Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Procedia Computer Science 00 (2012) 1–10
Procedia Computer Science

Adaptive Interpolation of Multidimensional Scaling

Seung-Hee Bae, Judy Qiu, Geoffrey Fox

School of Informatics and Computing, Indiana University

Abstract

The recent explosion of publicly available biology gene sequences and chemical compounds offers an unprecedented opportunity for data mining. To make data analysis feasible for such vast volume and high-dimensional scientific data, we apply high performance dimension reduction algorithms. It facilitates the investigation of unknown structures in a three dimensional visualization. Among the known dimension reduction algorithms, we utilize the multidimensional scaling (MDS) algorithm to configure the given high-dimensional or abstract data into the target dimension. However, MDS algorithm requires large physical memory as well as computational resources. In order to reduce computational complexity and memory requirement effectively, the interpolation method of MDS was proposed in 2010. With minor trade-off of approximation, interpolation method makes it possible to process millions of data points with modest amounts of computation and memory requirement. In this paper, we would like to improve the mapping quality of the interpolation approach to MDS by adapting the the original dissimilarity based on the ratio between the original dissimilarity and the corresponding mapping distances. Our experimental results illustrate that the quality of interpolated mapping results are improved by adding the adaptation step without runtime loss compared to the original interpolation method. With the proposed adaptive interpolation method, we construct a better configuration of millions of *out-of-sample* data into the target dimension than the previous interpolation method.

Keywords: dimension reduction, multidimensional scaling, interpolation, adaptation

1. Introduction

Due to the advancements in science and technologies for last several decades, every scientific and technical fields generates a huge amount of data in every minute in the world. We are really in the data deluge era. In reflection of data deluge era, data-intensive scientific computing [1] has been emerging in the scientific computing fields and getting more interested by many people. To analyze those incredible amount of data, many data mining and machine learning algorithms have been developed. Among many data mining and machine learning algorithms that have been invented, we focus on dimension reduction algorithms, which reduce data dimensionality from original high dimension to target dimension, in this paper.

Among many dimension reduction algorithms, such as principle component analysis (PCA), generative topographic mapping (GTM) [2, 3], self-organizing map (SOM) [4], and multidimensional scaling (MDS) [5, 6], we discuss about MDS in this paper since it is popular and theoretically strong. The parallelization of MDS algorithm

Email address: sebae@cs.indiana.edu(Seung-Hee Bae), xqiu@indiana.edu(Judy Qiu), gcf@indiana.edu(Geoffrey Fox) (Geoffrey Fox)

was studied in [7] which aims to utilize multicore clusters and to increase the computational capability with minimal overhead for the purpose of investigating large data sets, such as 100,000 data points. However, parallelization of an MDS algorithm, whose computational complexity and memory requirement is upto $O(N^2)$ where N is the number of points, is still limited by the memory requirement for huge data, e.g. millions of points, although it utilize distributed memory environments, such as clusters, for acquiring more memory and computational resources. In this paper, we try to solve the memory-bound problem by interpolation based on pre-configured mappings of the sample data for MDS algorithm, so that we can provide configuration of millions points in the target space.

This paper is organized as follows. First, we briefly discuss about multidimensional scaling (MDS) in Section 2. The various existed methods of *out-of-sample* approach related to the MDS are explained in Section 3 and Section 4. Then, the proposed adaptive interpolation method is described in Section 5. The quality comparison between interpolated results and full MDS running results and runtime evaluation of those algorithms are shown in Section 6 followed by our conclusion in Section 7.

2. Multidimensional Scaling (MDS)

Multidimensional scaling(MDS) [5, 6] is a general term for the techniques of configuration of the given high dimensional data into target dimensional space based on the pairwise proximity information of the data, while each Euclidean distance between two points becomes as similar to the corresponding pairwise dissimilarity as possible. In other words, MDS is a non-linear optimization problem with respect to mapping in the target dimension and original proximity information.

Formally, the pairwise proximity information is given as an $N \times N$ matrix ($\boldsymbol{\Delta} = [\delta_{ij}]$), where N is the number of points and δ_{ij} is the given dissimilarity value of the original data space between point i and j. (1) Symmetric ($\delta_{ij} = \delta_{ji}$), (2) non-negative ($\delta_{ij} \geq 0$), and (3) zero diagonal ($\delta_{ii} = 0$) are the constraints of the dissimilarity matrix $\boldsymbol{\Delta}$. By MDS algorithm, the generated mapping could be also represented as an $N \times L$ matrix ($\boldsymbol{X}$), where L is the target dimension, and each data point $\boldsymbol{x}_i \in \mathbb{R}^L$ ($i = 1, \ldots, N$) resides in i-th rows of $\boldsymbol{X}$.

The evaluation of the constructed configuration is done with respect to the well-known objective functions of MDS, namely STRESS [8] or SSTRESS [9]. Below equations are the definition of STRESS (1) and SSTRESS (2):

$$\sigma(\boldsymbol{X}) = \sum_{i<j\leq N} w_{ij}(d_{ij}(\boldsymbol{X}) - \delta_{ij})^2 \tag{1}$$

$$\sigma^2(\boldsymbol{X}) = \sum_{i<j\leq N} w_{ij}[(d_{ij}(\boldsymbol{X}))^2 - (\delta_{ij})^2]^2 \tag{2}$$

where $1 \leq i < j \leq N$ and w_{ij} is a weight value, so $w_{ij} \geq 0$.

