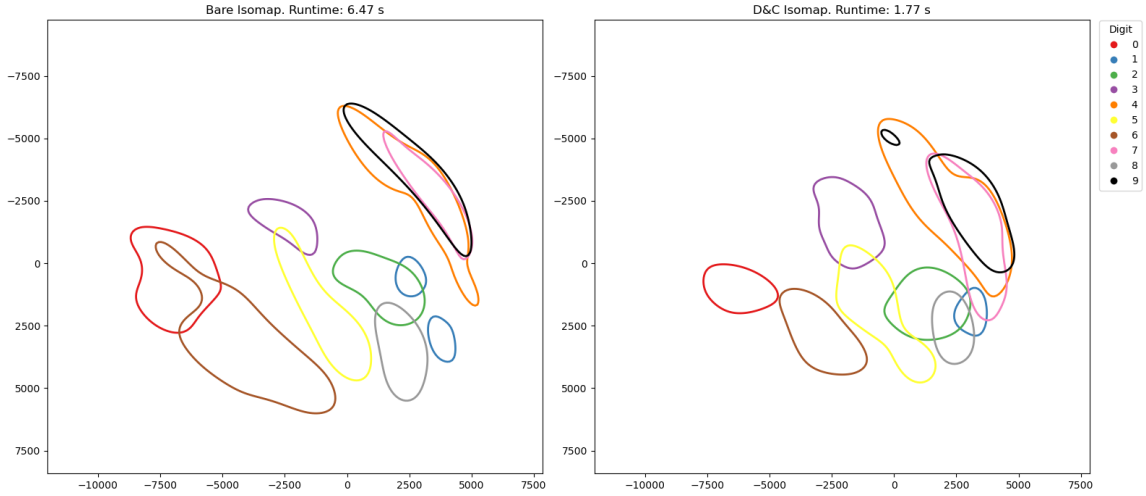

Figure 16: Kernel density estimation of the bidimensional embeddings of a 5,000 points subset of MNIST by bare (left) and divide-and-conquer (right) Isomap. The arguments used were $k = 5$ and in divide-and-conquer there also were $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

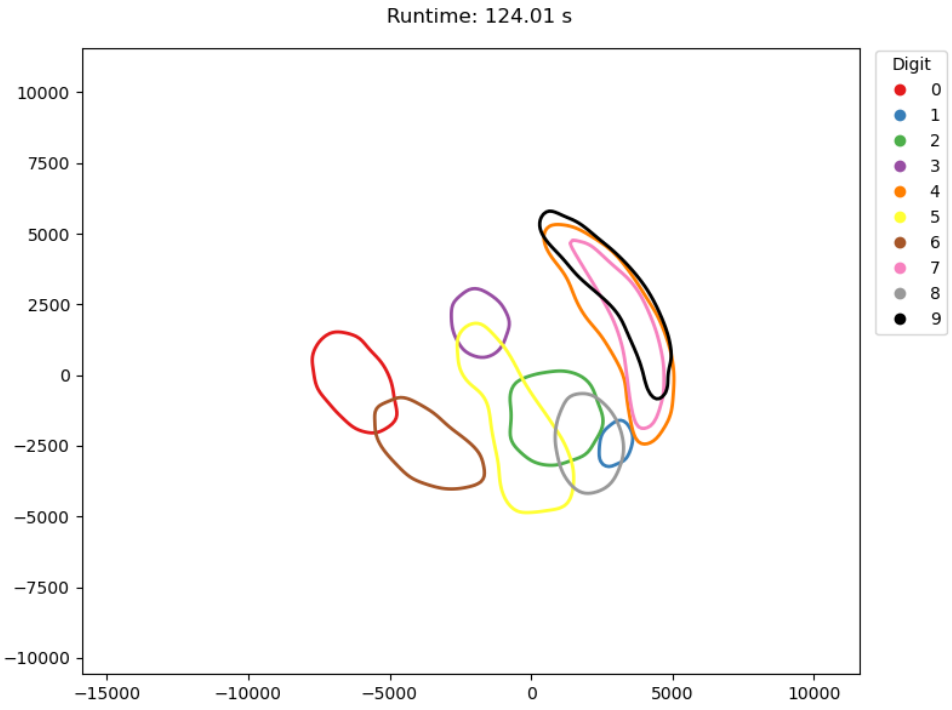

Figure 17: Kernel density estimation of the bidimensional embeddings of the whole MNIST dataset by divide-and-conquer Isomap. The arguments used were $k = 5$, $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

that partial configurations were inconsistent and Procrustes transformations were not able to merge then into a consistent global strucutre.

