Fast Multidimensional Scaling on Big Geospatial Data Using Neural Networks

Ioannis Mademlis[1*], Georgios Voulgaris[1] and Ioannis Pitas[1]

[1*]Department of Informatics, Aristotle University of Thessaloniki, AUTH Campus, Thessaloniki, GR-54124, Greece.

*Corresponding author(s). E-mail(s): imademlis@csd.auth.gr;
Contributing authors: voulgeorgios@gmail.com;
pitas@csd.auth.gr;

Abstract

This paper presents a fast approximation method for Multidimensional Scaling (MDS)-based dimensionality reduction on large cartography datasets. Since MDS preserves data point distances, it is useful in application domains where geolocation data are critical. Typical relevant tasks include smartphone user behavioral pattern extraction, animal motion tracking over long distances, or distributed sensor data monitoring. The input to MDS is a data distance matrix employed for reducing data point dimensionality under distance constraints. Similar procedures are crucial for analyzing and revealing the original hidden data structure, as well as for data visualization, feature extraction, or compression. For $\boldsymbol{N}$ data points, MDS has a computational complexity that exceeds $\boldsymbol{O(N^2)}$ which, may be excessive for a large $\boldsymbol{N}$, e.g., for several hundred thousands or millions of data points. The proposed method allows fast approximate MDS calculation on million-point datasets in less than a minute on a simple laptop, by sampling a small subset of the original dataset, performing regular MDS on it and training a neural regressor to learn the desired MDS mapping. Quantitative and qualitative empirical evaluation of the proposed fast MLP-MDS algorithm on a geospatial data mapping task, i.e., on reducing 3D Earth surface points (longitude, latitude, radius) to 2D maps, has resulted in promising findings and small approximation errors. The benefits are even greater in incremental settings, where new data points are obtained and projected over time. Unlike regular MDS or competing approximations, this is trivially supported in MLP-MDS due to the latter's model-based nature.

Keywords: Multidimensional Scaling, Approximate MDS, Incremental MDS, Big Data, MultiLayer Perceptron, Geospatial Mapping

1 Introduction

Multidimensional Scaling (MDS) is a classical method for data dimensionality reduction [1], typically with the purpose of visualizing and/or pre-processing the dataset for further analysis. Such algorithms may be employed either for preliminary feature extraction in several machine learning tasks, and/or as tools for visualizing the data in 1, 2 or 3 dimensions [2] [3] [4]. MDS performs this while attempting to preserve (not necessarily metric) dissimilarities between data points.

Earth surface map making, or cartography, is essentially a dimensionality reduction task, where the goal is to properly project a set of Earth surface points from 3 to 2 dimensions. Its applications are obviously innumerable, ranging from urban/regional studies [5] to ecology [6], demography [7] and robotics [8] [9] [10] [11]. Mathematically, there is no error-free method for projecting Earth 3D surface on a 2D map plane. Gauss' Theorema Egregium [12] states that a sphere and a plane are not isometric, even locally. This fact implies that no planar (flat) map of Earth can be perfect, not even for a portion of its surface. Thus, every cartographic projection necessarily distorts distances of geographical locations. In fact, the longer the distance of two Earth surface points, the higher the distance error on the 2D projection plane. Since MDS explicitly tries to preserve data point dissimilarities, it is in theory well-suited to geospatial data mapping tasks where geolocation data are crucial (e.g., smartphone user behavioral pattern extraction, animal motion tracking over long distances, or Internet-of-Things sensor data monitoring).

Given a dataset $\mathcal{X}$ of N P-dimensional data points, MDS is tasked to map all N points to a different set $\hat{\mathcal{X}}$ of N p-dimensional points, where typically $p \ll P$. MDS input is a $N \times N$ dissimilarity matrix, constructed using any distance function, while in case $p = 1, 2$ or 3 the output is a data scatterplot that can be used for visualization purposes. Different dissimilarity and similarity (or proximity) measures can be defined for any pair of data points in a dataset. The type of the distance function employed for constructing the input dissimilarity matrix determines the required MDS variant, like *Classical MDS*, *Metric MDS* or *Non-metric MDS*.

The aim of Classical and Metric MDS is to reduce data dimensionality while preserving data point distances [1] [13]. Given N data points $\mathbf{x}_i \in \mathbb{R}^P, \forall i \in \{1, ..., N\}$, and a scalar distance function $d(\mathbf{x}_i, \mathbf{x}_j)$, all pairwise dissimilarities between data points can be stored in data distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$, where entry $d_{ij} = \| \mathbf{x}_i - \mathbf{x}_j \|_d, \forall i, j \in \{1, ..., N\}$. MDS maps each $\mathbf{x}_i$ to a vector $\hat{\mathbf{x}}_i \in \mathbb{R}^p, p < P, \forall i \in \{1, ..., N\}$, so that $d(\mathbf{x}_i, \mathbf{x}_j) \approx \hat{d}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$, where $\hat{d}$ is Euclidean distance. When distance d is also Euclidean, Classical MDS is

preferred as it has an exact algebraic solution based on eigendecomposition. For non-Euclidean distances, Classical MDS may still be used at a significant penalty in solution robustness. Thus, typically, Metric MDS is employed in its place. Metric MDS is an iterative algorithm that tries to minimize a non-linear, non-convex objective function. Despite its flexibility, it is susceptible to being trapped in local minima due to the nature of non-convex optimization methods. Non-metric MDS is out of the scope of this paper, as in this case dissimilarities are known only by their rank order and the exact distance between successively ranked dissimilarities is of no interest or is unavailable.