3. Related Work

Out-of-sample method, which embeds new points with respect to previously configured points, has been actively researched for recent years, aimed at improving the capability of dimension reduction algorithms by reducing the computational and memory-wide requirement with the trade-off of slightly approximated mapping result.

In sensor network localization field, when there are only a subset of pairwise distances between sensors and a subset of anchor locations are available, people try to find out the locations of the remaining sensors. For instance, semi-definite programming relaxation approaches and its extended approaches has been proposed to solve it [10]. [11] and [12] proposed out-of-sample extension for the classical multidimensional scaling (CMDS) [13], which is based on spectral decomposition of a symmetric positive semidefinite matrix (or the approximation of positive semidefinite matrix), and the embeddings in the configured space are represented in terms of eigenvalues and eigenvectors of it. [11] projected the new point $\boldsymbol{x}$ onto the principal components, and [12] extends the CMDS algorithm itself to the out-of-sample problem. In [12], the authors describe how to embed one point between the embeddings of the original n objects through modification of the original CMDS equations, which preserves the mappings of the original n objects, with $(n+1)\times(n+1)$ matrix $\boldsymbol{A}_2$ instead of $n\times n$ matrix $\boldsymbol{\Delta}_2$, and extends to embedding a number of points simultaneously by using matrix operations. Recently, a multilevel force-based MDS algorithm was proposed as well [14].

In contrast to applying out-of-sample problem to CMDS, out-of-sample approach to metric MDS with STRESS criteria of Eq. (1) was proposed by Bae et al. [15], which finds embeddings of approximating to the distance (or dissimilarity) rather than the inner product as in CMDS, with an gradient descent optimization method, called iterative majorizing. The details of the iterative majorizing interpolation approach for the MDS problem [15] is explained in Section 4.

4. Majorizing Interpolation MDS

One of the main limitation of most MDS applications is that it requires $O(N^2)$ memory as well as $O(N^2)$ computation. Thus, though it is possible to run them with small data size without any trouble, it is impossible to execute it with large number of data due to memory limitation, so it could be considered as memory-bound problem. For instance, Scaling by MAjorizing of COmplicated Function (SMACOF) [16, 17], a well-known MDS application via Expectation-Maximization (EM) [18] like approach, uses six $N \times N$ matrices. If $N = 100,000$, then one $N \times N$ matrix of 8-byte double-precision numbers requires 80 GB of main memory, so the algorithm needs to acquire at least 480 GB of memory to store six $N \times N$ matrices. It is possible to run parallel version of SMACOF with MPI on the testbed system in Table 1 with $N = 100,000$. If the data size is increased only twice, however, then SMACOF algorithm should have 1.92 TB of memory, which is bigger than total memory of the system in Table 1 (1.536 TB), so it is impossible to run it within the cluster. Increasing memory size will not be a solution, even though it could increase the runnable number of points. It will encounter the same problem as the data size increases.

To solve this obstacle, Bae et al. developed a simple interpolation approach based on pre-mapped MDS result of the sample of the given data [15]. The interpolation algorithm [15] is similar to k nearest neighbor (k-NN) classification [19], but it approximates new mapping position of the new point based on the positions of k-NN, among pre-mapped subset data, instead of classifying it. For the purpose of deciding new mapping position in relation to the k-NN positions, iterative majorization method is applied as similar as SMACOF [16, 17] algorithm. The algorithm proposed in [15] is called Majorizing Interpolation of MDS (hereafter *MI-MDS*), and the summary of MI-MDS is in this section as below.

The MI-MDS algorithm is implemented as follows. We are given N data in high-dimensional space, say D-dimension, and proximity information ($\boldsymbol{\Delta} = [\delta_{ij}]$) of those data as in Section 2. Among N data, the configuration of the n sample points in L-dimensional space, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^L$, called X, are already constructed by an MDS algorithm, here we use SMACOF algorithm. Then, we select k nearest neighbors ($\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k \in \boldsymbol{P}$) of the given new point, among n pre-mapped points with respect to corresponding δ_{ix}, where $\boldsymbol{x}$ represents the new point. Linear search is used to find k-nearest neighbors among n-sampled data, so that the complexity of finding k-nearest neighbors is $O(n)$ per one interpolated point (here $\boldsymbol{x}$). Finally, the new mapping of the given new point $\boldsymbol{x} \in \mathbb{R}^L$ is calculated based on the pre-mapped position of selected k-NN and corresponding proximity information δ_{ix}. The finding new mapping position is considered as a minimization problem of STRESS (3) as similar as normal MDS problem with m points, where $m = k + 1$. However, only one point ($\boldsymbol{x}$) is movable among m points, so we can simplify STRESS equation (3) as belows (Eq. (4)), and we set $w_{ij} = 1$, for $\forall i, j$ in order to simplify.