5.6.1 MNIST

Even though t-SNE is very used nowadays (Wattenberg, Viégas, and Johnson, 2016), we have detected inconsistencies when embedding similar datasets. This could be fatal for divide-and-conquer DR because partial embeddings are only aligned with rigid motions, so geometric differences could not be vanished in the merging step. Hence, before comparing bare and divide-and-conquer t-SNE, we considered two non-intersecting partitions of the MNIST dataset with 172,517 points each and embedded them into the Euclidean plane. Figure 18 shows the kernel density estimation of every digit's pictures in both partitions. Even though most clusters are similar between partitions, twos and fours are swapped in partition 2. Therefore, these digits perfectly classified by bare t-SNE, would be confused by the divide-and-conquer variation were these partitions taken. What this means is that, as seen with Isomap (see Figure 8), divide-and-conquer t-SNE will probably diminish the local and global quality of MNIST's embedding.

Figure 19 and Figure 20 show the embeddings of bare and divide-and-conquer t-SNE on the previously sampled 5,000 points of MNIST and the whole dataset, respectively. As expected, bare t-SNE completely separates different digits, while divide-and-conquer t-SNE overlaps many of their clusters, especially in the smaller dataset. The dimension correlation between both methods on the 5,000 points dataset measures this disagreement with a 0.328 correlation in the second dimension, although the first dimensions have a correlation of 0.817.

Moreover, as explained in Section 2.4, we are using a very performant implementation of t-SNE provided by `openTSNE`, which can tackle big datasets on its own. This results in the bare method being faster than our divide-and-conquer approach, which had not happened with SMACOF, LMDS nor Isomap. Even though in the 5,000 points subset divide-and-conquer t-SNE was 1.3 times faster than t-SNE, in the full dataset, `openTSNE`'s implementation was 144 times faster than ours. Concluding, t-SNE has received many specific optimizations that make it very hard to improve.

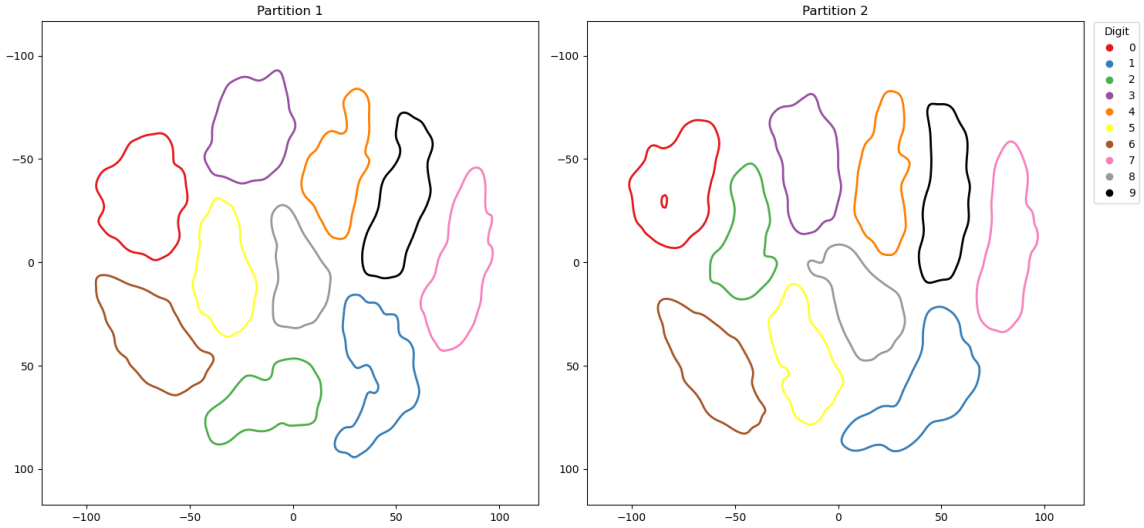

Figure 18: Kernel density estimation of the bidimensional embeddings of two halves of the MNIST dataset. Data was randomly ordered before being splitted. The DR method used was divide-and-conquer t-SNE with $Perp = 30$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

