Procedia Computer Science

Volume 51, 2015, Pages 572–581

ICCS 2015 International Conference On Computational Science

Very Fast Interactive Visualization of Large Sets of High-dimensional Data

Witold Dzwinel and Rafał Wcisło

AGH University of Science and Technology, Cracow, Poland
{dzwinel|wcislo}@agh.edu.pl

Abstract

The embedding of high-dimensional data into 2D/3D space is the most popular way of data visualization. Despite recent advances in developing of very accurate dimensionality reduction algorithms, such as BH-SNE, Q-SNE and LoCH, their relatively high computational complexity still remains the obstacle for interactive visualization of truly large datasets consisting of $M \sim 10^{6+}$ of high-dimensional $N \sim 10^{3+}$ feature vectors. We show that a new clone of the multidimensional scaling (MDS) – *nr*-MDS – can be up to two orders of magnitude faster than the modern dimensionality reduction algorithms. We postulate its linear $O(M)$ computational and memory complexities. Simultaneously, our method preserves in 2D/3D target spaces high separability of data, similar to that obtained by the state-of-the-art dimensionality reduction algorithms. We present the effects of *nr*-MDS application in visualization of data repositories such as 20 Newsgroups ($M = 1.8 \cdot 10^4$), MNIST ($M = 7 \cdot 10^4$) and REUTERS ($M = 2.67 \cdot 10^5$).

Keywords: multidimensional scaling, particle-based stress minimization, interactive visualization

1 Introduction

Visual exploration of high-dimensional data is an essential component of the big data analytics [10]. Such the data can be represented as a set $\Re^N \ni \mathbf{Y} = \{\mathbf{y}_i = (y_{i1}, ..., y_{iN})\}_{i=1,...,M}$, of M, N-dimensional feature vectors, with $N \sim 10^{3+}$. Data mapping $B : \mathbf{Y} \to \mathbf{X}$ to a visually perceived space $\Re^n \ni \mathbf{X} = \{\mathbf{x}_i = (x_{i1}, ..., x_{in})\}_{i=1,...,M}$, where $n = \dim \mathbf{X} = 2(3)$, which preserves significant topological properties of $\mathbf{Y}$ in $\mathbf{X}$, is a truly algorithmic challenge.

A plethora of dimensionality reduction techniques have been developed over the last decade (see e.g. [19, 2, 5]). However, most of them are not capable of retaining both the local and global properties of data in a single map. One of the most popular embedding technique – multidimensional scaling (MDS) – computes a low-dimensional map of points with interpoint Euclidean distances $\mathbf{d} = \{r_{ij}\}_{M \times M}$ that approximate input distance matrix $\mathbf{\Delta} = \{D_{ij}\}_{M \times M}$, where $D_{ij} = D(\mathbf{y}_i, \mathbf{y}_j) \to \Re^1$ is a dissimilarity measure. The type of properties of $\mathbf{Y}$ (local or global), which will be preserved in $\mathbf{X}$, depends on the error function $V : \Re^{n \times M} \to \Re^1$

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.325

(called "the stress"). The stress $V(\|\mathbf{\Delta} - \mathbf{d}\|)$ is the function of the norm of difference between distance matrices from the source $\mathbf{Y}$ and the target $\mathbf{X}$ spaces. In the classical MDS algorithms, a configuration of points $\mathbf{X} = \{\mathbf{x}_i\}_{i=1...M}$, which minimizes the stress:

$$E(X) = \min_x V(\|\mathbf{\Delta} - \mathbf{d}\|) = \min_x \sum_{i<j} w_{ij} \left(D_{ij}^k - r_{ij}^k\right)^m \tag{1}$$

represents the result of $\mathbf{Y}$ embedding in 2D/3D target space. The values of k, m are the parameters of $V(\cdot)$. For low-dimensional data ($N \sim 10$), MDS mapping is able to properly approximate the structure of the original data in 2D/3D spaces. However, the result of visualization of high-dimensional data with $N \sim 10^{2+}$ is highly unreliable.

For $N \sim 10^{2+}$, due to the "curse of dimensionality" principle, MDS produces the low-dimensional map with a characteristic spherical shape, and a low density in the sphere center (e.g., see Figure 2b). Thus the structure of the source space becomes strongly distorted and the clusters separability can be lost. Second, both the memory and computational complexities of the classic MDS algorithm are at least $O(M^2)$, i.e., prohibitively high for visualization of datasets with $M \sim 10^{5+}$. Moreover, the minimization of criterion (1) requires sophisticated and slow optimization methods to find the global minimum of $V(\cdot)$. The gradient based techniques can get stuck in a local minimum.