$$\sigma(\boldsymbol{X}) = \sum_{i<j\leq m} (d_{ij}(\boldsymbol{X}) - \delta_{ij})^2 \tag{3}$$

$$= C + \sum_{i=1}^{k} d_{ix}^2 - 2\sum_{i=1}^{k} \delta_{ix} d_{ix} \tag{4}$$

where δ_{ix} is the original dissimilarity value between $\boldsymbol{p}_i$ and x, d_{ix} is the Euclidean distance of mappings in L-dimension between $\boldsymbol{p}_i$ and $\boldsymbol{x}$, and C is constant part. The second term of Eq. (4) can be deployed as following:

$$\sum_{i=1}^{k} d_{ix}^2 = \|\boldsymbol{x} - \boldsymbol{p}_1\|^2 + \cdots + \|\boldsymbol{x} - \boldsymbol{p}_k\|^2 \tag{5}$$

$$= k\|\boldsymbol{x}\|^2 + \sum_{i=1}^{k} \|\boldsymbol{p}_i\|^2 - 2\boldsymbol{x}^t\boldsymbol{q} \tag{6}$$

where $\boldsymbol{q}^t = (\sum_{i=1}^{k} p_{i1}, \ldots, \sum_{i=1}^{k} p_{iL})$ and p_{ij} represents j-th element of $\boldsymbol{p}_i$. In order to establish majorizing inequality, we apply *Cauchy-Schwarz* inequality to $-d_{ix}$ of the third term of Eq. (4). Please, refer to chapter 8 in [6] for details of how to apply *Cauchy-Schwarz* inequality to $-d_{ix}$. Since $d_{ix} = \|\boldsymbol{p}_i - \boldsymbol{x}\|$, $-d_{ix}$ could have following inequality based on *Cauchy-Schwarz* inequality:

$$-d_{ix} \leq \frac{\sum_{a=1}^{L}(p_{ia} - x_a)(p_{ia} - z_a)}{d_{iz}} \tag{7}$$

$$= \frac{(\boldsymbol{p}_i - \boldsymbol{x})^t(\boldsymbol{p}_i - \boldsymbol{z})}{d_{iz}} \tag{8}$$

where $\boldsymbol{z}^t = (z_i, \ldots, z_L)$ and $d_{iz} = \|\boldsymbol{p}_i - \boldsymbol{z}\|$. The equality in Eq. (7) occurs if $\boldsymbol{x}$ and $\boldsymbol{z}$ are equal. If Eq. (8) is applied to the third term of Eq. (4), then we obtain

$$-\sum_{i=1}^{k} \delta_{ix} d_{ix} \leq -\sum_{i=1}^{k} \frac{\delta_{ix}}{d_{iz}} (\boldsymbol{p}_i - \boldsymbol{x})^t (\boldsymbol{p}_i - \boldsymbol{z}) \tag{9}$$

$$= -\boldsymbol{x}^t \sum_{i=1}^{k} \frac{\delta_{ix}}{d_{iz}} (\boldsymbol{z} - \boldsymbol{p}_i) + C_\rho \tag{10}$$

where C_ρ is a constant. If Eq. (6) and Eq. (10) are applied to Eq. (4), then it could be like following:

$$\sigma(\boldsymbol{X}) = C + \sum_{i=1}^{k} d_{ix}^2 - 2\sum_{i=1}^{k} \delta_{ix} d_{ix} \tag{11}$$

$$\leq C + k\|\boldsymbol{x}\|^2 - 2\boldsymbol{x}^t\boldsymbol{q} + \sum_{i=1}^{k} \|\boldsymbol{p}_i\|^2 - 2\boldsymbol{x}^t \sum_{i=1}^{k} \frac{\delta_{ix}}{d_{iz}} (\boldsymbol{z} - \boldsymbol{p}_i) + C_\rho \tag{12}$$

$$= \tau(\boldsymbol{x}, \boldsymbol{z}) \tag{13}$$

where both C and C_ρ are constants. In the Eq. (13), $\tau(\boldsymbol{x}, \boldsymbol{z})$, a quadratic function of $\boldsymbol{x}$, is a majorization function of the STRESS. Through setting the derivative of $\tau(\boldsymbol{x}, \boldsymbol{z})$ equal to zero, we can obtain minimum of it; that is

$$\nabla\tau(\boldsymbol{x}, \boldsymbol{z}) = 2k\boldsymbol{x} - 2\boldsymbol{q} - 2\sum_{i=1}^{k} \frac{\delta_{ix}}{d_{iz}} (\boldsymbol{z} - \boldsymbol{p}_i) = 0 \tag{14}$$

$$\boldsymbol{x} = \frac{\boldsymbol{q} + \sum_{i=1}^{k} \frac{\delta_{ix}}{d_{iz}} (\boldsymbol{z} - \boldsymbol{p}_i)}{k} \tag{15}$$

where $\boldsymbol{q}^t = (\sum_{i=1}^{k} p_{i1}, \ldots, \sum_{i=1}^{k} p_{iL})$, p_{ij} represents j-th element of $\boldsymbol{p}_i$, and k is the number of nearest neighbor we selected.