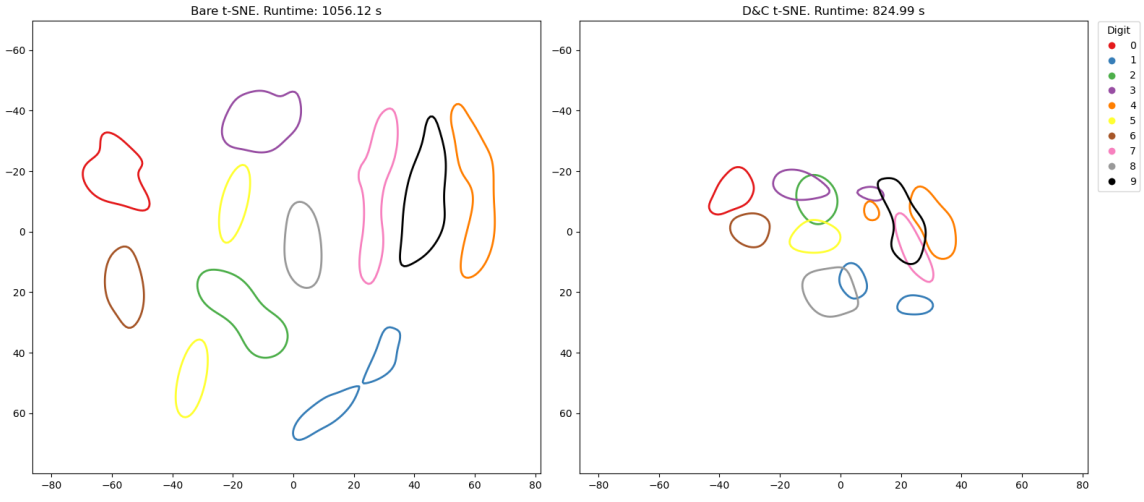

Figure 19: Kernel density estimation of the bidimensional embeddings of a 5,000 points subset of MNIST by bare (left) and divide-and-conquer (right) t-SNE. The arguments used were $Perp = 30$ and in divide-and-conquer there also were $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

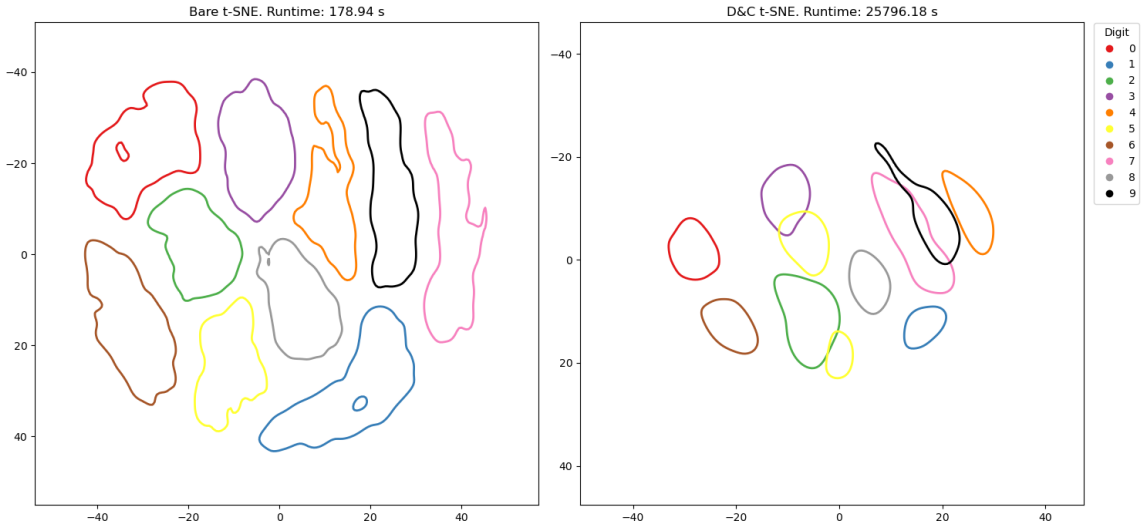

Figure 20: Kernel density estimation of the bidimensional embeddings of the whole MNIST dataset by bare (left) and divide-and-conquer (right) t-SNE. The arguments used were $Perp = 20$, $n\_iter = 100$ and in divide-and-conquer there also were $l = 1000$ and $c = 100$. Contour lines are at 70% of the maximum estimated density for each digit and embedding.