Classical MDS uses a double centering operator and Singular Value Decomposition (SVD), so its computational complexity is $O(N^3)$. For Metric MDS, its time complexity is that of the employed optimization strategy. SMACOF ("Scaling by MAjorizing a COmplicated Function") is typically used in practical implementations, having a time and space complexity of $O(N^2)$ [14] [15]. More generally, since all MDS variants take a dissimilarity matrix as input, its computational complexity is independent of data point dimensionality, but depends on dataset size: it is safe to say that it always equals or exceeds $O(N^2)$ for N data points (i.e., the computational complexity of calculating a $N \times N$ distance matrix).

As a result, current MDS algorithms are notoriously slow and their applicability is limited to small datasets. Efficiently solving large-scale MDS problems arising in numerous applications has been a long-time challenge. In the geospatial mapping case the issue is especially pronounced, since the number of Earth landmarks to be mapped can be in the range of millions. Thus, an approximate method is the only practical approach.

In this paper, we propose a novel method for approximating MDS on large geospatial datasets of N data points, where N is in the range of millions. The novel *MLP-MDS* method, relies on training a MultiLayer Perceptron (MLP) neural network for solving a regression problem with relatively few data points and, subsequently, employing the trained model for reducing the dimensionality of the entire dataset. MLP was selected as it is one of the most simple and traditional neural network architectures that is a universal function approximator. Thus, as MDS is essentially a mapping $\hat{\mathbf{x}} = f(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^P$ and $\hat{\mathbf{x}} \in \mathbb{R}^p$, MLP learns to regress this function using M regression training pairs $\{\mathbf{x}_i, \hat{\mathbf{x}_i}\}, i \in \{1, ..., M\}, M \ll N$, where $\hat{\mathbf{x}_i}$ have been pre-obtained by performing Metric MDS on M data points randomly sampled from the original dataset. Since M is a relatively small integer, regular Metric MDS may be performed without computational/memory issues. The benefits of the MLP-MDS method are even greater in incremental dimensionality reduction settings, where new data points are obtained and projected over time. Unlike regular MDS or competing approximations, this can be trivially supported by MLP-MDS. Its empirical evaluation demonstrates good dimensionality reduction results on a geospatial mapping task.

2 Related Work

Data dimensionality reduction methods relying on neural networks are not a novelty [16], including multidimensional projection algorithms exploiting MLPs. However, all existing approaches either introduce non-MDS approximation methods that don't maintain the original distance of data-points (not even up to scaling) in the projected visual dimension, or implement data transformation pipelines [17] [18] [19] that are not applicable on big data due to large memory and computation constraints. As shown later in this paper, many of these methods work by sampling initial control points from the original dataset and, subsequently, computing their distance or similarity matrix. In large-scale datasets with millions of data points, the overall process is still very demanding in computational and memory resources.

Applying Metric MDS on very large datasets is known to be difficult, due to memory and computational requirements [20]. As a result, several non-neural MDS approximations have been developed for use in similar cases [21]: FastMap, MetricMap and Landmark MDS (LMDS) [20], all try to reduce computational complexity by calculating MDS on one or more smaller distance matrices and then finding approximate solutions on the remaining data (split-and-combine methods) [13] [20]. For example, on Landmark MDS, a smaller set of landmark points is randomly sampled from the full dataset and Metric MDS is performed on them. The remaining data are then approximately projected on the low-dimensional space, since they have known distances from the MDS-employed landmark points. FastMap MDS is similar to Landmark MDS, but using only 2 landmarks: the ones with the greatest distance. Although faster than Classical MDS, these methods cannot be readily applied on real-world large datasets due to unacceptably high approximation errors [22]. More landmarks can be used in order to increase accuracy, but overall runtime/memory requirements are increased as well. Thus, simply parallelizing MDS is typically preferred, by distributing distance matrix blocks on multiple machines. Even then, applying MDS on a dataset of 100000 data points needs a tremendous amount of resources to complete, as shown in Figure 1: at least a 64-bit memory space, 480 GB of RAM and more than 2^{11} seconds of computing time, using 256 CPU cores [15].

3 MLP-MDS Method

Given an input dataset $\mathcal{X}$, a distance matrix $\mathbf{D}$ defined on $\mathcal{X}$ and a monotonic function $f(d_{ij}) = \alpha + \beta d_{ij}$, Metric MDS attempts to find an optimal output configuration $\hat{\mathcal{X}}$ so that $\hat{d_{ij}} = \parallel \hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j \parallel_2 \approx f(d_{ij})$. The so-called *stress function* $\mathcal{L}\left(\hat{\mathcal{X}}\right)$ defines the robustness of the solution:

$$\mathcal{L}\left(\hat{\mathcal{X}}\right) = \left(\frac{\sum_i \sum_j \left(\hat{d_{ij}} - f(d_{ij}) \right)^2}{\parallel \mathbf{D} \parallel_F} \right)^{1/2} . \tag{1}$$

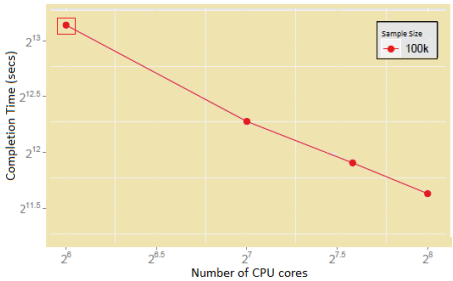


Fig. 1 Required parallel MDS running time versus number of CPU cores, in an example dataset of 100000 data points [15].

