Evaluation of Space Partitioning Data Structures for Nonlinear Mapping

Myasnikov E.V.

Samara State Aerospace University,
Image Processing Systems Institute of the Russian Academy of Sciences
Samara, Russia

mevg@geosamara.ru

ABSTRACT

Nonlinear mapping (Sammon mapping) is a nonlinear dimensionality reduction technique operating on the data structure preserving principle. Several possible space partitioning data structures (vp-trees, kd-trees and cluster trees) are applied in the paper to improve the efficiency of the nonlinear mapping algorithm. At the first step specified structures partition the input multidimensional space, at the second step space partitioning structure is used to build up the list of reference nodes used to approximate calculations. The further steps perform initialization and iterative refinement of the low-dimensional coordinates of objects in the output space using created lists of reference nodes. Analyzed space partitioning data structures are evaluated in terms of the data mapping error and runtime. The experiments are carried out on the well-known datasets.

Keywords

Dimensionality reduction, nonlinear mapping, Sammon mapping, multidimensional scaling, MDS, space partitioning, space decomposition, vp-tree, kd-tree, cluster tree

INTRODUCTION

Nonlinear mapping algorithm [Sam69] also known as a Sammon mapping is one of the most well-known explorative data analysis techniques. It belongs to a more broad class of nonlinear dimensionality reduction techniques operating on the data structure preserving principle.

Nonlinear mapping is widely used in scientific research, and in many areas of production activities.

In signal and image analysis nonlinear mapping has been applied in creation of navigation systems for image and multimedia collections. For example, for digital image collections feature information based on visual characteristics of images is extracted at first. Then the nonlinear mapping algorithm is used to map the images from multidimensional feature space to the navigation space (2 or 3 dimensional). Similar images are placed close to each other in this navigation space and a user can browse the collection due to visual characteristics of images.

Another application of nonlinear mapping is automated segmentation and thematic classification of multispectral satellite image. In this case separate pixels are first clusterized due to its feature information and then the clusters are mapped into two-dimensional space using nonlinear mapping algorithm. In this space similar clusters are placed close to each other that allows user to merge clusters belonging to same segments. Such merged clusters can be later semantically labeled to perform thematic classification of an image.

Nonlinear mapping performs projection from some input multidimensional space to output low-dimensional space for a given set of objects (are often referred to as data points) $O = \{o_1, o_2, \ldots, o_N\}$. We consider the preservation of the data structure as the preservation of the pairwise distances between the objects.

That is the distances $d^*(o_i, o_j)$ between pairs of objects o_i and o_j in the output low-dimensional space should approximate corresponding distances $d(o_i, o_j)$ in the input multidimensional space. It is obvious that such a projection cannot preserve distances exactly and the following data

representation error allows to estimate the quality of the mapping:

$$\varepsilon = \frac{1}{\sum_{\substack{o_i,o_j \in O \\ i<j}} d(o_i,o_j)} \cdot \sum_{\substack{o_i,o_j \in O \\ i<j}} \frac{\left(d(o_i,o_j) - d^*(o_i,o_j)\right)^2}{d(o_i,o_j)} \quad (1)$$

Applying gradient descent approach to minimize the data representation error we obtain the following iterative equation for coordinates $\mathbf{y}_i$ of object o_i in the output low-dimensional space:

$$\mathbf{y}_i(t+1) = \mathbf{y}_i(t) + m \cdot \sum_{o_j \in O} \varsigma(o_i,o_j) \quad (2)$$

where

$$m = \frac{2 \cdot \alpha}{\sum_{\substack{o_i,o_j \in O \\ i<j}} d(o_i,o_j)},$$

$$\varsigma(o_i,o_j) = \frac{d(o_i,o_j) - d^*(o_i,o_j)}{d(o_i,o_j) \cdot d^*(o_i,o_j)} \cdot \left(\mathbf{y}_i - \mathbf{y}_j\right) \quad (3)$$

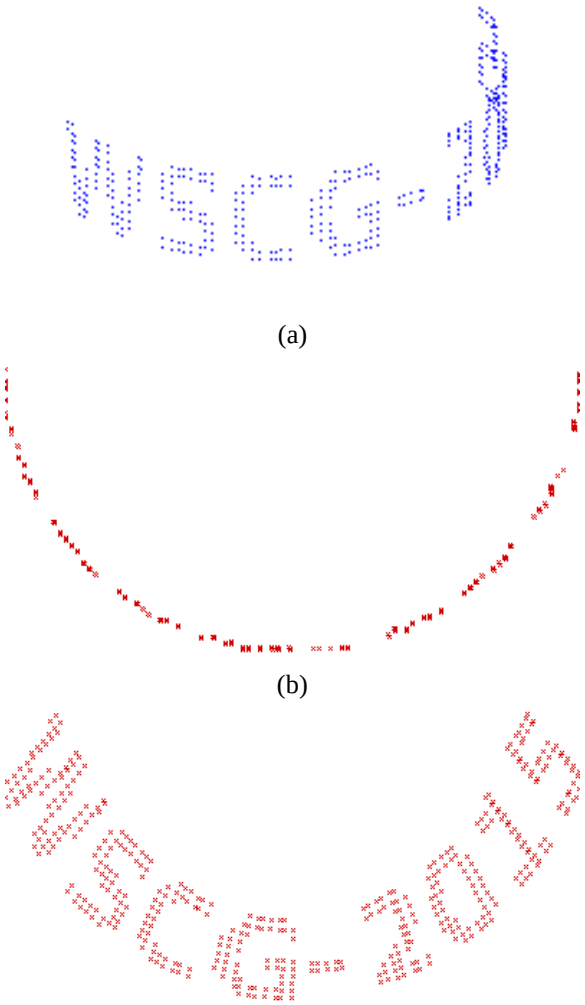
Here t is the number of iteration, α is some coefficient that affects the convergence of the algorithm. The notations in the above equations are different from the ones usually used in literature to simplify the further description.