The first problem can be solved by using recently developed dimensionality reduction algorithms that preserve small pairwise distances via stochastic neighbor embedding, such as t-SNE and its clones [18]. From methodological point of view, these algorithms consist in replacing $V(\|\mathbf{\Delta} - \mathbf{d}\|)$ in the criterion (1) with a more general one $V(\mathbf{\Delta}, \mathbf{d})$. In t-SNE and its clones the low-dimensional map is obtained as the result of minimization of $V(\mathbf{\Delta}, \mathbf{d})$, which is equal to the Kullback-Leibler divergence between the two probability distributions computed in the source and the target spaces with respect to the locations of the points in the latter one. Despite impressive success of t-SNE, its memory and computational complexities remain the same as for MDS. Larger data sets ($M \sim 10^{4+}$) can be visualized by using approximated versions of t-SNE, such as BH-SNE [17] and Q-SNE [7] or LoCh [4] algorithms. The computational complexities of these techniques are $O(M \log M)$ for BH-SNE and Q-SNE while $O(M\sqrt{M})$ for LoCH.

Though these algorithms reconstruct very precisely the local properties of the original data in the low-dimensional map - such as the nearest neighbors lists and distances distribution – they are still too slow to be used in interactive visualization of big data. For example, the fastest BH-SNE algorithm requires as many as 12 minutes (serial code running on Intel® Core™ i5-4258U, 2.6GHz) to visualize $7 \cdot 10^4$ samples of MNIST data set [17]. Up to now, there are not any published parallel implementations of BH-SNE, Q-SNE and LoCH to estimate how the algorithms will perform on GPU or MIC architectures. Moreover, although the memory complexity was decreased in BH-SNE to $O(M)$, the number of distances, which has to be kept in the operational memory, remains too large. It is proportional to the squared average number of points in Barnes–Hut cells.

Moreover, the gradient descent algorithm used for minimization of the error function is too stiff. It does not allow for interactive steering during visualization, e.g., changing manually the error function type and its parameters to find the best embedding. Meanwhile, interactive data mining enables to match optimal machine learning tools to data being explored and allows for deeper penetration of their parameter spaces.

We describe a new clone of MDS – *nr*-MDS – which outperforms the state-of-the-art dimensionality reduction techniques. We postulate its $O(M)$ memory and computational complexities. Thus it is not only orders of magnitude faster than state-of-the-art dimensionality reduction techniques but, unlike classical MDS, properly reconstructs separability of high-dimensional

data sets in 2D/3D spaces. In the following section we present our idea, its methodological background and, consequently, the results of experiments. Finally, we present the conclusions and discuss directions for future work.

2 MDS with reduced dissimilarity matrix

In [12] we presented the parallel versions of particle-based MDS algorithm, which do not use all of distances $\mathbf{\Delta} = \{D_{ij}\}_{M\times M}$, i.e., $L_{\max} = M\cdot(M-1)/2$. We showed that the acceptable approximations of the source space $\mathbf{Y}$ can be obtained by using only a small subset $\mathbf{D}$, $nd = \#\mathbf{D}$ of all distances from $\mathbf{\Delta}$. In the simplest case, they can be taken in random from the full set of $L_{\max}$ distances.

The minimum number of distances, which is sufficient to obtain a stable solution in $\mathbf{X}$ is equal to $L_{\min} = n\cdot M - n\cdot(n+1)/2$, i.e., for small n and large M $L_{\min} = n\cdot M$. Therefore, the number of distances nd taken in random from the full set of $L_{\max}$ distances, allowing for unambiguous reconstruction data topology, $nd \geq L_{\min}$. As shown in [12], our RANDOM algorithm performs surprisingly well, giving errors and quality of embeddings very similar to the full-distance MDS but in incomparable shorter time. The timings obtained suggest $O(M)$ complexity of RANDOM MDS algorithm for $nd \sim 20M{-}50M$. Moreover, the SUBSET version of RANDOM algorithm, in which the distances were selected in a more rigorous way [12], demonstrate excellent scaling properties on the multi-thread architectures, such as CPU and GPU, speeding up the serial code by additional two orders of magnitude.

$\mathbf{X}_0 =$ *set_random_configuration*()
do (*for timestep* $n = 0, 1, 2, \ldots$ *; for every particle* $i = 1 \ldots M$)
$\quad \mathbf{F}_{n,i} := -\sum_{j\in D_i} w_{ij}\nabla(D_{ij}^k - r_{n,ij}^k)^m$ // 1. compute forces acting on each particle i
$\quad$ // from particles j, where $r_{n,ij} := \|\mathbf{x}_{n,i} - \mathbf{x}_{n,j}\|$
$\quad \mathbf{p}_{n+1,i} := \mathbf{p}_{n,i} + \frac{1}{2}\mathbf{F}_{n,i}\cdot\Delta t$ // 2. calculate momenta $\mathbf{p}_i$ for each particle
$\quad$ **if** $|\mathbf{p}_{n+1,i}| > (m=1)\cdot v_{\max}$ **then**
$\quad\quad \mathbf{p}_{n+1,i} := v_{\max}\cdot\mathbf{p}_{n+1,i}/|\mathbf{p}_{n+1,i}|$ // 3. avoid numerical blow up
$\quad \mathbf{x}_{n+1,i} := \mathbf{x}_{n,i} + \mathbf{p}_{n+1,i}\cdot\Delta t/(m=1)$ // 4. move particles, create new configuration $\mathbf{X}_{n+1}$
$\quad \mathbf{F}_{n+1,i} :=$ *calculate_forces*() // 5. calculate new forces $\mathbf{F}_{n+1,i}$
$\quad \mathbf{p}_{n+1,i} := \varphi\cdot(\mathbf{p}_{n+1,i} + \frac{1}{2}\mathbf{F}_{n+1,i}\cdot\Delta t)$ // 6. use "damping" factor φ
until $(|E(\mathbf{X}_{n+1} - E(\mathbf{x}_n)| < \delta)$