Metric MDS is an iterative algorithm that tries to find the set $\hat{\mathcal{X}}$ that minimizes $\mathcal{L}\left(\hat{\mathcal{X}}\right)$ over all α, β. As this process involves minimizing a non-linear non-convex objective function based on $O(N^2)$ distances, its computational complexity is at least $O(N^2)$. Thus, its application on big data is theoretically challenging and practically impossible [14] [15].

To overcome this problem we propose a novel algorithm, called MLP-MDS, where we learn to approximate the mapping g from $\mathcal{X}$ to $\hat{\mathcal{X}}$ using regression and a small subset of $\mathcal{X}$.

We assume an integer $M \ll N$, a training subset $\tilde{\mathcal{X}}$ of $\mathcal{X}$ composed of M data points $\tilde{\mathbf{x}}_i \in \mathbb{R}^P, i \in \{1, ..., M\}$ sampled in a systematic manner from $\mathcal{X}$ (sampling details can be found in Subsection 4.1), Metric MDS as a function g and projections $\mathbf{y}_i = g(\tilde{\mathbf{x}}_i), \mathbf{y} \in \mathbb{R}^p$. We employ the projections $\mathbf{y}_i$ of all $\tilde{\mathbf{x}}_i$ on the low-dimensional space $\mathbb{R}^p$ to train a regression model that approximately performs the mapping g:

$$\mathbf{y} = \boldsymbol{g}(\mathbf{x};\ \boldsymbol{\varphi}), \tag{2}$$

for any $\mathbf{x} \in \mathbb{R}^P$, where $\mathbf{y} \in \mathbb{R}^p$ and $\boldsymbol{\varphi}$ are the learnable MLP model parameters.

Given the above definition, MLP-MDS is defined only by the choice of a MultiLayer Perceptron (MLP) as the regression model. The MLP is trained by solving the following optimization problem:

$$\min_{\boldsymbol{\varphi}} \sum_{i=1}^{M} J(\mathbf{y}_i, \hat{\mathbf{y}}_i), \tag{3}$$

where J depends on the selected loss function and $\hat{\mathbf{y}}_i \in \mathbb{R}^p$ is the model's output prediction vector for input data point $\tilde{\mathbf{x}}_i$. Actual training details (e.g., optimization method, regularization, loss function, hyperparameters, etc.) can be adjusted according to the task.

After training, $\hat{\mathbf{y}}_i \approx \mathbf{y}_i$ but, in general, $\mathbf{y}_i = g(\tilde{\mathbf{x}}_i) \neq \hat{\mathbf{x}}_i$. That is, the projection of a data point when MDS is performed on the entire input dataset $\mathcal{X}$ is different than when MDS is performed on the sample $\tilde{\mathcal{X}}$. MLP-MDS relies on the hypothesis that this difference is negligible for practical purposes, assuming an effective sampling strategy is employed when constructing $\tilde{\mathcal{X}}$, instead of naive uniform selection.

Thus, in the proposed method, data point clustering with $M \ll N$ clusters is exploited for deriving $\tilde{\mathcal{X}}$. After clustering is completed, the cluster medoids are employed for performing MDS and, thus, constructing the M input/output $(\tilde{\mathbf{x}}_i/\mathbf{y}_i)$ training pairs for regression. Note that clustering is orders of magnitude more efficient than Metric MDS, especially with regard to memory requirements.

After the MLP model has been trained, it can be employed directly for finding the low-dimensional projection of any $\mathbf{x} \in \mathbb{R}^P$ (approximately according to mapping g). This process may be performed for all data points of $\mathcal{X}$ that were not included in $\tilde{\mathcal{X}}$, but also for any other P-dimensional vector. Note that this is not possible with regular MDS, which outputs a fixed set of N projected data points, while even if enough memory were available for applying the full Metric MDS algorithm on the entire set $\mathcal{X}$, MLP inference on it would still be several orders of magnitude faster.

4 MLP-MDS Perfomance Evaluation

To evaluate the proposed MLP-MDS method, an Earth surface map making dataset was employed. Since cartography is a dimensionality reduction problem, where $P = 3$ and $p = 2$, it can be approached using MDS. The employed dataset consists of $N = 4708904$ geographic landmarks, with the following attributes provided per landmark: latitude, longitude, continent, name, type, population (if valid) and altitude. However, out of these, only three numerical attributes ("latitude", "longitude", "altitude") are needed for MLP-MDS evaluation purposes, while "continent" was employed in post-processing solely for better visualising the MLP-MDS results. We will refer to them as data points because of the special meaning of landmarks in Landmark-MDS.