6 Analysis of sustainability and ethical implications

6.1 GHG emissions

The rise of data collection, transfer and computation brought by big data has caused the construction and operation of a big amount of data centers around the world. This infrastructure consumes a lot of electricity and therefore is responsible for abundant greenhouse gases (GHG) emissions that cannot be neglected (IEA, 2025; Brierley, 2023). In order to cope with the current demand of AI and ML tools without creating too large of an impact on the environment, their energetic cost should be reduced as much as possible, mainly through algorithm innovation, decrease of data transfer and storage costs, and improvement of CPUs', GPUs' and TPUs' efficiency.

Our divide-and-conquer framework falls into the first category of energy reduction modes by reducing the runtime and required system capabilities to reduce the dimensionality of a dataset. Indeed, the global warming potential of the GHG emitted when running a procedure on any given computer linearly depends on the time it takes to complete (Lannelongue, Grealey, and Inouye, 2021). Therefore, our divide-and-conquer framework not only reduces the time complexity of DR but also its pollution complexity. Additionally, our framework makes DR less dependent on supercomputers (which contaminate throughout its construction, maintanance and operation) thanks to its low space complexity.

Concluding, even though there is still an environmental cost to consider in DR, given that it is not such a common task as prompting LLMs or other neural networks, we are confident that our algorithm is currently sustainable and it does not pose a threat to the environment. Actually, using our algorithm instead of traditional DR methods would reduce the left carbon footprint.

6.2 Visibility of small communities

DR methods can emphasize biases present in the data. This happens because when projecting only a few coordinates of a dataset, small clusters can be left behind in the remaining, not projected, coordinates and do not show in the final embedding. Hence, small communities can become invisible.

However, our divide-and-conquer framework allows users to diminish the bias added by DR techniques by increasing the number of individuals embedded. Indeed, adding more individuals also means increasing the chance to represent small communities in the low-dimensional configuration of the data. So, by utilizing divide-and-conquer DR and providing more data, the resulting embedding can become more truthful to diverse realities than if bare DR techniques were used instead.

We refer to Figures 10 and 12 for an example of the above-mentioned pehonomenon. While bare SMACOF is only capable of embedding small subsets of the MNIST dataset, divide-and-conquer SMACOF effectively embeds the whole dataset. This results in bare SMACOF homogenizing the class of pictures of sevens and divide-and-conquer SMACOF detecting two distinct ways of handwriting sevens which were misrepresented in the sampled subset. As a matter of fact, the kernel density estimation of the configuration embedded by divide-and-conquer SMACOF shows two separate modes in the class of sevens. Had we considered a social dataset instead of MNIST, divide-and-conquer DR would have allowed us to reduce the dimensionality of the data without discriminating underrepresented communities. Thus, proving our framework useful in these scenarios.

7 Conclusions

The primary objective of this thesis was to develop and evaluate a generalized divide-and-conquer framework for distance-based dimensionality reduction methods to make them applicable to large datasets. We successfully met the objectives of this research by first reviewing the literature on prominent DR techniques and then developing a generalized framework that uses a divide-and-conquer strategy with orthogonal Procrustes transformations to reduce time and memory complexities. We implemented the framework in Python for non-classical MDS (SMACOF), LMDS, Isomap, and t-SNE, although it is easily extendable to any distance-based DR technique.

Later, we experimented with the framework by measuring its runtime and size limitations and assessing the embedding quality it provided on benchmark datasets. Results were mostly favorable, the most impressive one being the flawless embedding of a 100 million points Swiss roll on a standard computer in about 3 hours. In comparison, bare DR methods would crash when trying to process datasets with more than 10,000 observations because of lacking memory. We also contrasted the embeddings of the bare and divide-and-conquer versions of SMACOF, LMDS, Isomap and t-SNE on smaller datasets. Results showed that divide-and-conquer DR was remarkably faster than bare methods while presenting small differences in the resulting embeddings. The only exception of this behaviour was found on t-SNE, since its `openTSNE` implementation is very optimized and clearly outruns our framework both in runtime and embedding quality.

Ultimately, this work contributes to making advanced DR techniques more accessible and sustainable for the large-scale data challenges in science and industry by mitigating the prohibitive quadratic time and memory complexities of distance-based dimensionality reduction methods.