Finally, if we substitute $\boldsymbol{z}$ with $\boldsymbol{x}^{[t-1]}$ in Eq. (15), then we generate an iterative majorizing equation like following:

$$\boldsymbol{x}^{[t]} = \overline{\boldsymbol{p}} + \frac{1}{k} \sum_{i=1}^{k} \frac{\delta_{ix}}{d_{iz}} (\boldsymbol{x}^{[t-1]} - \boldsymbol{p}_i) \tag{16}$$

where $d_{iz} = \|\boldsymbol{p}_i - \boldsymbol{x}^{[t-1]}\|$ and $\overline{\boldsymbol{p}}$ is the average of k-NN's mapping results. Eq. (16) is an iterative equation used to embed newly added point into target-dimensional space, based on pre-mapped positions of k-NN. The iteration stop condition is essentially same as that of SMACOF algorithm, which is

Algorithm 1 Majorizing Interpolation (MI) algorithm

```
 1: Find k-NN: find k nearest neighbors of $\boldsymbol{x}$, $\boldsymbol{p}_i \in \boldsymbol{P}$ $i = 1, \ldots, k$ of the given new data based on original dissimilarity
    $\delta_{ix}$.
 2: Gather mapping results in target dimension of the k-NN.
 3: Calculate $\bar{\boldsymbol{p}}$, the average of pre-mapped results of $\boldsymbol{p}_i \in \boldsymbol{P}$.
 4: Generate initial mapping of $\boldsymbol{x}$, called $\boldsymbol{x}^{[0]}$, either $\bar{\boldsymbol{p}}$ or a random variation from $\bar{\boldsymbol{p}}$ point.
 5: Compute $\sigma(\boldsymbol{S}^{[0]})$, where $\boldsymbol{S}^{[0]} = \boldsymbol{P} \cup \{\boldsymbol{x}^{[0]}\}$.
 6: while $t = 0$ or $(\Delta\sigma(\boldsymbol{S}^{[t]}) > \varepsilon$ and $t \leq$ MAX_ITER) do
 7:     increase $t$ by one.
 8:     Compute $\boldsymbol{x}^{[t]}$ by Eq. (16).
 9:     Compute $\sigma(\boldsymbol{S}^{[t]})$.
10: end while
11: return $\boldsymbol{x}^{[t]}$;
```

$$\Delta\sigma(\boldsymbol{S}^{[t]}) = \sigma(\boldsymbol{S}^{[t-1]}) - \sigma(\boldsymbol{S}^{[t]}) < \varepsilon, \tag{17}$$

where $\boldsymbol{S} = \boldsymbol{P} \cup \{\boldsymbol{x}\}$ and ε is the given threshold value.

The time complexity of the MI-MDS algorithm [15] to find mapping of one interpolated point is $O(k)$ on the basis of Eq. (16), if we assume that the number of iterations of finding one interpolated mapping is very small. Since finding nearest neighbors takes $O(n)$ and mapping via MI-MDS requires $O(k)$ for one interpolated point, the overall time complexity to find mappings of overall out-of-sample points (N-n points) via the MI-MDS algorithm is $O(kn(N-n)) \approx O(n(N-n))$, due to the fact that k is usually negligible compared to n or N.

Process of the overall out-of-sample MDS with large dataset could be summarized as following steps: (1) Sampling, (2) Running MDS with sample data, and (3) Interpolating the remain data points based on the mapping results of the sample data.

The summary of MI-MDS algorithm for interpolation of a new data, say $\boldsymbol{x}$, in relation to pre-mapping result of the sample data is described in Alg. 1. Note that the algorithm uses $\bar{\boldsymbol{p}}$ as an initial mapping of the new point $\boldsymbol{x}^{[0]}$ unless initialization with $\bar{\boldsymbol{p}}$ makes $d_{ix} = 0$, since the mapping is based on the k-NN. $\bar{\boldsymbol{p}}$ makes $d_{ix} = 0$, if and only if all the mapping position of the k-NN is on the same position. If $\bar{\boldsymbol{p}}$ makes $d_{ix} = 0$ $(i = 1, \ldots, k)$, then the algorithm initializes a random variation from the $\bar{\boldsymbol{p}}$ point with the average distance of $\boldsymbol{\delta_{ix}}$ as a starting position of $\boldsymbol{x}^{[0]}$.

5. Adaptive Interpolation of MDS

An intrinsic assumption of MI-MDS [15] is that the mapping distances of k-NNs might be highly likely similar to the original dissimilarities of k-NNs. In real life, it is unlikely happened because the MDS results, which is used for prior mapping of sampled data, are usually produced with positive normalized STRESS values, which represent normalized errors for constructing mappings in target dimension. There is a possibility, however, that the smaller original dissimilarities will be represented in the smaller distances and the larger will be represented in the larger distances by the full MDS mappings of sampled data with regard to the distribution of the original dissimilarities and the distribution of the mapping distances of sampled data in target space. In other words, the mappings within a small region might be mapped in the target space within a proportional distances of the original dissimilarities by the full MDS running of the sampled data. On a basis of the above phenomenon, in this paper, we would like to propose an adaptive interpolation MDS algorithm which aims on the mapping quality improvement compared to the MI-MDS method.

5.1. Mapping Distance and Original Dissimilarity Comparison

At first, we investigated the distance ratio ($r = \delta_{ij}/d_{ij}$) between mapping distances among k-NNs (d_{ij}) and the corresponding original dissimilarities among k-NNs (δ_{ij}) of each interpolated points based on the prior mappings of