Regular Metric MDS on a small sample of 60000 Earth data points requires TBs of memory, significant computational time and a cluster of machines to complete. Applying regular Metric MDS on 4 million Earth data points is practically impossible, making approximate methods a necessity.

4.1 Evaluation Methodology

MLP-MDS was implemented using Python and scikit-learn library and was evaluated on a HP workstation of dual 24-core Xeon processors, 256 GB RAM and 840 GBs swap space, running Ubuntu Linux. DBScan clustering [23] was selected as the sampling mechanism of MLP-MDS for creating $\tilde{\mathcal{X}}$, since typically interesting landmarks are unequally distributed on Earth surface.

DBScan is a density-based clustering algorithm that proceeds by first determining how many data points are within "reachable" distance *eps* from each point in the given dataset. Based on this information and on a second hyperparameter defining the minimum required number of points, each data point is designated as either noise or as part of a cluster; overlapping clusters are handled by appropriate merging. Essentially, the vector space is partitioned into various dense areas (i.e., clusters) and sparse areas. In this paper, DBScan

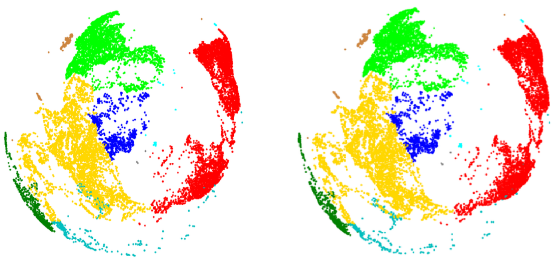
was executed iteratively with a Haversine distance metric, until the desired set of data points was extracted. Initial *eps* was set to $7km\ per\ radian$, but it was gradually increased until reaching $14km\ per\ radian$ in the final iteration. This process resulted in a balanced small sample set of $M = 66863$ data points. It is essential for MLP-based regression to include extreme original landmarks, as this leads to better distance normalization.

After obtaining the training set $\tilde{\mathcal{X}}$, the 3D normalized Cartesian and the polar coordinates of its data points were employed to derive the Euclidean and the haversine [24] $M \times M$ distance matrices, i.e., $\mathbf{D}_E$ and $\mathbf{D}_H$, respectively. Subsequently, we separately apply regular Metric MDS with $p = 2$ on these matrices. Below, Metric MDS output on $\mathbf{D}_E/\mathbf{D}_H$ is referred to as the cardinality-M set $\mathcal{Y}_E/\mathcal{Y}_H$, respectively. Each element of these two sets is a p-dimensional vector; thus, combining $\tilde{\mathcal{X}}$ as input and either $\mathcal{Y}_E$ or $\mathcal{Y}_H$ as regression targets/labels, provides us with two different MLP training/testing datasets: $\mathcal{T}_E$ and $\mathcal{T}_H$, correspondingly. Note that input vectors from $\tilde{\mathcal{X}}$ are expressed in 3D Euclidean/polar coordinates for $\mathcal{T}_E/\mathcal{T}_H$, respectively.

4.2 MLP Network Training

For MLP-MDS evaluation purposes, sets $\mathcal{T}_E/\mathcal{T}_H$ (both containing $M = 66863$ paired input/target vectors $\tilde{\mathbf{x}} \in \mathbb{R}^3/\mathbf{y} \in \mathbb{R}^2$) were partitioned into two subsets: $\mathcal{T}_E^1/\mathcal{T}_H^1$ for MLP training and $\mathcal{T}_E^2/\mathcal{T}_H^2$ for MLP testing, according to the 80/20 rule of thumb. Thus, training/test sets included 53491/13372 labelled data points, respectively.

In order to evaluate the degree to which the amount of training data affects method performance, the training sets were further split into two subsets each: $\mathcal{T}_E^{1A}/\mathcal{T}_H^{1A}$ and $\mathcal{T}_E^{1B}/\mathcal{T}_H^{1B}$. The former/latter ones contain 10693/42793 data points, respectively.



a) b)

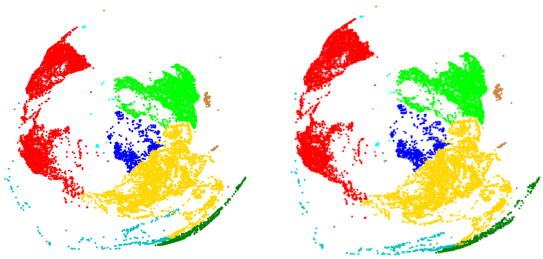
Fig. 2 2D map output of applying MLP-MDS on the test set $\mathcal{T}_E^2$, after training on: a) $\mathcal{T}_E^{1A}$ (left), or b) $\mathcal{T}_E^{1B}$ (right).

A simple, shallow MLP architecture was employed, with two hidden layers of 300 nodes each and ReLU activation functions. Training was performed with the ADAM optimizer [25], using a learning rate of 0.001. Four different MLP

models were trained for 200 iterations each, using $\mathcal{T}_E^{1A}$, $\mathcal{T}_H^{1A}$, $\mathcal{T}_E^{1B}$ and $\mathcal{T}_H^{1B}$ as the corresponding training sets.