7.1 Future work

Based on the research and findings of this thesis, we have identified several avenues for future work. To start with, the implemented code could be formalized into a distributable Python package to make it more accessible. Moreover, a comparison could be conducted between our divide-and-conquer framework and the out-of-core approach by Reichmann, Hägele, and Weiskopf (2024), since both reduce the space and time complexities of already established DR methods. In terms of less abstract tasks, we realized during the testing of LMDS that it performed worse than expected on the Swiss roll dataset. Therefore, further investigation on LMDS could lead to a better version of this algorithm that would handle the nonlinearities present in the Swiss roll dataset.

Regarding experimentation, future tests could explore how the number of connecting points, c, affects performance and embedding quality. Additionally, testing divide-and-conquer DR on more datasets and use cases would further validate its robustness and generalizability.

References

Aycock, John (2003). "A brief history of just-in-time". In: *ACM Comput. Surv.* 35.2, pp. 97–113. DOI: 10.1145/857076.857077.

Baldi, Pierre and Kurt Hornik (1989). "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural Networks* 2.1, pp. 53–58. DOI: 10.1016/0893-6080(89)90014-2.

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.

Brierley, Craig (2023). *Big data's hidden cost: The carbon footprint of computational science*. URL: https://www.cam.ac.uk/stories/green-algorithms (visited on 06/17/2025).

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.

Cohen, Gregory, Saeed Afshar, Jonathan Tapson, and André van Schaik (2017). "EMNIST. Extending MNIST to handwritten letters". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217.

de Silva, Vin and Joshua B. 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.

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.

Gower, John C (1966). "Some distance properties of latent root and vector methods used in multivariate analysis". In: *Biometrika* 53.3-4, pp. 325–338. DOI: 10.1093/biomet/53.3-4.325.

Guttman, Louis (1968). "A General Nonmetric Technique for Finding the Smallest Coordinate Space for a Configuration of Points". In: *Psychometrika* 33.4, pp. 469–506. DOI: 10.1007/BF02290164.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant (2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

Hunter, J. D. (2007). "Matplotlib. A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

IEA (2025). *Energy and AI*. URL: https://www.iea.org/reports/energy-and-ai (visited on 06/16/2025).

Kluyver, Thomas, Benjain Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team (2016). "Jupyter Notebooks—a publishing format for reproducible computational workflows". In: *IOS Press*. IOS Press, pp. 87–90. DOI: 10.3233/978-1-61499-649-1-87.

Kramer, Mark A. (1991). "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2, pp. 233–243. DOI: 10.1002/aic.690370209.

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.

Kruskal, Joseph B. (1964a). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis". In: *Psychometrika* 29.1, pp. 1–27. DOI: 10.1007/BF02289565.

— (1964b). "Nonmetric multidimensional scaling. A numerical method". In: *Psychometrika* 29.2, pp. 115–129. DOI: 10.1007/BF02289694.

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694.

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert (2015). "Numba. A llvm-based python jit compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. Association for Computing Machinery, pp. 1–6. ISBN: 978-1-4503-4005-2. DOI: 10.1145/2833157.2833162.

Lannelongue, Loïc, Jason Grealey, and Michael Inouye (2021). "Green Algorithms: Quantifying the Carbon Footprint of Computation". In: *Advanced Science* 8.12, 2100707. DOI: 10.1002/advs.202100707.

Leeuw, Jan de and Patrick Mair (2009). "Multidimensional Scaling Using Majorization. SMACOF in R". In: *Journal of Statistical Software* 31.3, pp. 1–30. DOI: 10.18637/jss.v031.i03.

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.

McKinney, Wes (2010). "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. SciPy, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

Pearson, Karl (1901). "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572. DOI: 10.1080/14786440109462720.

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, 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.

Spiwokv (2007). *Pysomap. Python Library for Isometric Feature Mapping (Isomap)*. URL: https://web.vscht.cz/~spiwokv/.

Standards, National Institute of and Technology (2024). *The EMNIST dataset*. URL: https://www.nist.gov/itl/products-and-services/emnist-dataset (visited on 05/30/2025).

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.

Torgerson, Warren S. (1952). "Multidimensional scaling. I. Theory and method". In: *Psychometrika* 17.4, pp. 401–419. DOI: 10.1007/BF02288916.

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). "How to Use t-SNE Effectively". In: *Distill*. DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.

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.