An example of a nonlinear mapping for a synthetic dataset is shown on fig. 1. The dataset consists of a set of points obtained from the text depicted on a cylinder in 3D space (fig. 1.a). Fig 1.b shows the results obtained using the principal component analysis (PCA). Dataset is mapped on the first two principal components. The results of the described above nonlinear mapping algorithm is shown on the fig. 1.c. As it can be seen the local data structure appears.

Fig. 2 shows a mapping of a set of multi-spectral values (4 spectral bands) of pixels in 3x3 neighborhoods (total 36 attributes) in a LANDSAT satellite image [LAN]. Different classes are shown in different color.

To obtain a solution one should initialize the coordinates $\mathbf{y}_i(0)$ and iteratively refine coordinates of all the objects in accordance to (2) until the coordinates become stable.

Unfortunately, this simple iterative procedure given above has a significant drawback. If the set O contains N objects then the computational complexity is $O(N^2)$ per iteration (the computational complexity for the data representation error is roughly the same).



(a)

(b)

(c)

Figure 1. Mapping for synthetic dataset:

(a) synthetic 3D data; (b) mapping on the first two principal components; (c) nonlinear mapping (data representation error is ε = 0.0046)

The algorithm becomes time consuming for relatively small sets of objects containing hundreds and thousands of objects. The problem becomes more significant for sets containing some tens thousands of objects as the memory needed to store precomputed distances between the objects in the input multidimensional space is *N*(N-1)/2* (having account that the distance matrix is symmetrical). For example, for a set of 20 000 objects we need about 1,5 Gb of operational memory to store the distances as a 8 byte floating point values. Recomputing the distances makes the optimization process even more time consuming and dependent on the dimensionality of the input space. As an example, the time needed to compute the error in accordance to (1) for a MNIST dataset containing 60 000 objects used in this paper for an experimental study takes more than one hour.

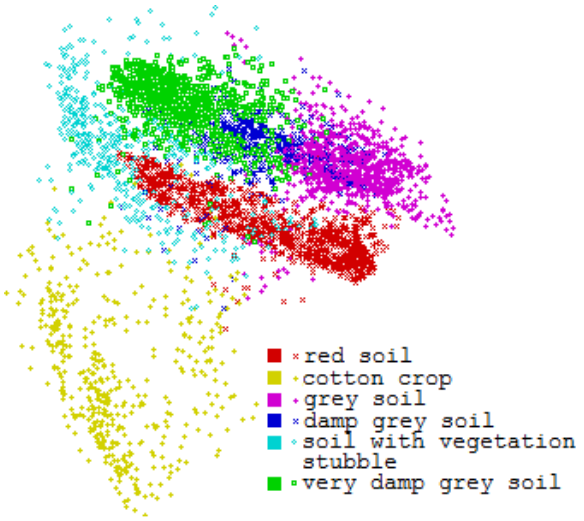


Figure 2. Nonlinear mapping for the Landsat Satellite dataset (data representation error is ε = 0.0177)

One of the most effective and promising approaches to reduce the computational complexity is to perform hierarchical partitioning of space. In the case of the considered problem such partitioning can be performed using different space partitioning structures. This study is devoted to the comparison of several space partitioning structures in terms of the quality of mapping, as well as in terms of operating time.

Related works

To address the problem of a computational complexity of the nonlinear mapping the number of techniques has been proposed. For this purpose the triangulation [Lee77], and linear transformation [Pek99] are used.

Later the performance of the nonlinear mapping has been improved by extending data representation error using left [Sun11a] and right [Sun11b] Bregman divergence.

Taking into account that nonlinear mapping belongs to the multidimensional scaling techniques it is worth mentioning the methods based on the stochastic optimization [Cha96].

The idea of a hierarchical space partitioning to reduce the computational complexity has originated from physics where it is widely used in modelling systems consisting of a huge number of objects (n-body simulations).

The partitioning of the space has been implemented in graph drawing, for example, in [Fru91] (regular decomposition) and [Qui01] (hierarchical decomposition) to approximate the forces acting on the vertices of the graph.

In dimensionality reduction the hierarchical partitioning of the input multidimensional space similar to Barnes-Hut [Bar86] algorithm has been implemented in [Mya12] to improve nonlinear mapping. The hierarchical partitioning of the output low-dimensional space using Barnes-Hut algorithm has been implemented in [Van13, Yan13] to accelerate stochastic neighbor embedding algorithm (t-SNE). In [Vla14] another widely used fast multipole method [Gre87] has been applied to speed up elastic embedding algorithm.