4.3 Evaluation Results

After model training was successfully completed using error back-propagation and Stochastic Gradient Descent, MLP-MDS performance was evaluated by obtaining predictions on the test sets, separately for each of the four trained models. Subsequently, the Root Mean Square error (RMS) between the distance matrix of the predictions and that of regular Metric MDS output (taking only testing data points into account) was computed ("RMSE-DM"). The goal was to measure the degree to which the pairwise distances of the original input data are retained after projection, i.e., to quantify how good MLP-MDS is in approximating the operation of regular Metric MDS. Additionally, RMS error was measured between the MLP-predicted projected data point coordinates and the corresponding regular Metric MDS outputs, so as to evaluate the regression component of the proposed method ("RMSE-D"). All results are depicted in Table 1, where the reported running times are total inference/prediction times for the entire test set of 13372 data points. Relevant visualizations of the results are depicted in Fig. 2 and Fig. 3.



a) b)

Fig. 3 2D map output of applying MLP-MDS on the test set $\mathcal{T}_H^2$, after training on: a) $\mathcal{T}_H^{1A}$ (left), or b) $\mathcal{T}_H^{1B}$ (right).

Table 1 MLP-MDS prediction RMS errors (in km) and running time for test set (cardinality of 13372 data points).

Training Set	*Distance*	*RMSE-D*	*RMSE-DM*	*Time*
$\mathcal{T}_H^{1A}$	Haversine	0.03587	0.00153	0.0012 min
$\mathcal{T}_E^{1A}$	Euclidean	0.03527	0.00134	0.0012 min
$\mathcal{T}_H^{1B}$	Haversine	0.03668	0.00139	0.0010 min
$\mathcal{T}_E^{1B}$	Euclidean	0.03304	0.00108	0.0010 min

Moreover, the trained MLP models were employed to approximate MDS on the original set $\mathcal{X}$ of $N = 4708904$ data points. Applying regular MDS in

Fig. 4 A visualization of the cardinality-M set $\mathcal{Y}_H$, where color denotes continent and $M = 66863$. As this type of projection tries to preserve the original haversine distance, the resulting 2D map appears very different from the widely used Mercator map. Distortion due to Earth curvature is strong on regions near the poles.

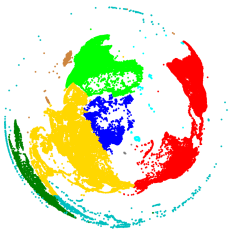
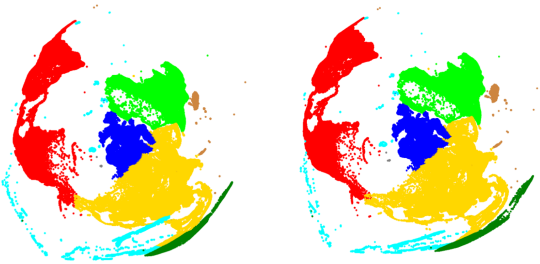


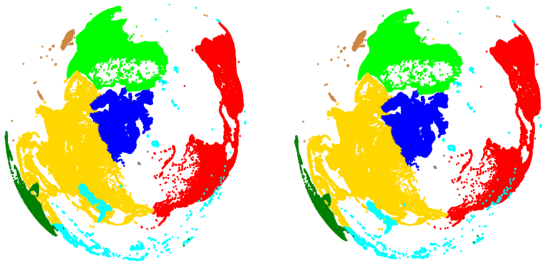
Fig. 5 A visualization of the cardinality-M set $\mathcal{Y}_E$, where color denotes continent and $M = 66863$. By visually inspecting the continent shapes, it becomes evident that regular Metric MDS using Euclidean distances correctly preserves distances. However, by comparing with the projection depicted in Fig. 4 where haversine distances were employed, flipping and rotation differences become obvious.

this case is not practically feasible, thus ground-truth cannot be obtained for quantitative comparison purposes. Therefore, we validated the results qualitatively by visually comparing the output 2D map produced by the trained MLP models with the ones derived when applying regular Metric MDS only on the sample $\tilde{\mathcal{X}}$ of $M = 66863$ data points. By exploiting continent-based coloring of the projected landmarks, as well as by assuming that a proper MDS approximation would project all identically-colored data points of $\mathcal{X}$ to the same 2D map region as their neighbours from $\tilde{\mathcal{X}}$, we are able to verify through visual inspection that MLP-MDS projection of $\mathcal{X}$ is indeed correct. Figures 4,5 and 6,7 largely only differ with regard to landmark density, with no colour diffusion into multiple continents being noticeable in the latter ones.



a) b)

Fig. 6 2D map output of applying MLP-MDS on the entire set $\mathcal{X}$, after training on: a) $\mathcal{T}_H^{1A}$ (left), or b) $\mathcal{T}_H^{1B}$ (right). Visually comparable to Fig. 4.



a) b)

Fig. 7 2D map output of applying MLP-MDS on the entire set $\mathcal{X}$, after training on: a) $\mathcal{T}_E^{1A}$ (left), or b) $\mathcal{T}_E^{1B}$ (right). Visually comparable to Fig. 5.