Figure 1: The pseudocode of the particle-based MDS. D_i are the sets of random neighbors of each data sample i.

For minimization of $V(\|\mathbf{\Delta} - \mathbf{d}\|)$ we used very fast and robust particle-based MDS method, which pseudocode is shown in Figure 1. The same approach can be used for minimization of any error function $V(\mathbf{\Delta}, \mathbf{d})$, such as that employed in t-SNE. Shortly, the particle-based MDS represents a data sample $\mathbf{y}_i$ as a particle $\mathbf{x}_i$ in 2D/3D target space. They interact with each other via semi-harmonic forces $\boldsymbol{f}_{ij}(\mathbf{X}_n) = -\mathbf{grad}[w_{ij}(D_{ij}^k - r_{n,ij}^k)^m]$. Every two particles i and j located in $\mathbf{x}_{n,i}$ and $\mathbf{x}_{n,j}$ in timestep n are separated by the Euclidean distance $r_{n,ij} = \|\mathbf{x}_{n,i} - \mathbf{x}_{n,j}\|$. The total force $\mathbf{F}_{n,i}$ acting on a particle i is equal to the sum of $\boldsymbol{f}_{ij}(\mathbf{X}_n)$ forces from particles belonging to its random neighbors rn_i. We simulate the Newtonian dynamics of (initially) random configuration of particles $\mathbf{X}_0$ in discrete timesteps n. We assume additionally that particles $\mathbf{X}_n$ evolves in a dissipative environment represented by a damping factor φ (see Figure 1). The particle system $\mathbf{X}_n$ converges to a stationary state $\mathbf{X}$ with a minimal potential

energy $E(\mathbf{X}) = V(\mathbf{\Delta}, \mathbf{d})$. Due to more extensive penetration of the stress function domain, the particle-based method allows for obtaining a better minimum (i.e., closer to the global one) that any other gradient-based techniques [1, 3]. Moreover, particle-based MDS is really fast. By using the "leap-frog" numerical scheme, which is very resistant on fluctuations and very stable, one can use large timesteps Δt without worrying about unphysical effects and precision of $\mathbf{X}_n$ time evolution. In the beginning of simulation the particle shifts can be very large. Thus, initially, the method works like the simulated annealing algorithm with high temperature. The numerical "heating" can be controlled by using large or variable damping factor φ (from $[0, 1)$) and small value of $v_{\max}$. At the end, close to the attractor point, the algorithm behaves like the gradient descent method. The particle-base heuristics is fast and self-adaptive. It requires only two parameters to be set (Δt and φ). Furthermore, it fits perfectly to interactive visualization purposes. The time evolution of the particle system $\mathbf{X}$ can be visualized and controlled interactively by changing not only the parameters but the stress function type as well.

However, as shown in Figure 2b, MDS yields very poor embeddings for high-dimensional and multi-class data [17]. As shown in Figure 2a, even PCA can outperform MDS in visualization of 10 class 784-dimensional MNIST data set. This is the consequence of "curse of dimensionality" principle, i. e. in high-dimensional spaces all the distances between data samples are practically the same. Moreover, the high-dimensional data usually create a low-dimensional manifold Σ embedded in the feature space $\mathbf{Y}$, such that $\dim \Sigma \ll \dim \mathbf{Y}$. Because the manifold can be very complicated and folded, the distances between samples in $\mathbf{Y}$ can be completely different than those on the manifold Σ. Therefore, in the high-dimensional spaces, only small pairwise distances between neighboring data vectors are reliable. The isomap-based MDS assumes that pair-wise distances between the nearest neighboring points are only known. To approximate all other distances on the manifold Σ the Floyd-Warshall algorithm can be employed [16].