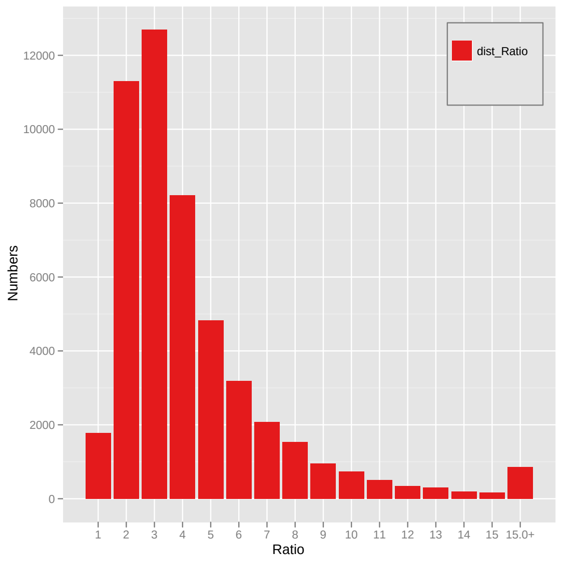


Figure 1: Distance Ratio between original dissimilarities among k nearest neighbors and the corresponding mapping distances among k nearest neighbors. The tested data here is pubchem data of total 100k and 50k sampled case and $k = 2$.

sampled data and the original pubchem data set. The explanation of the pubchem data is in Section 6. Fig. 1 shows the distribution of the distance ratio of 100,000 pubchem data with sampling 50,000 when $k = 2$.

Among about 50,000 cases, over 96% of the distance ratio (r) is larger than 1.0 and around 75% of that is in between 1.0 and 5.0. Interestingly, the cases of larger than 15.0 occur around 1,000 times. Those are the cases that the original dissimilarities (δ_{ij}) are too small but larger than ZERO. We exclude the case of $\delta_{ij} = 0.0$ in Fig. 1. Based on Fig. 1, we think that the adaptation of the original dissimilarities (δ_{iz}) between the interpolated point ($\boldsymbol{p}_z$) and the k-NNs ($\boldsymbol{p}_i$, where $1 \le i \le k$) can be helpful to get better interpolation mapping result.

5.2. *Adaptive Original Dissimilarity of k Nearest Neighbors of an Interpolated Point*

In the above section, we found that there are some difference between the prior mapping distances and the corresponding original dissimilarity of k-NNs. Since the interpolation method generates the mapping of each interpolated point based on the relation to the prior mappings of k-NNs of the point, it will be better to adjust the given original dissimilarities (δ_{iz}) between the interpolated point and its k-NNs based on the distance ratio of its k-NNs. Thus, we propose an adaptive interpolation MDS algorithm which constructs a mapping of an interpolated point based on the adapted dissimilarities.

The proposed adaptive interpolation of MDS (hereafter called *AI-MDS*) will interpolate points in relation to the prior mappings of the sampled data based on the adaptive dissimilarities between interpolated points and k-NNs. Those adaptive dissimilarities ($\widehat{\delta_{iz}}$)are calculated by multiplying the original dissimilarities (δ_{iz}) and the distance ratio between the average of the original dissimilarities of k-NNs ($\overline{\delta}_k$) and the average of the mapping distances of the corresponding k-NNs ($\overline{d}_k$), as shown in Eq. (18).

$$\widehat{\delta_{iz}} = \delta_{iz} \cdot \frac{\overline{\delta}_k}{\overline{d}_k} \tag{18}$$

Respectively, AI-MDS substitutes Eq. (16) with Eq. (19) for the iterative majorizing interpolation process, by exchanging δ_{ix} with $\widehat{\delta_{ix}}$.

$$\boldsymbol{x}^{[t]} = \overline{\boldsymbol{p}} + \frac{1}{k} \sum_{i=1}^{k} \frac{\widehat{\delta_{ix}}}{d_{iz}} (\boldsymbol{x}^{[t-1]} - \boldsymbol{p}_i). \tag{19}$$

Table 1: Compute cluster systems used for the performance analysis

# Nodes	32
CPU	Intel Xeon E7450 2.4 GHz
# CPU / # Cores per node	4 / 24
Total Cores	768
Memory per node	48 GB
Network	20 Gbps Infiniband
Operating System	Windows Server 2008 HPC Edition (Service Pack 2) - 64 bit

5.3. Parallelization of MDS Interpolation Algorithms

Suppose that, among N points, mapping results of n sample points in the target dimension, say L-dimension, are given so that we could use those pre-mapped results of n points via MDS interpolation algorithms which are described above to embed the remaining points ($M = N - n$). Though interpolation approach is very fast algorithm, i.e. $O(Mn)$, implementing parallel MDS interpolation algorithms is essential, since the out-of-sample size can be still huge, like millions. In addition, most of clusters are now in forms of multicore-clusters after multicore-chip invented, so we are using hybrid-model parallelism, which combine processes and threads together as used in [20, 1].

In contrast to the original MDS algorithm that the mapping of a point is influenced by the other points, interpolated points are totally independent one another, except selected k-NN in the MDS interpolation algorithms, and the independency of among interpolated points makes the MDS interpolation algorithm to be pleasingly-parallel. In other words, there must be minimum communication overhead. Also, load-balance can be achieved by using a simple modular calculation to assign interpolated points to each parallel unit, either between processes or between threads, as the number of assigned points are different at most one each other. Thus, we can parallelize MDS interpolation algorithms via not only traditional MPI but also the emerging MapReduce [21, 22] runtimes.

6. Analysis of Experimental Results

6.1. Experimental Enviroment

In this section, we provide some experimental analysis for the proposed AI-MDS algorithm. To explore the quality and performance of the proposed AI-MDS approach discussed in this paper compared to MI-MDS [15], we have used 166-dimensional chemical dataset obtained from PubChem project database[1], which is a NIH-funded repository for over 60 million chemical molecules and provides their chemical structures and biological activities, for the purpose of chemical information mining and exploration. In this paper we have used randomly selected up to 4 million chemical subsets for our testing. The computing cluster system we have used in our experiments is demonstrated in Table 1.