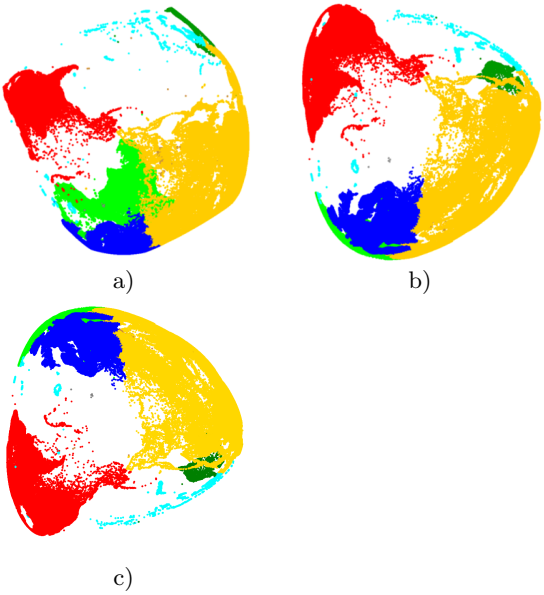


Fig. 8 Output 2D map of Landmark MDS applied on $N = 4708904$ data points, when using: a) 30 landmark points, b) 3000 landmark points or c) 10693 landmark points.

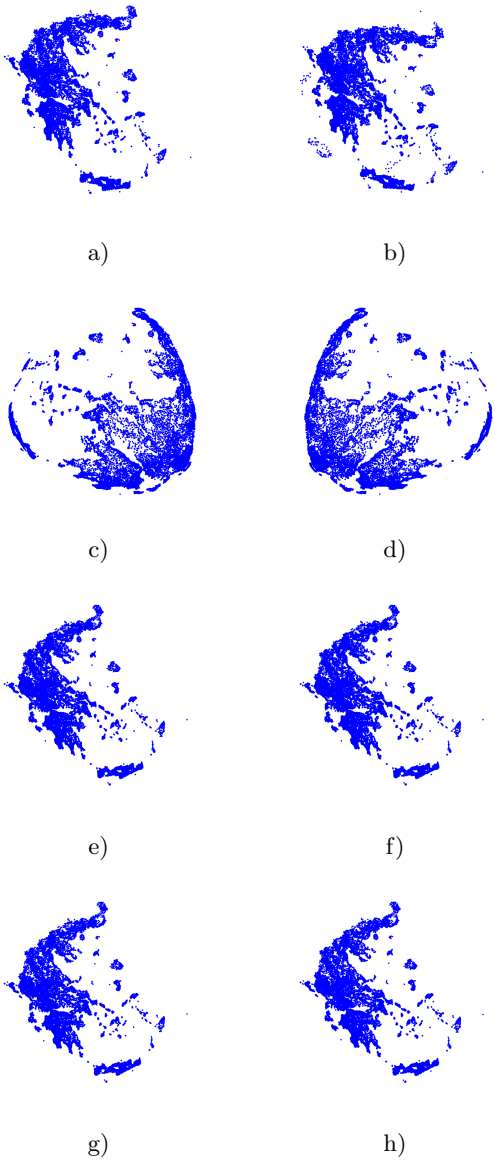


Fig. 9 Output 2D map of Greece using 14969 data points and: a) Greece Ground Truth, b) Metric MDS, c) Landmark MDS with 3000 landmarks, d) Landmark MDS with 10693 landmarks, e) MLP-MDS full after training with 10693 data points, f) MLP-MDS incr with pre-trained MLP on 42793 data points, g) MLP-MDS incr with pre-trained MLP on 10693 data points, h) MLP-MDS full after training with 3000 data points

Required runtime comparisons were not directly possible, since regular Metric MDS is not practically feasible on $N = 4708904$ data points. As a proxy, we measured Metric MDS on $M = 66863$ data points, i.e., on the sample

$\tilde{\mathcal{X}}$: it required 3111/3341 minutes to complete for haversine/Euclidean distance matrices, respectively. Memory-wise, it used 1 TB of combined computer memory, while on 200000 points it demanded over 5 TB.

On the other hand, MLP-MDS requires several orders of magnitude less runtime. Sampling 10693/42793 from 66863 data points using DBScan takes 2.18/2.54 minutes, respectively, performing regular Metric MDS on 10693/42793 data points requires 14.25/142 minutes, respectively, while training an MLP of the employed architecture on 10693/42793 input/output pairs requires 0.764/2.193 minutes for Euclidean distances, or 0.86/2.575 minutes for haversine distances, respectively. Finally, trained MLP inference on the remaining $N - M = 55900/N - M = 24070$ data points requires at most 0.005/0.002 minutes. Overall, MLP-MDS demands at worst 17.20/147.12 minutes to complete, for $M = 10693/M = 42793$, respectively. Given the results of Table 1, selecting the fastest option, i.e., $M = 10693$, does not incur any significant accuracy penalty.