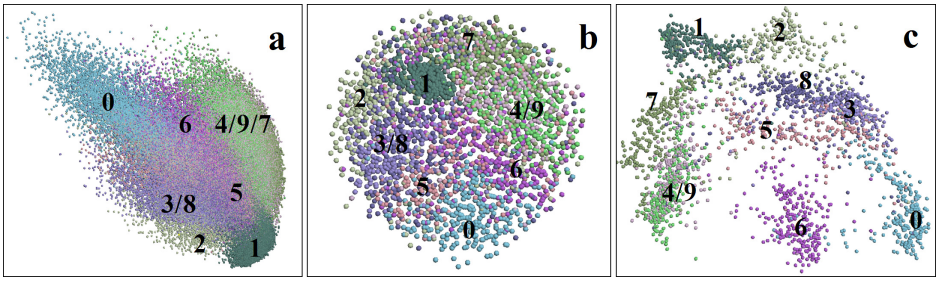

Figure 2: The results of mapping into 2D space of MNIST data set by using a) PCA ($M = 7 \cdot 10^4$), b) MDS ($M = 2.5 \cdot 10^3$, $k = 1$, $m = 2$), c) nr-MDS ($M = 2.5 \cdot 10^3$, $k = 1$, $m = 2$).

The main idea of nr-MDS is different than in [16]. We are trying to "unfold" the manifold and fit it into the hyperplane of minimal dimensionality by modifying the distances $\mathbf{\Delta}$ in the original multi-dimensional space $\mathbf{Y}$, prior to its embedding in $\mathbf{X}$ space. Simultaneously, to speed up the calculations, we select the smallest possible set of distances $\mathbf{D} = \{D_{ij}\}$, $nd = \#\mathbf{D}$, sufficient to obtain a proper data embedding. To this end, we construct the neighbor graph $G(\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is the set of nodes representing data samples $\mathbf{y}_i \in \mathbf{Y}$, and $\mathbf{E}$ is the set of edges corresponding to the selected distances $\mathbf{D}$. To define $\mathbf{E}$ we compute the sets $\mathbf{Knn}_i$ and $\mathbf{Krn}_i$ of edges connecting each point $\mathbf{y}_i \in \mathbf{Y}$ to their nn nearest neighbors and rn random neighbors, respectively. The nearest neighbor of particle $\mathbf{y}_i$ cannot be simultaneously its random neighbor. Then, the set of edges $\mathbf{E} = \mathbf{Knn} \cup \mathbf{Krn}$, where $\mathbf{Knn} = \bigcup_{i=M} \mathbf{Knn}_i$ and $\mathbf{Krn} = \bigcup_{i=M} \mathbf{Krn}_i$.

It means that a node $i \in \mathbf{V}$ has at least $nt = nn + rn$ outgoing edges to its neighbors (both the nearest and random ones) and, possibly, some additional edges connecting it to other data samples $j \in \mathbf{V}$, which do not belong to $\mathbf{E}_i = \mathbf{Knn}_i \cup \mathbf{Krn}_i$ while i belongs to their nearest or random neighbors. Let us assume that all distances between the nodes of $G(\mathbf{V}, \mathbf{K})$ connected by the edges from **Knn** are set to 0. The number of zeroed distances $nnd = \#\mathbf{Knn} > nn$, while the number of random non-zero distances $rnd = \#\mathbf{Krn} > rn$. The total number of distances $nd = \#\mathbf{E} = \#\mathbf{D} = nnd + rnd$ should be $nd \geq L_{\min}$.

Setting the distances of the nearest data samples to 0 causes the manifold "shrinkage" while non-zero random distances produce the "surface tension", which is trying to unfold it. The quality of the final result depends mainly on selecting the proper balance between the nearest nn and random neighbors rn set for each data sample. This balance depends on the stress function type and the structure of data visualized. The influence of long random distances inscribing the global structure of data visualized can be additionally controlled by employing different weights w_{nn} and w_{rn} for forces acting between the nearest neighbors and random neighbors respectively (see step 1 in the pseudocode from Figure 1). Hence, in *nr*-MDS, the forces acting on each particle i are computed as follows:

$$\mathbf{F}_{n,i} := -\sum_{j \in \mathbf{Krn}_i} w_{rn} \cdot \nabla \left(D_{ij}^k - r_{n,ij}^k\right)^m - \sum_{j \in \mathbf{Knn}_i} w_{nn} \cdot \nabla \left(r_{n,ij}^k\right)^m \tag{2}$$

In all the tests we have assumed that w_{nn} and w_{rn} are constant and $w_{nn} = 1$. As shown in Figure 2c and Figure 3a,b, we obtained very good cluster separation similar to [18, 9], better than in [6] and much better than in [20]. In the following section we present the preliminary results of visualization of much larger data sets.

3 Experiments

3.1 Data sets

We performed experiments using three very popular data sets: MNIST, REUTERS and the 20-NG. We briefly describe each of them below and in Table 1.