Approximate computations

The main idea of speeding up computations using the space partitioning techniques is to divide the whole set of objects O to some subsets $s_1, s_2, \ldots$ so that all objects $o_i \in s_k$ from some subset s_k should possess similar characteristics. That is all objects from some subset s_k are situated close to each other in the given space. In this case objects in a subset s_k can be analyzed not individually, but as a single object under some circumstances. Assume that the subset s_k is situated far from the object o_i. Then the exact summation

$$\sum_{o_j \in s_k} \frac{d(o_i, o_j) - d^*(o_i, o_j)}{d(o_i, o_j) \cdot d^*(o_i, o_j)} \cdot \left(\mathbf{y}_i - \mathbf{y}_j\right)\Big)$$

can be approximated as follows

$$|s_k| \frac{d(o_i, s_k) - d^*(o_i, s_k)}{d(o_i, s_k) \cdot d^*(o_i, s_k)} \cdot \left(\mathbf{y}_i - \mathbf{y}_{s_k}\right)$$

where $|s_k|$ is the number of objects in the considered subset, $d(o_i, s_k)$ is the distance from the object o_i to the center of the subset s_k in the input space, $d^*(o_i, s_k)$ is the distance from the object o_i to the center of the subset in the output space, $\mathbf{y}_{s_k}$ is the coordinates of the center of the group in the output space.

Now if all the objects of the original set O are divided into subsets $s_k \in S$, then (2) takes the form

$$\mathbf{y}_i(t+1) = \mathbf{y}_i(t) + m \cdot \sum_{s_k \in S} \tilde{\xi}(o_i, s_k), \qquad (4)$$

$$\tilde{\xi}(o_i, s_k) = |s_k| \cdot \frac{d(o_i, s_k) - d^*(o_i, s_k)}{d(o_i, s_k) \cdot d^*(o_i, s_k)} \cdot \left(\mathbf{y}_i - \mathbf{y}_{s_k}\right), \quad (5)$$

It is obvious that the equation (4) allows to approximate (2) with a certain accuracy that depends on how well objects are divided into subsets.

For this reason, there is a question about how to perform such a decomposition.

Description of the Methods

The base scheme for the method described below is taken from [Mya12] with some modifications. In general the considered method consists of the four following steps:

1. Partitioning of the input space
2. Construction of reference nodes lists
3. Initialization in the output space
4. Iterative optimization

The above stages are discussed below in more detail.

Partitioning of the input space

The first step of the original method assumed the hierarchical clustering to partition the input space. This lead to the $O(N^2)$ complexity in the case of effective agglomerative clustering. In the case of divisive method based on the neural network (WTA) that was implemented in [Mya12] the runtime of hierarchical clustering is highly dependent on the parameters of the algorithm.

At the same time there are a number of space partitioning trees that can be built in less time and that used widely in multimedia databases, geographic information systems, information retrieval, computer graphics and so on. The review of such data structures is beyond the scope of this paper. A comprehensive work on space partitioning trees can be found, for example, in [Sam06].

In this work several binary space partitioning trees were used and compared for input space partitioning: kd-tree [Ben75], metric tree (vp-tree) [Uhl91, Yia93], and binary tree based on the simplification of minmax distance clustering approach [Tou74] (further "cluster tree").

Such choice is motivated to study structures that are different in their properties. Kd-tree is one of the old and well-known space partitioning structures based on the recursive splitting a space with hyperplanes orthogonal to the coordinate axes. Vp-tree is another well-known space partitioning structure based on the hypersherical partitioning of a space. Cluster tree is the distance based structure that partitions a space with hyperplanes orthogonal to a pair of chosen distant points.

All the mentioned structures can be built in $O(N \log N)$ operations (if the tree is balanced) that is more suitable for the considered task. As the construction algorithms for the first two structures are well known only the algorithm for latter structure is given below by a pseudo code.

```
Function CreateTree( SetOfObjects O ) returns Node
begin
    Create new node S
    if Number of objects in O is less than threshold then
    begin
        S.Children = O
        return S
    end
    Find mean vector value in O
    Find o1 object farthest from mean
    Find o2 object farthest from o1
    Create new SetOfObjects s1
    Create new SetOfObjects s2
    for each object o in O begin
        if ( d(o1, o) < d( o2, o ) )
            add o to s1
        else
            add o to s2
    end
    Node n1 = CreateTree( s1 )
    Node n2 = CreateTree( s2 )
    add n1 to S.Children
    add n2 to S.Children
    return S
end
```

Let us assume that using some partitioning method (e.g. using function given above) we get the tree-like data structure (fig. 3), containing the objects of the initial set O as leaves (terminal nodes).

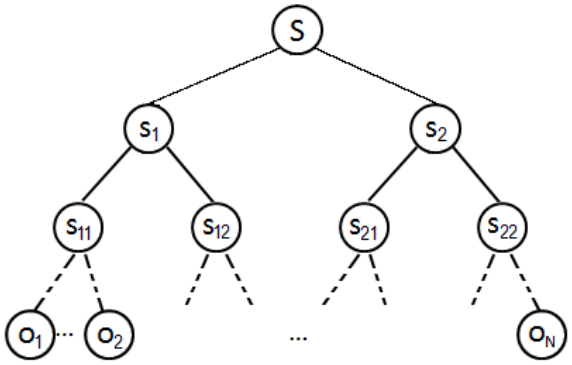


Figure 3. The structure of a binary space partitioning tree

The tree created at the first step of an algorithm can be used in optimization process immediately in the following way. We iteratively update low-dimensional coordinates for each object $o_i \in O$ in accordance to (4), using top-level nodes of the tree if such nodes are far enough from the object o_i, and low-level nodes of the tree or the objects of the initial set O in the other case.

However, the constructed tree is not used directly in the optimization process in this paper. Instead of it the special data structure is used in optimization, which is described in the next subsection. The details of the optimization process are given in "Iterative optimization" subsections.

Construction of reference nodes lists

It is worth noting that as the partitioning is performed in the input multidimensional space then the

partitioning tree is constant in the optimization process.