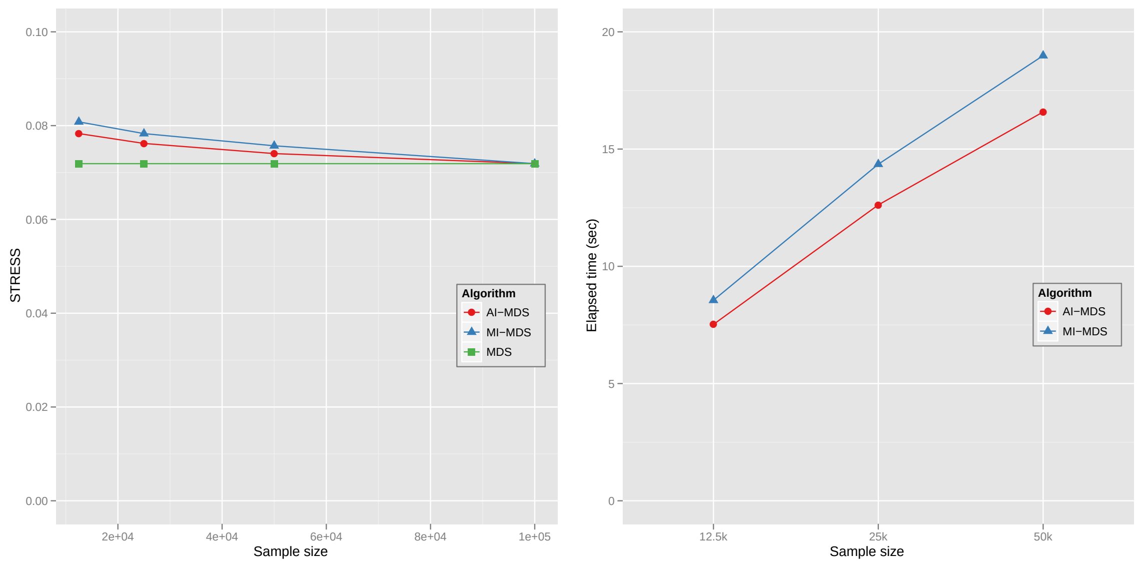
6.2. Quality and Runtime Analysis

In this section, we would like to compare the interpolation mapping quality of the proposed AI-MDS to that of the MI-MDS algorithm as well as the running time of both algorithms. For the quality measurement, we use the normalized STRESS value (σ) with uniform weights ($w_{ij} = 1$) defined as in Eq. (20).

$$\sigma(\boldsymbol{X}) = \frac{\sum_{i<j\leq N}(d_{ij}(\boldsymbol{X}) - \delta_{ij})^2}{\sum_{i<j\leq N}\delta_{ij}^2} \tag{20}$$

The normalized STRESS will be ONE if all the points are configured at the same position. In this section, when we use STRESS value term, it means the normalized STRESS value.

[1]PubChem, `http://pubchem.ncbi.nlm.nih.gov/`



(a) Mapping Quality (b) Interpolation Runtime

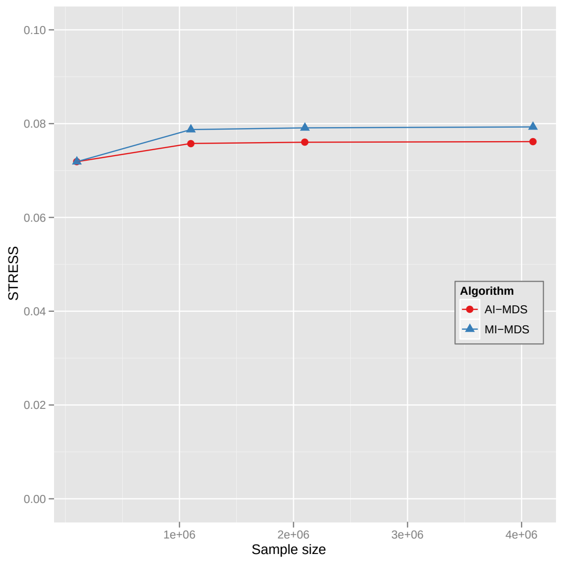
Figure 2: (a) Mapping quality comparison among full MDS, MI-MDS, and AI-MDS algorithm with 100k pubchem data set with respect to different sample sizes (n), i.e. 12.5k, 25k, and 50k, when $k = 2$, and (b) corresponding running time of MI-MDS and AI-MDS.

Fig. 2-(a) illustrates the mapping quality difference between AI-MDS results and MI-MDS results of 100,000 points (hereafter *100k*) (N) with respect to the different sample sizes (n). As shown in Fig. 2-(a), the proposed AI-MDS performs better than MI-MDS in terms of the mapping quality. In detail, if we define the quality degradation of MI-MDS ($\sigma_{full} - \sigma_{MIMDS}$) is equal to 1.0, then the quality degradation of AI-MDS ($\sigma_{full} - \sigma_{AIMDS}$) is 71.9%, 67.2%, and 55.4% of that of MI-MDS with respect to n = 12.5k, 25k, and 50k, correspondingly. In other words, if we assume that the possible best quality of interpolation approach will be the quality of full MDS running, the proposed AI-MDS improves the mapping quality about 28.1%, 32.8%, and 44.6% compared to the MI-MDS algorithm with the test dataset.