MNIST. The source MNIST data set (http://yann.lecun.com/exdb/mnist/) contains $M = 7 \cdot 10^4$ grayscale handwritten digit images of size $N = 28 \times 28 = 784$ pixels, each of which belongs to one of 10 classes. We made experiments for various number of vectors taken randomly from the source data set. We use as input both the raw pixel values and vectors preprocessed by using PCA to reduce data dimensionality (784D $\rightarrow$ 30D).

REUTERS. This text corpus is known as "Reuters Corpus, Volume1" or RCV1. We used a subset of this repository (http://www.jmlr.org/papers/v5/lewis04a.html) containing $M = 2.67 \cdot 10^5$ documents and 8 topics. We used two types of data. (1) Data set of 2000-dimensional BoW (bag of words) vectors of the most frequently used words without "stop words" and preprocessed by using *Porter2* stemming. The vectors are normalized to 1 and, finally, preprocessed by using *tf.idf* ranking. (2) Data set of 30-dimensional vectors preprocessed by PCA projection of 6000-dimensional structural – semantic vector representation. The data set is highly imbalanced. For $M = 2.67 \cdot 10^5$ the topics and their sizes as follows: GDIS (8081), C12 (11 944), ECAT (89 131), M131 (25 719), C151 (89 683), E212 (27 257), G154 (1342), M143 (21 774).

20NG. The twenty newsgroups data set (http://qwone.com/ jason/20Newsgroups/) is a collection of $M = 1.8 \cdot 10^4$ documents, partitioned evenly across 20 various newsgroups. They 2000D BoW vectors were preprocessed in a similar way as the REUTERS dataset. The 30D vectors were obtained by using PCA projection 2000D→30D.

For all of these data sets we calculate the values of classification factor *cf*. It is defined as the ratio of the number of data samples for which the closest neighbor belongs to the same class, to the total number of samples M. We compute the *cf* values for the source data set and its PCA and *nr*-MDS embeddings in 3D space. Of course, the *cf* value cannot be a measure of the quality of embedding, but is one of important indicators of data separation [11].

Data name	#vectors (M)	dimensionality (N)	coordinates	metrics	# classes	*cf* source N-D	*cf* PCA 3D	*cf* *nr*-MDS 3D
MNIST	70 000	784	grayscale	Euclidian	10	0.9745	-	0.9547
	10 100	784	(0-255)			0.9491	-	0.9031
	2 500	784				0.9044	-	0.8316
	70 000	PCA (784→30)				0.9778	0.4364	0.9655
	10 000	PCA (784→30)				0.9491	0.4334	0.9238
	2 500	PCA (784→30)				0.9044	0.4620	0.8492
REUTERS	53 386	2000	*tf.idf*	1-cosinus	8	0.8856	0.6498	0.8335
	26 693	2000				0.8675	0.6822	0.8071
	266 931	PCA (6000→30)				0.9359	0.5614	0.9005
	53 386	PCA (6000→30)				0.9265	0.5461	0.8900
	26 693	PCA (6000→30)				0.9232	0.4971	0.8873
20NG	18 759	2000	*tf.idf*	1-cosinus	20	0.6593	-	0.6162
	18 759	PCA (2000→30)				0.6917	0.2911	0.6004

Table 1: The list of data sets used for tests.

3.2 Experimental setup

The most of important parameters are as follows. Setting up k, m, w parameters allows for selecting a proper "stress" function (1). They can be changed anytime, even during the simulation. Similarly, the parameters of optimization procedure such as the timestep Δt and dumping factor φ, can be controlled interactively. To prevent numerical blow-up the velocity of particles can be bounded i. e. $|\mathbf{v}_i| < v_{\max}$. The other parameters such as nn, rn and the weights w_{rn} and w_{nn} should be properly matched. For example, for the stress with $k = 1$ and $m = 2$ (1) and for 2D and 3D maps we assume the smallest possible number of neighbors pair (nn, rn) i.e., (2, 1) and (3, 1), respectively. This assumption yields the number of distances computed very close to the minimal number of degrees of freedom $L_{\min}$. In fact, the distances to the nearest neighbors have not to be stored in the computer memory, because we assume that all of them are set to 0. To diminish the influence of long range (random) distances we decreased the weights of the forces acting between the samples and their random neighbors as follows: $w_{rn} = 0.01$ for 2D and $w_{rn} = 0.1$ for 3D maps while w_{nn}=1 in all the experiments. When the value of nn is larger the influence of long distances in 3D is weaker. For the stress function with $k = 1$ and $m = 1$, the influence of long distances is even smaller, so the weights w_{rn} are set to 1. The simulations were performed by using non-optimized and scalar version of C++ code working in on-line mode with visualization framework.

The code was run on a notebook computer with an Intel® Core™ i5 3317U CPU with 16GB of memory running at 1.7GHz (i.e., 60% of the CPU clock used in [17]).