That is why in [Mya12] it was proposed to calculate and to store the set of nodes and the objects which are involved in the optimization process (called the list of reference nodes) for each object $o_i \in O$ of the initial set. The use of the lists of reference nodes requires additional memory but allows to avoid recalculating the partitioning criterion during the optimization process and allows to store precomputed distances to reference nodes that makes the optimization process independent on the input dimensionality.

The straightforward algorithm for creation of reference nodes is recursive and described below by pseudo code.

```
Procedure CreateRefList( Object o, Node s, RefNodeList rl)
begin
    if o far from s then
        add s to rl
    else if s contains objects then
        for each object a in s
            add a to rl
    else    /* s contains subtrees */
        for each node d contained in s
            CreateRefList( o, d, rl )
end
```

To avoid recursive calls the iterative implementation using special pointers in tree nodes was used in the experiments.

The described above algorithm is slightly different compared to [Mya12] by removing the notion of incomplete node. The use of incomplete nodes allow to slightly accelerate calculations by aggregating some portion of objects in decomposed nodes of the tree but it requires extra memory to store information about aggregated objects.

The first condition "*o* is far from *s*" defines the decomposition criterion of the given node *s* with respect to the object *o*. Different decomposition criterions were described in physics literature (e.g. [Sal94]). And here we adapt (as in [Mya12]) simple decomposition criterion based on the radius *R* of the given node *s*. That is we decompose the node of the space partitioning tree under the following condition:

$$d(s,o)/R(s) \le T$$

where T is a predefined threshold parameter. Otherwise we add the node *s* of the tree to the list of the reference nodes.

Fig. 4 illustrates this process. The node s_1 will be divided into two nodes if $d(s_1,o)/R(s_1) \le T$. The node s_2 will be considered as a single node if $d(s_2,o)/R(s_2) > T$.

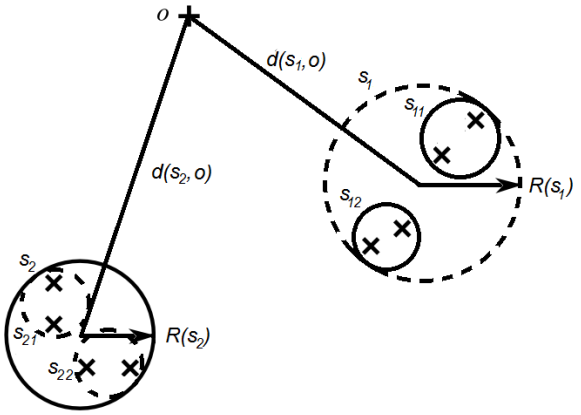


Figure 4. Illustration to the construction of reference nodes list

It is worth noting that the number of the nodes in the list of the reference nodes is dependent on the distribution of the data. Assuming that the list of reference nodes contains on average L nodes the memory needed to store the lists is O[LN].

Initialization in the output space

The third step of the original method performs an initialization of the low-dimensional coordinates of the objects. Only two first principal components are computed to initialize objects in the output space and then input multidimensional coordinates are projected to the plane formed by these components. This approach allows to reduce the computation time compared to random initialization as the next step of iterative optimization process starts with the better initial conditions.

Although principal components finding by the covariance matrix is dependent on the dimensionality of the input space D and can be estimated as O(D^2) per iteration the overall process can be time consuming as covariance matrix finding is O(ND^2) and depends not only on the dimensionality but on the number N of objects also. To reduce the computational time we use only a small subset of randomly selected objects of the initial set O making the overall complexity independent on the number of objects. There are other solutions, e.g. use of neural network approach based on generalized Hebbian (Sanger) learning rule [San89] used in [Mya12].

Iterative optimization

After initializing low-dimensional coordinates of objects the optimization procedure is performed to find sub optimal coordinates of objects in output low-dimensional space. It consists of iterative refinement of output coordinates of all objects in accordance with

$$\mathbf{y}_i(t+1) = \mathbf{y}_i(t) + m \cdot \sum_{s_{ik} \in S_i} \tilde{\boldsymbol{\varsigma}}(o_i, s_{ik})$$

 ISBN 978-80-86943-65-7

where set S_i is the list of reference nodes for object o_i, and subsets s_{ik} are reference nodes for object o_i.

The computational complexity of the optimization stage of the method can be estimated as O(LN) on average where L is the average length of lists of reference nodes.

In practice it may be reasonable to control the data mapping error along the process of optimization as it can indicate the divergence of the process or it may be used in a stop criterion. The computation complexity of the data mapping error (1) is roughly the same as the complexity of one step of the base version of the optimization process (O[N^2]). Using the lists of reference nodes data mapping error can be estimated in O(LN) on the average:

$$\tilde{\varepsilon} = \frac{1}{\sum_{\substack{o_i \in O \\ s_{ik} \in S_i}} d(o_i, s_{ik})} \cdot \sum_{\substack{o_i \in O \\ s_{ik} \in S_i}} \frac{\left(d(o_i, s_{ik}) - d^*(o_i, s_{ik})\right)^2}{d(o_i, s_{ik})}$$

It is worth noting that the above equation may give greatly underestimated values of the data mapping error especially in case when approximation based on the input multidimensional information becomes too coarse. For example, this can take place due to poor configuration of objects in the low-dimensional space.

In the present work this equation was used to control the optimization process. When the estimated error value was increasing then the α coefficient was decreasing until the process became convergent. Also this equation was used to stop the optimization process when the relative decrease in estimated error for a given number of iterations did not exceed the predefined value.