Unlike regular MDS, MLP-MDS is practically feasible in the actual big data scenario where $N = 4708904$. Sampling $M = 66863$ data points out of all N via DBScan requires 67 minutes, while total subsequent runtime for obtaining $N - M$ predictions after training on $M = 42793$ data points is 2.56 minutes, including just 0.2443 minutes required for MLP inference over the entire dataset. Finally, MLP-MDS is trivially employable in the incremental MDS setting, where new data points are acquired and must be projected over time, by simply feed-forwarding each of them through the pre-trained MLP. Forward pass for a single data point requires only 0.00032 seconds. In contrast, regular MDS must be fully performed from scratch for the entirety of the dataset each time a new input data point is acquired and must be projected.

Landmark MDS is a competing approximation method that does not support the incremental setting either, unlike MLP-MDS. Thus, for each new point added to the dataset, the entire process must be executed from scratch. Unlike regular MDS, however, it is applicable to big data scenarios. The results of applying it over the entire dataset $\mathcal{X}$ of $N = 4708904$ data points are visually depicted in Figure 8. Landmark MDS completed the task in 12 minutes with a set of 30 landmarks, in 71 minutes with a set of 3000 landmarks and in 1320 minutes with a set of 10693 landmarks. The quality of the projection improves as the number of landmarks increases, but so does the computational time and memory complexity. In contrast, if a pre-trained MLP is available, MLP-MDS requires only 0.2443 minutes for simple MLP inference across all N data points in $\mathcal{X}$, since it naturally supports the incremental setting. In the non-incremental setting, where no MLP has been pre-trained (thus, MDS has to be performed in a sample and an MLP model must be constructed), MLP-MDS is still overall faster than Landmark MDS, since the entire method requires 33.50 minutes, including clustering, applying regular MDS and training an MLP on 10693 data points.

To achieve a quantitative comparison regarding projection quality between Landmark MDS and MLP-MDS, we employed a similar but smaller dataset

consisting of 14963 data points belonging to a European country (Greece). The results are visually depicted in Figure 9. Due to the small size region, haversine and Euclidean distances are approximately equal. We employed as ground-truth the real Euclidean distances between all original 14963 data points. RMSE between the distance matrix of the approximately projected data points and the ground-truth one was utilized for quantitative evaluation. Results are presented in Table 2. Note that we included four versions of MLP-MDS: two incremental ones (denoted as "MLP-MDS incr") and two full ones (denoted as "MLP-MDS full"). The former ones assume that a good regression model has been pre-constructed using a different dataset (trained on 10693/42793 data points), while it is now available for direct use with the current dataset. Thus, we showcase a significant advantage of the proposed method, compared to traditional MDS approximations which do not inherently support the incremental setting. The latter ones consist in non-incrementally applying MLP-MDS from scratch, performing Metric MDS on a sample of 3000/10693 data points (selected from the 14963 data points of Greece) and training MLP regressors using the corresponding output.

All of the reported errors/accuracies in Table 2 are averaged over the entire Greece detaset of all 14963 data points. Note that in the MLP-MDS incr cases, although there is zero overlap between this dataset and the dataset used for training the regressors, the accuracy is still higher than that of Landmark MDS. Also, while the most accurate MLP-MDS incr model was successfully pre-trained on 42793 data points without significant hustle, we were unable to use more than about 11000 landmarks with Landmark MDS (which could increase its accuracy) due to its unacceptably high memory requirements.

Metric MDS is solved using the numerical Sammon Maping [26] or the SMACOF[14] method. Note that stochasticity both in these solvers (the starting point, for instance) and in the MLP-MDS training process (e.g., training set sampling) is most likely the trivial reason behind the seemingly strange effect of approximate MLP-MDS appearing to have marginally better accuracy compared to regular Metric MDS (lower RMSE with respect to ground-truth), as shown in Table 2.

Table 2 Output accuracy and computation time of projecting Greece dataset (cardinality of 14963 data points), using Euclidean distances. Lower RMSE is better.

Method	***RMSE(km)***	***Time***
Metric MDS	0.00045	14.73 min
Landmark MDS with 3000 landmarks	0.02955	0.1120 min
Landmark MDS with 10693 landmarks	0.02887	2.50 min
MLP-MDS incr with 42793 data points	0.000057	0.0014 min
MLP-MDS incr with 10693 data points	0.000101	0.00095 min
MLP-MDS full with 10693 data points	0.000116	5.18 min
MLP-MDS full with 3000 data points	0.000211	0.811 min

5 Conclusions

Multidimensional Scaling (MDS) is a well-known approach for dimensionality reduction and dataset visualization, that tries to preserve data points distances during data projection and, thus, is well-suited to geospatial mapping applications. However, its applicability to big data scenarios and/or to incremental settings is limited due to very high computational/memory requirements. Existing approximation algorithms attempt to extend its use to large datasets at an accuracy penalty, but do not inherently support incremental MDS. The proposed fast MLP-MDS method addresses these challenges by constructing a regression model that approximately learns the mapping performed by MDS using a small sample of the complete dataset and, subsequently, may be used for projecting any data point with minimum fuss, negligible computational/memory cost and small approximation errors. Empirical evaluation on a cartography/geospatial mapping dataset demonstrated the high performance and robustness of the proposed method, as well as its convenience in incremental settings.

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951911 (AI4Media).

Statements and Declarations

5.1 Funding

This study was funded by European Union's Horizon 2020 research and innovation programme under grant agreement No 951911 (AI4Media).

5.2 Competing interests

The authors declare they have no financial or non-financial interests.