3.3 Results

The 2D mappings of larger MNIST data sets than that visualized in Figure 2c are shown in Figure 3. They were performed for the original data set with dimensionality $N = 784$ and the stress function with $k = 1$ and $m = 2$ (1). As shown, in Figure 3a,b, the results of embeddings for large M have improved significantly. The cluster overlaps are much smaller for the full dataset with $M = 7 \cdot 10^4$ samples, what is reflected in increasing cf value (see Table 1). It grows from 0.68 for $M = 2.5 \cdot 10^3$ to 0.88 for the full MNIST data set. The cluster separability increases considerably for the stress function with $k = 1$ and $m = 1$. However, as shown in Figure 3c, the clusters shrink too much, and their local structure becomes invisible. As shown in Table 1, the PCA preprocessing of raw data improves distinctly (2-3%) the values of cf and data separability.

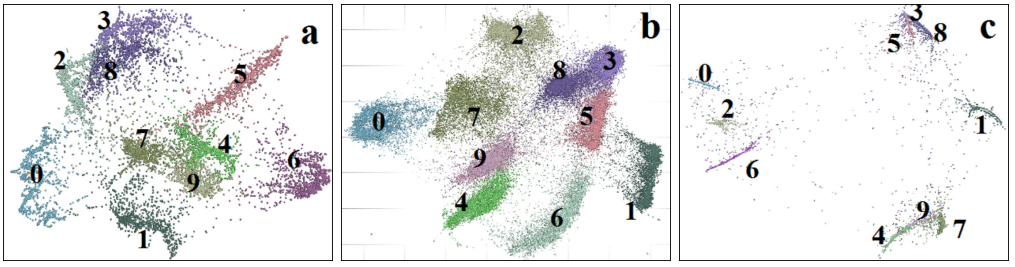

Figure 3: The results of mapping into 2D space of MNIST by using nr-MDS for the original data set ($N = 784$) a) $M = 10^4$, $k = 1$, $m = 2$, $w = 0$, b) $M = 7 \cdot 10^4$, $k = 1$, $m = 2$, $w = 0$, c) $M = 7 \cdot 10^4$, $k = 1$, $m = 1$, $w = 0$.

For the full MNIST data set the number of distances for $nn = 2$ i $nr = 1$ is equal to $L = 1.8 \cdot 10^5$, i.e., very close to the minimal configuration $L_{\min} = 2M = 1.4 \cdot 10^5$. However, comparing L to $L_{\max} = 2.4 \cdot 10^9$, the improvement in memory and computational complexity in comparison to the classical MDS is overwhelming. Moreover, because all the distances to the nearest neighbors are set to 0, only distances to random neighbors should be stored. The same "minimal" setup was used for M from $2.5 \cdot 10^3$ to $7 \cdot 10^4$ MNIST sets. This suggests that the computational complexity can be $O(M)$. The measured timings obtained for visualization of MNIST data sets in 2D are as follows: $t = 1$ sec. for $M = 2.5 \cdot 10^3$ (Figure 2c); $t = 3.5$ sec. for $M = 10^4$ (Figure 3a); $t = 20$ sec. for $M = 7 \cdot 10^4$ (Figure 3b,c)). As reported in [17], the BH-SNE needs 750 sec. to obtain similar separation of clusters. By normalizing the computational power of CPUs used for visualization in both cases, nr-MDS is more than 60 times faster than BH-SNE on MNIST dataset. Moreover, in this particular case (2D mapping), we need to store only one random distance per data sample, one index of random neighbor and 2 indices of its nearest neighbors. So, the memory complexity is also very low.

However, MNIST is the data set of rather well separated-classes. Much more difficult are text repositories such as REUTERS and 20NG. The documents and news are represented as *tf.idf* vectors based on the most frequent words. Such the data representation is often very noisy and, in some situations, highly unreliable. Moreover, REUTERS is a very unbalanced data set. Nevertheless, as shown in Figure 3 and Table 1, our algorithm was able to obtain 2D/3D maps of well separated clusters. It is also clearly seen that 30D data sets of PCA projections of 6000D