Experimental study

Two well-known datasets were used in the presented study. The first one is the MNIST database of handwritten digits [MNI]. The second one is the Corel Image Features Data Set [COR].

The first database contains digital grayscale images of handwritten digits. The database is divided in two sets: a training set containing 60 000 instances, and test set containing 10 000 instances. Images of the training set with size 28x28 pixels are treated as vectors in 784-dimensional space in the experiments.

The second dataset contains features, calculated from the digital images of the Corel image collection (http://corel.digitalriver.com/). The Corel Image Features Data Set contains 68 040 instances.

The following features have been used in the experiments:

- color histograms [Swa91] constructed in the HSV color space. Color space was divided into 8 ranges of H and 4 ranges of S. The dimensionality of the feature space is 32.

- color moments [Sti95]. Three features were calculated for each color component: mean, standard deviation, and skewness. The dimensionality of the feature space is 9.

- texture features based on co-occurrence matrices [Har73]. Four co-occurrence features (second angular moment, contrast, inverse difference moment, and entropy) were computed in four directions (horizontal, vertical, and two diagonal). The dimensionality of the feature space is 16.

To evaluate the effectiveness of the methods a number of characteristics has been measured and calculated:

- building time of the binary space partitioning tree,

- building time of the list of reference nodes,

- length of the list of reference nodes,

- initialization time of the low-dimensional coordinates,

- per iteration execution time of the optimization procedure,

- multidimensional data representation error.

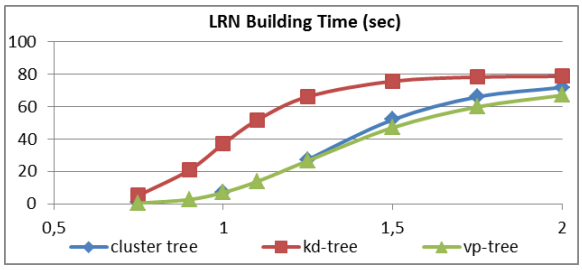
All the described methods were implemented in C++. The studies with the MNIST dataset have been carried out on PC based on Intel Core i5-3470 CPU 3.2 GHz. The studies with the COREL dataset have been carried out on laptop based on Intel Core i3 M370 CPU 2.4 GHz.

The work of the methods stopped when the relative decrease in estimated error for ten iterations did not exceed 0.01. In all cases, the dimension of the target space has been set equal to two (two-dimensional data mapping).

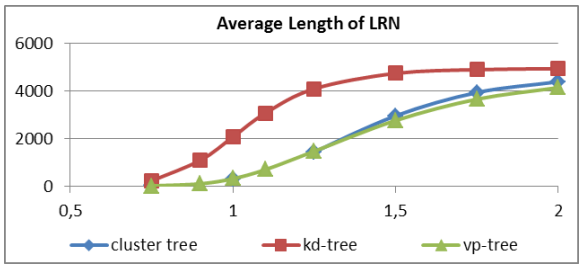
Some results are shown in fig. 5-7.

Fig. 5 shows the dependence of the qualitative and temporal characteristics on the threshold parameter T at which the algorithm moves to the child nodes of the corresponding partitioning structure (metric (vp-) tree, kd-tree or cluster tree).

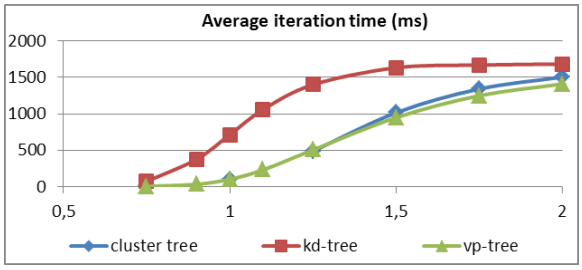
As it can be seen from these results, the small values of T (especially T<1) leads to the expected deterioration of the mapping quality due to a coarser approximation, which is reflected in the higher values of the multidimensional data representation error ε (see fig. 5d). The time it takes to perform a single iteration, increases with increasing T (see fig. 5c), due to the large number of the processed reference nodes of the corresponding hierarchical structure (see fig. 5b).



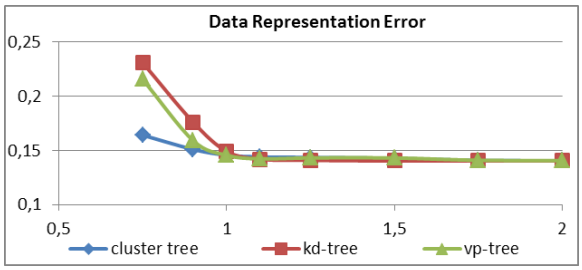
(a)



(b)



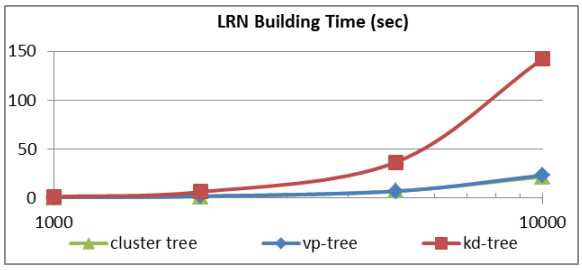
(c)



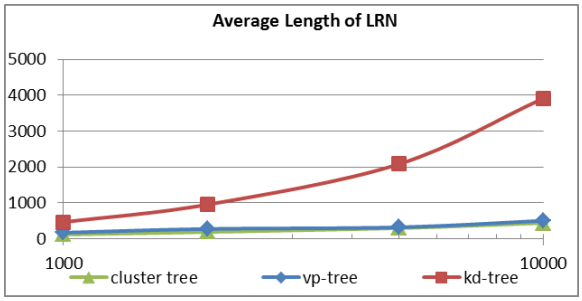
(d)