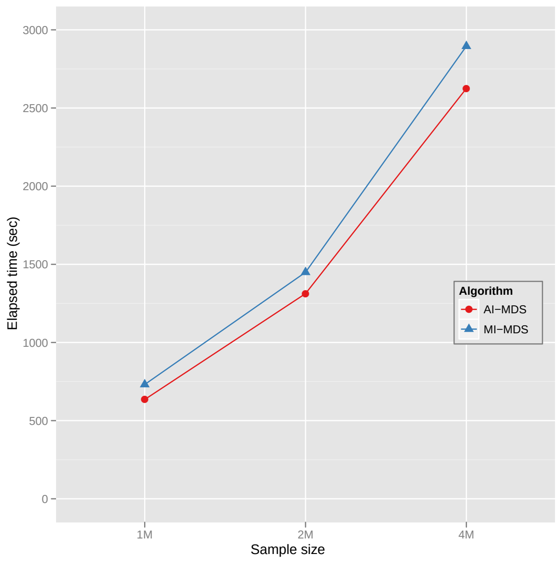
We have also compared the interpolation running time of the MI-MDS and the proposed AI-MDS algorithms in Fig. 2-(a), which is demonstrated in Fig. 2-(b), and it shows very interesting result. We expect the runtime of the proposed AI-MDS algorithm could be taken slightly longer than or compatible to that of MI-MDS, since the distance adaptive step is added for the interpolation procedure of each point. However, the runtime analysis in Fig. 2-(b) shows in the opposite to our expectation. The AI-MDS runs faster than MI-MDS for all test cases. Thus, we have also looked into detail of the number of iterations for interpolation of each point by MI-MDS and AI-MDS with 50k sample of 100k full data set. The average of iteration numbers by AI-MDS is 1.731, and the average of iteration numbers by MI-MDS is 1.834. We could understand this phenomenon as the search space of interpolation for each point becomes more suitable via adapting the dissimilarity.

In addition to the experiments of the fixed full data set in the above analysis, we have also tested the proposed AI-MDS with larger *out-of-sample* size cases, i.e. millions of points, as shown in Fig. 3. For the large data test, we randomly selected 100k pubchem data among over 60-million compounds data as an *in-sample* set, then we have tried to interpolate 1 million, 2 millions, and 4 millions (hereafter *1M, 2M, and 4M*, correspondingly) chemical compounds data, which are also randomly selected from the same dataset.

As shown in Fig. 3-(a), the proposed AI-MDS outperforms the MI-MDS method in terms of the mapping quality. The full MDS is infeasible to generate a mapping of millions of points as the motivation of MDS interpolation approach [15]. However, we could still compare the mapping quality of those interpolation approach with respect to the normalized STRESS value of the sample mapping, on the basis of the assumption that the sample mapping quality could be similar to the full MDS mapping result of all points. If we define the quality degration of MI-MDS and AI-MDS as same as the previous experiment, the quality degradation of AI-MDS is only about 56.7%, 57.5%,

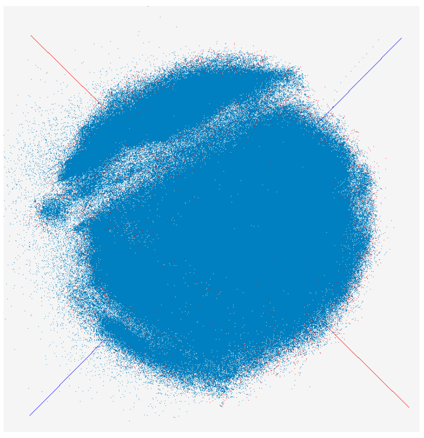


(a) Mapping Quality

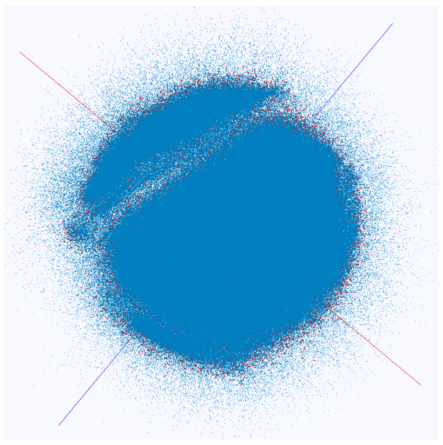


(b) Interpolation Runtime

Figure 3: (a) Mapping quality comparison between MI-MDS and AI-MDS algorithm with respect to large out-of-sample size (M), i.e. 1M, 2M, and 4M, with 100k sample pubchem data when $k = 2$, and (b) corresponding running time of MI-MDS and AI-MDS.



(a) AI-MDS mapping result



(b) MI-MDS mapping result

Figure 4: Interpolation Mapping results of 2M compounds by (a) AIMDS and (b) MIMDS with 100k sample data, when $k = 2$.

and 57.7% compared to the quality degradation of MI-MDS with 100k sample mapping and 1M, 2M, and 4M out-of-sample data, respectively, which means the AI-MDS improves the quality of interpolation more than 40% compared to the MI-MDS.