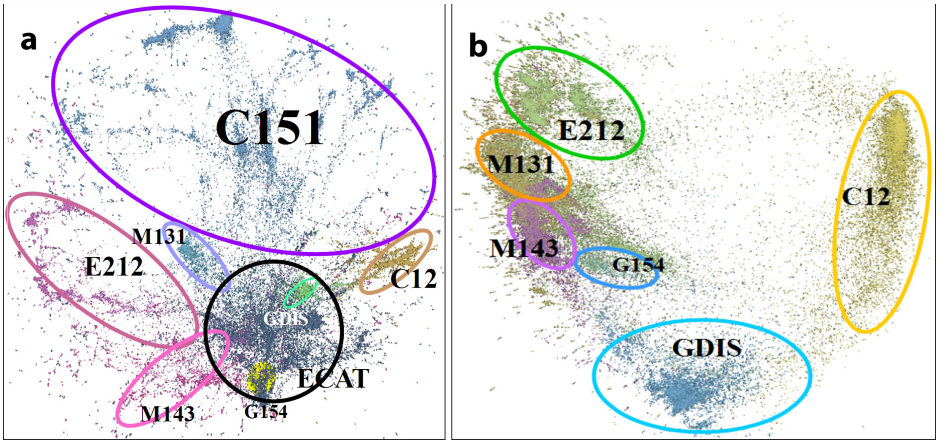

Figure 4: The results of mapping into 2D space of REUTERS data set by using *nr*-MDS. a) data set with $M = 5.3 \cdot 10^4$, b) $M = 2.67 \cdot 10^5$ samples were used, however, the largest two clusters C151 $(8.9 \cdot 10^4)$ and ECAT $(8.9 \cdot 10^4)$ were removed for better visualization of the smaller topics $(M = 9.6 \cdot 10^4)$.

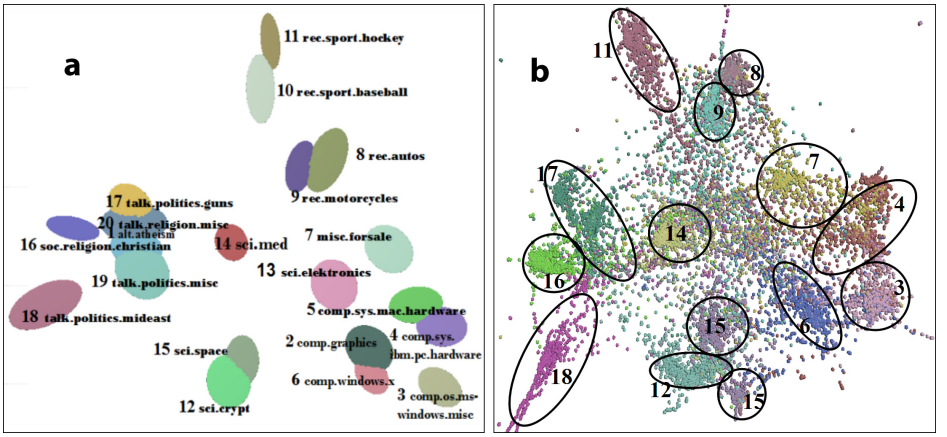

Figure 5: The results of mapping into 2D space of 20NG data set by using *nr*-MDS. a) The clusters of all 20 topics. b) The scatter-plot showing 14 the most visible clusters.

structural-semantic representation of documents and their 3D embeddings give greater values of cf than 2000D BoW vectors. The computational time needed for embedding of $M = 2.67 \cdot 10^5$ samples in 2D is about 2 minutes what is acceptable for interactive data visualization. However, for out-of-date Intel® i5 3317U CPU (1.7GHz) processor used in the experiments, this is rather the upper limit.

The 20NG data set is the most difficult data set out of the three. It is rather small, contains shorter documents from as many as 20 classes. It is more noisy than REUTERS. Additionally, many classes are sematically and structurally very close and they overlap each other. As shown in Figure 5a, the *nr*-MDS produces 2D map, which reflects the global data structure, reconstructing properly similarity relations between the newsgroups. As shown in

Figure 5b, as many as 14 clusters can be clearly visible. The result of clusters separability is inferior to that obtained by van der Maaten (see http://lvdmaaten.github.io/tsne/), which uses Simon's FDA generated features. However, it is much better than others, which used BoW text representation (see e.g. [15]) where only six the best separated clusters are visualized and there is no room too fit 14 more).

4 Conclusions

In this paper we present the effects of adaptation of the particle-based MDS to interactive visualization of large high-dimensional data sets. First, to decrease the computational complexity of multidimensional scaling we compute the smallest possible set of distances between the original high-dimensional data $\mathbf{Y}$, which are sufficient to obtain a proper data embedding in $\mathbf{X}$. We show that for each data vector in $\mathbf{Y}$ only indices of a small number of the nearest neighbors nn and distances to the random neighbors rn have to be computed, such that $nn+nr \geq n$. Second, the main idea of nr-MDS consists in modification of these distances prior to data embedding into 2D/3D space setting all distances between a data sample and its nearest neighbors nn to zero. Thus, only the distances to random neighbors nr are stored. All of these improvements make the algorithm very fast. Much faster than the top dimensionality reduction techniques due to its lower $O(M)$ computational and memory complexities. The efficiency of nr-MDS can be increased considerably by employing modern multi-thread computer architectures such as GPU and MIC [2, 8, 12, 13, 14]. Furthermore, the results of embeddings obtained by nr-MDS are amazingly good, reconstructing in 2D/3D properly separated classes from high-dimensional data sets. In general, this concept can be applied for various type of data reduction techniques, because it does not depend on the particular technique used.