Figure 5. Dependencies on the threshold parameter *T* (MNIST dataset): (a) average building time of the list of reference nodes (LRN); (b) average length of the list of reference nodes; (c) average time per one iteration of the optimization process; (d) multidimensional data representation error ε

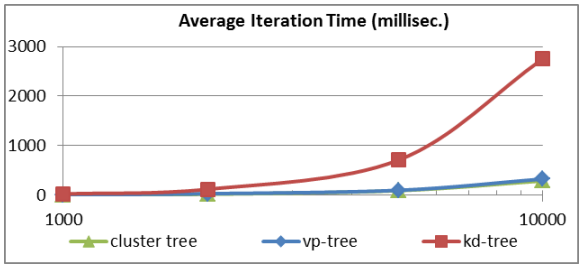
Dependencies of quality and temporal characteristics on the number of objects is shown in the fig. 6 and 7. The quality of mapping, measured by the multidimensional data representation error ε (see fig. 6d, 7d) is weakly dependent on the type of space partitioning structure. At the same time the average length of lists of reference nodes is significantly larger when we use kd-tree (see fig. 6b, 7b). This confirms that kd-trees are poorly suited for multidimensional data processing.



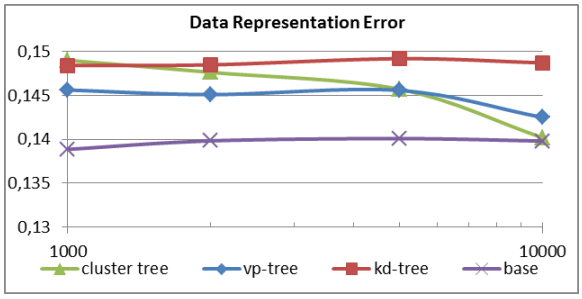
(a)



(b)



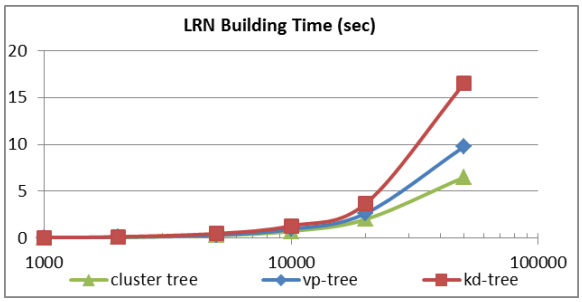
(c)



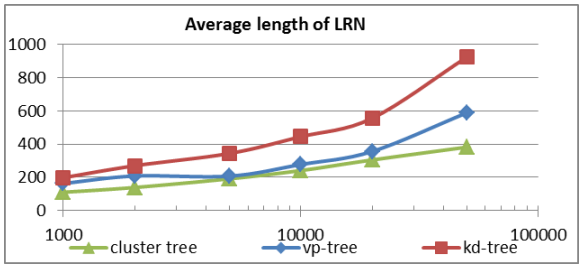
(d)

Figure 6. Dependencies on the sample size (MNIST dataset): (a) average building time of the list of reference nodes (LRN); (b) average length of the list of reference nodes; (c) average time per one iteration of the optimization process; (d) multidimensional data representation error ε

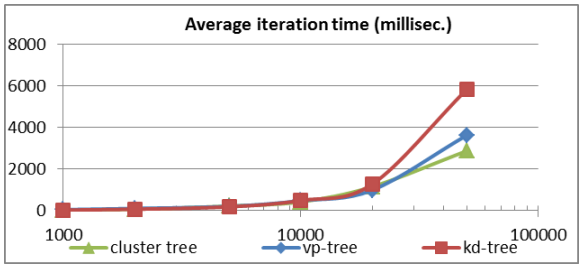
Some results obtained for the base nonlinear mapping algorithm is shown on fig. 8 (logarithmic scale). Timings for the case of precomputed distances are not shown for large sample sizes due to memory limitations. The multidimensional data representation error is shown on fig. 7 for comparison.



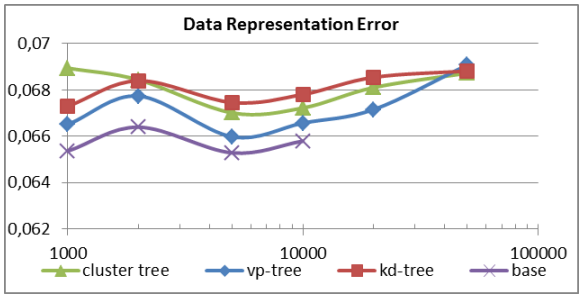
(a)



(b)



(c)



(d)

Figure 7. Dependencies on the sample size (COREL dataset): (a) average building time of the list of reference nodes (LRN); (b) average length of the list of reference nodes; (c) average time per one iteration of the optimization process; (d) multidimensional data representation error ε

As we can see the error values for the base method is only slightly better than error values obtained using the studied methods.

Note that the experiments performed on other datasets described above, show similar results.