5.3 Informed consent

The research presented in this paper did not involve human or animal participants.

5.4 Data availability

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

5.5 Author's contribution

IM and IP developed the algorithm and designed the evaluation process. IM wrote the manuscript and submitted the paper. GV implemented the code, ran the experiments and prepared the figures, as part of his M.Sc. thesis. All authors reviewed and approved the manuscript.

References

[1] Pohlheim, H.: Multidimensional scaling for evolutionary algorithms: visualization of the path through search space and solution space using Sammon Mapping. Artificial Life **12**, 203–209 (2006)

[2] Spathis, D., Passalis, N., Tefas, A.: Fast, visual and interactive semi-supervised dimensionality reduction. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2018)

[3] Spathis, D., Passalis, N., Tefas, A.: Interactive dimensionality reduction using similarity projections. Knowledge-Based Systems **165**, 77–91 (2019)

[4] Passalis, N., Tefas, A.: PySEF: A python library for similarity-based dimensionality reduction. Knowledge-Based Systems **152**, 186–187 (2018)

[5] Mahabir, R., Croitoru, A., Crooks, A.T., Agouris, P., Stefanidis, A.: A critical review of high and very high-resolution remote sensing approaches for detecting and mapping slums: Trends, challenges and emerging opportunities. Urban Science **2**(1), 8 (2018)

[6] Capitani, C., Mukama, K., Mbilinyi, B., Malugu, I.O., Munishi, P.K.T., Burgess, N.D., Platts, P.J., Sallu, S.M., Marchant, R.: From local scenarios to national maps: a participatory framework for envisioning the future of tanzania. Ecology and Society **21**(3) (2016)

[7] Nusrat, S., Alam, M.J., Kobourov, S.: Evaluating cartogram effectiveness. IEEE Transactions on Visualization and Computer Graphics **24**(2), 1077–1090 (2016)

[8] Kakaletsis, E., Mademlis, I., Nikolaidis, N., Pitas, I.: Multiview vision-based human crowd localization for UAV fleet flight safety. Signal Processing: Image Communication **99**, 116484 (2021)

[9] Kakaletsis, E., Mademlis, I., Nikolaidis, N., Pitas, I.: Bayesian fusion of multiview human crowd detections for autonomous UAV fleet safety. In: Proceedings of the European Signal Processing Conference (EUSIPCO) (2021). IEEE

[10] Kakaletsis, E., Symeonidis, C., Tzelepi, M., Mademlis, I., Tefas, A., Nikolaidis, N., Pitas, I.: Computer vision for autonomous UAV flight safety:

An overview and a vision-based safe landing pipeline example. ACM Computing Surveys (CSUR) **54**(9), 1–37 (2021)

[11] Symeonidis, C., Kakaletsis, E., Mademlis, I., Nikolaidis, N., Tefas, A., Pitas, I.: Vision-based UAV safe landing exploiting lightweight Deep Neural Networks. In: Proceedings of the International Conference on Image and Graphics Processing (ICIGP) (2021)

[12] Pressley, A.: Gauss' theorema egregium. In: Elementary Differential Geometry, pp. 247–268. Springer, New York (2010)

[13] De Silva, V., Tenenbaum, J.B.: Sparse multidimensional scaling using landmark points. Stanford University. technical report (2004)

[14] Borg, I., Groenen, P.J.F. Springer (2005)

[15] Choi, J.Y., Bae, S.-H., Qiu, X., Fox, G.: High performance dimension reduction and visualization for large high-dimensional data analysis. In: Proceedings of the EEE/ACM International Conference on Cluster, Cloud and Grid Computing (2010)

[16] Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (2006)

[17] Espadoto, M., Hirata, N.S.T., Telea, A.C.: Deep learning multidimensional projections. Information Visualization **19**, 247–269 (2020)

[18] Joia, P., Coimbra, D., Cuminato, J.A., Paulovich, F.V., Nonato, L.G. IEEE Transactions on Visualization and Computer Graphics **17**, 2563–2571 (2011)

[19] Paulovich, F.V., Nonato, L.G., Minghim, R., Levkowitz, H.: Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. IEEE Transactions on Visualization and Computer Graphics **14**, 564–575 (2008)

[20] Platt, J.: FastMap, MetricMap, and Landmark MDS are all Nystrom algorithms. In: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS) (2005)

[21] Yang, L.: Locally multidimensional scaling for nonlinear dimensionality reduction. In: Proceedings of the International Conference on Pattern Recognition (ICPR) (2006)

[22] Kim, E., Lee, S., Kim, C., Kim, K.: Mobile beacon-based 3D-localization with multidimensional scaling in large sensor networks **14**, 647–649 (2010)

[23] Xu, X., Ester, M., Kriegel, H.-P., Sander, J.: A distribution-based clustering algorithm for mining in large spatial databases. In: Proceedings of the International Conference on Data Engineering (1998)

[24] Brummelen, G.V.: Heavenly Mathematics: the Forgotten Art of Spherical Trigonometry. Princeton University Press, ??? (2012)

[25] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (ICLR) (2015)

[26] Sammon, J.W.: A nonlinear mapping for data structure analysis. IEEE Transactions on Computers **100**, 401–409 (1969)