Of course, our algorithm has also some important drawbacks. It yields poor results in reconstructing the lists of the nearest neighbors (NN). The "crowding problem", similar as in SNE [18], causes that particles gather in the center of clusters (see Figure 3, especially Figure 3c). Furthermore, the method is not fully understood, and more research on its theoretical basis is required. Also more experiments on various data sets and direct comparisons to other data reduction techniques should be carried out.

However, the major goal in interactive data visualization is very fast generation of 2D/3D maps, which preserve in $\mathbf{X}$ the clusters separation from $\mathbf{Y}$. The nr-MDS fits perfectly for this purpose. The accurate reconstruction of the nearest neighbors lists for each particle i in $\mathbf{X}$ is rather a secondary requirement. First, the NN lists were computed from the original data and are known before mapping, so they can be recalled in any moment of interactive visualization and presented visually, e.g., as the spines pointing from a given sample to its nn nearest neighbors. Second, the lists of NN are not reliable from the context of both measurement errors and the "curse of dimensionality" principle. Thus the knowledge they provide is very volatile. Meanwhile, just the clusters in various data resolutions represent "primordial" granules of knowledge. When the accurate reconstruction of the nearest neighbors in 2D/3D will be required, the nr-MDS can be employed first to generate an initial configuration for more advanced techniques such as BH-SNE (Q-SNE). Summing up, we expect that the method presented in this paper will be the inspiration of developing more efficient and reliable interactive visualization tools for exploration of really big data sets.

Acknowledgments. This research is supported by the Polish National Center of Science (NCN) DEC-2013/09/B/ST6/01549. We thank dr Marcin Kurdziel and dr Wojciech Czech from AGH Department of Computer Science for their contribution to this paper.

References

[1] M. Andrecut. Molecular dynamics multidimensional scaling. *Physics Letters A*, 373(23-24):2001–2006, 2009.

[2] Jan De Leeuw and Patrick Mair. Multidimensional scaling using majorization: Smacof in r. *Department of Statistics, UCLA*, 2011.

[3] Witold Dzwinel and Jan Błasiak. Method of particles in visual clustering of multi-dimensional and large data sets. *Future Generation Computer Systems*, 15(3):365–379, 1999.

[4] Samuel G. Fadel, Francisco M. Fatore, Felipe S.L.G. Duarte, and Fernando V. Paulovich. Loch: A neighborhood-based multidimensional projection technique for high-dimensional sparse spaces. *Neurocomputing*, 150, Part B(0):546–556, 2015.

[5] S.L. France and J.D. Carroll. Two-way multidimensional scaling: A review. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(5):644–661, Sept 2011.

[6] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[7] Stephen Ingram and Tamara Munzner. Dimensionality reduction for documents with nearest neighbor queries. *Neurocomputing*, 150, Part B(0):557–569, 2015.

[8] Stephen Ingram, Tamara Munzner, and Marc Olano. Glimmer: Multilevel mds on the gpu. *Visualization and Computer Graphics, IEEE Transactions on*, 15(2):249–261, March 2009.

[9] Tomoharu Iwata, Neil Houlsby, and Zoubin Ghahramani. Active learning for interactive visualization. *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 342–350, 2013.

[10] Daniel Keim, Huamin Qu, and Kwan-Liu Ma. Big-data visualization. *Computer Graphics and Applications, IEEE*, 33(4):20–21, July 2013.

[11] Robson Motta, Rosane Minghim, Alneu de Andrade Lopes, and Maria Cristina F. Oliveira. Graph-based measures to assist user assessment of multidimensional projections. *Neurocomputing*, 150, Part B(0):583–598, 2015.

[12] Piotr Pawliczek and Witold Dzwinel. Interactive data mining by using multidimensional scaling. *Procedia Computer Science*, 18(0):40–49, 2013. 2013 International Conference on Computational Science.

[13] Piotr Pawliczek, Witold Dzwinel, and David A. Yuen. Visual exploration of data by using multidimensional scaling on multicore cpu, gpu, and mpi cluster. *Concurrency and Computation: Practice and Experience*, 26(3):662–682, 2014.

[14] Piotr Pawliczek, Witold Dzwinel, and David A. Yuen. Interactive data exploration with multithread mic computer architecture. *International Conference on Artificial Intelligence and Soft Computing. ICAISC-2015, June 14-18, Zakopane, Poland*, 2015 (submitted for publication).

[15] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[16] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[17] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, January 2014.

[18] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[19] Laurens JP van der Maaten, Eric O Postma, and H Jaap van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(1-41):66–71, 2009.

[20] Zhao Zhang, Tommy W.S. Chow, and Mingbo Zhao. Trace ratio optimization-based semi-supervised nonlinear dimensionality reduction for marginal manifold visualization. *Knowledge and Data Engineering, IEEE Transactions on*, 25(5):1148–1161, May 2013.