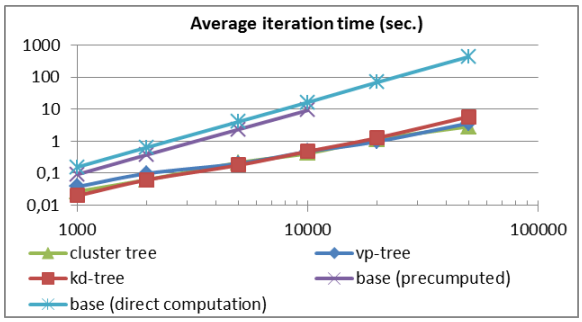


Figure 8. Dependency of the average time per one iteration on the sample size for the base algorithm (COREL dataset)

Some examples of the nonlinear mapping obtained using space partitioning structure and the base algorithm for a subset containing 5000 instances of the MNIST dataset is shown in fig. 9.

An example of the nonlinear mapping using the described approach for 60 000 objects of the MNIST database is shown in fig. 10. The database was processed in less than 30 minutes including error estimation at each iteration (19 minutes without error estimation). The multidimensional data representation error was equal to 0.14.

Conclusion

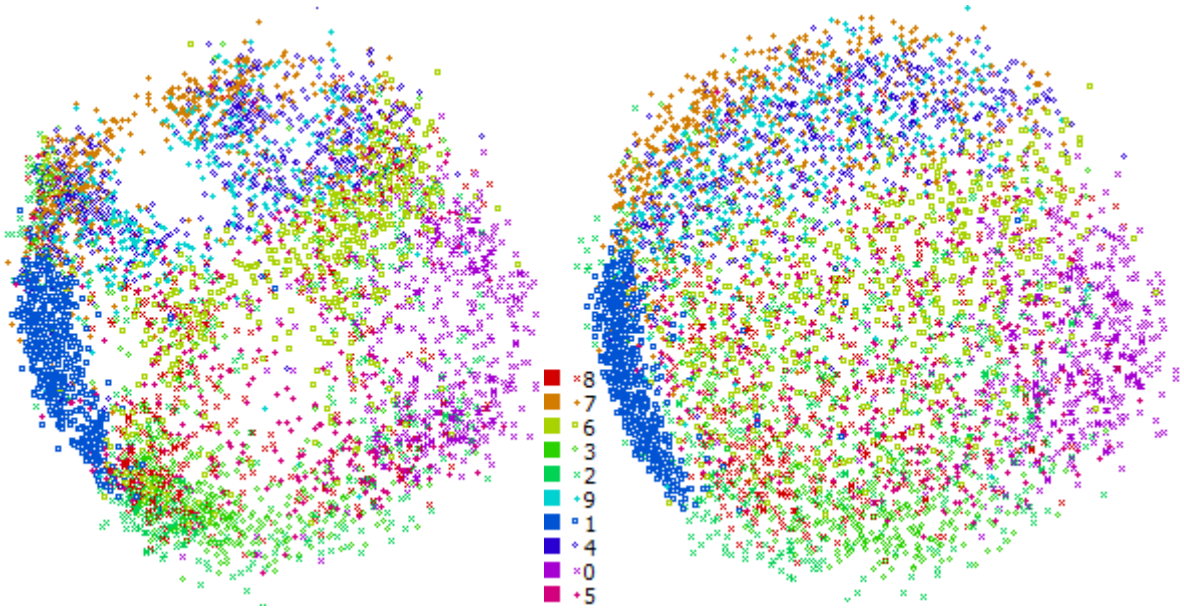
In this paper, we conducted a study of several space partitioning structures namely metric trees (vp-trees), kd-trees, and cluster trees to speed up the nonlinear mapping. The study showed that the quality of mapping is weakly dependent on the type of the structure but the average iteration time was different for the considered structures. For kd-tree the number of reference nodes was significantly larger than for the other structures, hence the average iteration time was larger.

Thus for the presented data sets metric trees (vp-trees) and cluster trees can be efficiently applied to partition the input space for the considered problem. Using of the considered data structures made it possible to generate low-dimensional embeddings for relatively large datasets in a comfortable time.

At the same time the data representation error for the base algorithm had slightly lower values. To improve the quality of the mapping one can use the considered structures increasing the threshold parameter T or use the obtained low-dimensional configuration as an initial configuration for the base method.

ACKNOWLEDGMENTS

This work was financially supported by the Russian Foundation for Basic Research, project № 15-07-01164-a.



(a) **(b)**

Figure 9. Nonlinear mapping for 5000 instances of MNIST database:
(a) cluster tree, 207 iterations, threshold T = 1, multidimensional data representation error ε = 0.14255;
(b) base method, 194 iterations, multidimensional data representation error ε = 0.14077