Relating to the runtime of those algorithms, AI-MDS is about 10% faster than MI-MDS as shown in Fig. 3-(b), which is consistent to Fig. 2-(b). The runtime results in Fig. 2-(b) and Fig. 3-(b) are the runtime of both AI-MDS and MI-MDS algorithms in hybrid parallel method by using 384 cores of the system in Table 1.

Fig. 4 illustrates the actual interpolation mapping result of 2M out-of-sample points (shown in blue color) based on 100k prior mapping (which is represented in red color) via (a) AI-MDS and (b) MI-MDS methods. As shown in Fig. 4-(b), MI-MDS result have points which are configured in the outside of the boundary of the prior mapping much more than AI-MDS result in Fig. 4-(a). We could interpret those outside mappings in Fig. 4-(b) are affected by the distance discrepancy between the original dissimilarity of each interpolated point and its k-NN and the established mapping space in the target dimension by the sample mapping.

7. Conclusion

Majorizing interpolation method for multidimensional scaling (MI-MDS) was proposed for the purpose of configuring millions of points via a commodity cluster systems based on the prior mapping of sample data [15]. As in [15], the MI-MDS method produces mappings of millions of points, which is infeasible via normal MDS methods, in reduced computational complexity by the cost of mapping quality degradation. Although the quality of MI-MDS is acceptable, we have investigated how to improve the mapping quality of MI-MDS algorithm in [15].

In this paper, we propose an adaptive interpolation method of MDS (AI-MDS) which aims to improve the mapping quality of MI-MDS based on the distance ratio between the original dissimilarities and the corresponding mapping distances in the target space. The proposed AI-MDS shows significant improvement of the mapping quality for the tested cases. For instance, AI-MDS algorithm configures mappings of millions of out-of-sample data cases which are improved more than 40%, with regard to the quality degradation from the full MDS mapping quality, compared to corresponding mappings of MI-MDS. Furthermore, the proposed AI-MDS generates better configuration of the tested data during faster running time than MI-MDS. The average of iterations for all interpolated points via the proposed AI-MDS is less than via the previous MI-MDS method, interestingly.

References

[1] G. Fox, S. Bae, J. Ekanayake, X. Qiu, H. Yuan, Parallel data mining from multicore to cloudy grids, in: Proceedings of HPC 2008 High Performance Computing and Grids workshop, Cetraro, Italy, 2008.

[2] C. Bishop, M. Svensén, C. Williams, GTM: A principled alternative to the self-organizing map, Advances in neural information processing systems (1997) 354–360.

[3] C. Bishop, M. Svensén, C. Williams, GTM: The generative topographic mapping, Neural computation 10 (1) (1998) 215–234.

[4] T. Kohonen, The self-organizing map, Neurocomputing 21 (1-3) (1998) 1–6.

[5] J. B. Kruskal, M. Wish, Multidimensional Scaling, Sage Publications Inc., Beverly Hills, CA, U.S.A., 1978.

[6] I. Borg, P. J. Groenen, Modern Multidimensional Scaling: Theory and Applications, Springer, New York, NY, U.S.A., 2005.

[7] J. Y. Choi, S.-H. Bae, X. Qiu, G. Fox, High performance dimension reduction and visualization for large high-dimensional data analysis, in: Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 331–340.

[8] J. B. Kruskal, Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, Psychometrika 29 (1) (1964) 1–27.

[9] Y. Takane, F. W. Young, J. de Leeuw, Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features, Psychometrika 42 (1) (1977) 7–67.

[10] Z. Wang, S. Zheng, Y. Ye, S. Boyd, Further relaxations of the semidefinite programming approach to sensor network localization, SIAM Journal on Optimization 19 (2) (2008) 655–673. doi:http://dx.doi.org/10.1137/060669395.

[11] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, M. Ouimet, Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering, in: Advances in Neural Information Processing Systems, MIT Press, 2004, pp. 177–184.

[12] M. W. Trosset, C. E. Priebe, The out-of-sample problem for classical multidimensional scaling, Computational Statistics and Data Analysis 52 (10) (2008) 4635–4642. doi:http://dx.doi.org/10.1016/j.csda.2008.02.031.

[13] W. S. Torgerson, Multidimensional scaling: I. theory and method, Psychometrika 17 (4) (1952) 401–419.

[14] S. Ingram, T. Munzner, M. Olano, Glimmer: Multilevel mds on the gpu, IEEE Transactions on Visualization and Computer Graphics 15 (2) (2009) 249–261.

[15] S. Bae, J. Choi, J. Qiu, G. Fox, Dimension reduction and visualization of large high-dimensional data via interpolation, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, ACM, 2010, pp. 203–214.

[16] J. de Leeuw, Applications of convex analysis to multidimensional scaling, Recent Developments in Statistics (1977) 133–145.

[17] J. de Leeuw, Convergence of the majorization method for multidimensional scaling, Journal of Classification 5 (2) (1988) 163–180.

[18] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the em algorithm, Journal of the Royal Statistical Society. Series B (1977) 1–38.

[19] T. M. Cover, P. E. Hart, Nearest neighbor pattern classification, IEEE Transaction on Information Theory 13 (1) (1967) 21–27.

[20] J. Qiu, S. Beason, S. Bae, S. Ekanayake, G. Fox, Performance of windows multicore systems on threading and mpi, in: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE, 2010, pp. 814–819.

[21] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, in: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, USENIX Association, Berkeley, CA, USA, 2004, pp. 137–150.

[22] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (2008) 107–113.