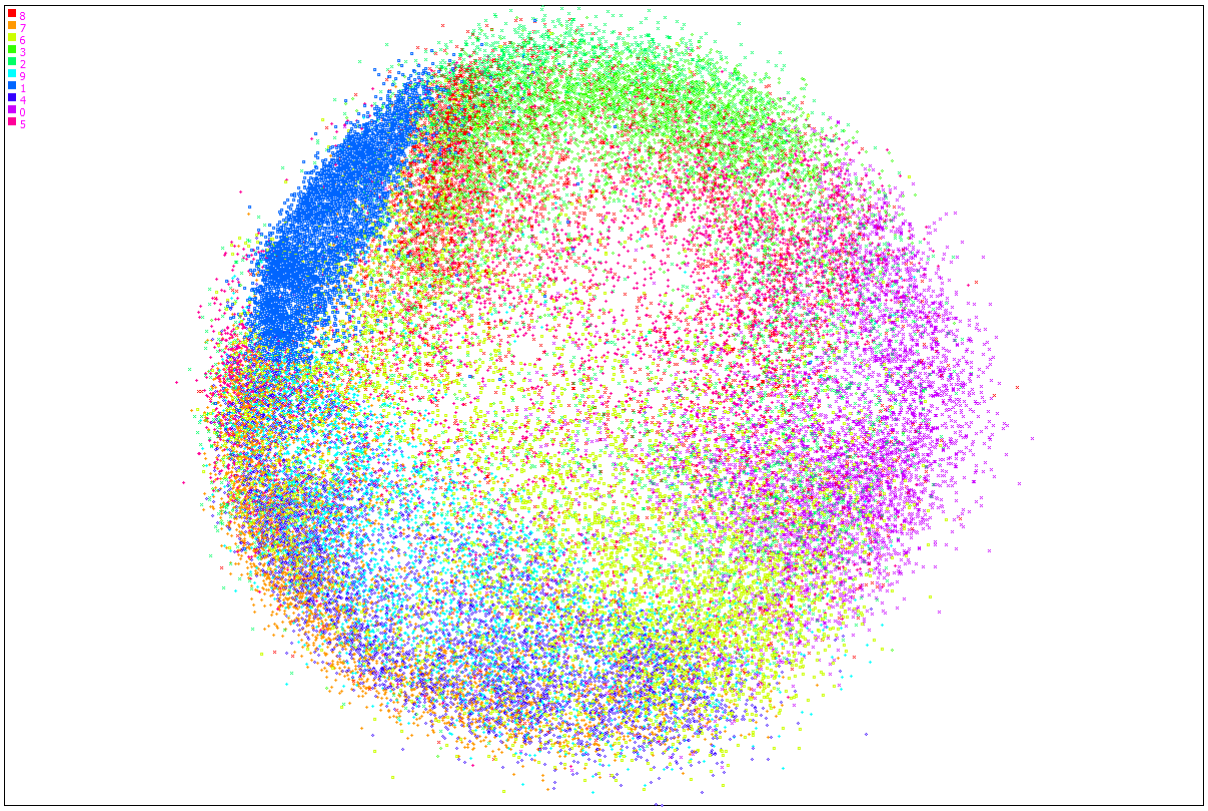


Figure 10. Nonlinear mapping for training set of MNIST database (60,000 examples)

 ISBN 978-80-86943-65-7

REFERENCES

[Bar86] Barnes J., Hut P. A hierarchical O(N log N) force-calculation algorithm // Nature, 324 (4). – 1986. – pp. 446–449.

[Ben75] Bentley J. L. Multidimensional binary search trees used for associative searching // Communications of the ACM, 18 (9). – 1975. – 509

[Cha96] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data // In Proceedings of IEEE Visualization. – 1996. – pp. 127–132.

[COR] https://archive.ics.uci.edu/ml/datasets/Corel+Image+Features

[Fin74] Finkel R., Bentley J.L. Quad Trees: A Data Structure for Retrieval on Composite Keys. – 1974. - Acta Informatica, 4 (1). – pp. 1–9.

[Fru91] Fruchterman T., Reingold E. Graph Drawing by Force-directed Placement. //Software – Practice and Experience. 1991. vol. 21, no. 11. pp. 1129-1164.

[Gre87] Greengard L., Rokhlin V. A fast algorithm for particle simulations. Journal of Computational Physics, 73. – 1987. – pp. 325–348.

[Har73] Haralick R.M., Shanmugam K., Dinstein I. Texture features for image classification. IEEE Trans. on Sys. Man. and Cyb. SMC-3(6), 1973

[LAN] https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)

[Lee77] Lee R.C.T., Slagle J.R., Blum H. A Triangulation Method for the Sequential Mapping of Points from N-Space to Two-Space //IEEE Transactions on Computers. – 1977. - V. 26, №3. - pp. 288-292.

[MNI] http://yann.lecun.com/exdb/mnist/

[Mya12] E.V. Myasnikov A Nonlinear Method for Dimensionality Reduction of Data Using Reference Nodes // Pattern Recognition and Image Analysis, 2012, Vol. 22, No. 2, pp. 337–345.

[Pek99] Pekalska E., de Ridder D., Duin R.P.W., Kraaijveld M.A. A new method of generalizing Sammon mapping with application to algorithm speed-up. //Proc. ASCI'99, 5th Annual Conf. of the Advanced School for Computing and Imaging - Heijen, The Netherlands. – 1999. - June 15-17. - P. 221-228.

[Qui01] Quigley A., Eades P. FADE: Graph Drawing, Clustering, and Visual Abstraction". //Proceedings of the 8th International Symposium on Graph Drawing. 2001. pp. 197–210.

[Sal94] Salmon J.K., Warren M.S. Skeletons from the Treecode Closet // J. Comp. Phys. V.111 – 1994. – pp. 136-155.

[Sam06] Samet, H. Foundations of multidimensional and metric data structures // Morgan Kaufmann. – 2006. – 1024 p.

[Sam69] Sammon J.W., Jr. A nonlinear mapping for data structure analysis. //IEEE Transactions on Computers. – 1969. - V. C-18, No.5. - P.401-409.

[San89] Sanger T.D. Optimal unsupervised learning in a single-layer linear feedforward neural network // Neural Networks, 2 (6). – 1989. – pp. 459–473.

[Sti95] Stricker M., Orengo M. Similarity of color images // In Proc. SPIE Conf. on Vis. Commun. and Image Proc., 1995

[Swa91] Swain M., Ballard D. Color indexing. International Journal of Computer Vision. 7(1), 1991.

[Tou74] Tou J.T., Gonzalez R.C. Pattern Recognition Principles // Addison-Wesley Publishing Company. – 1974.

[Uhl91] Uhlmann J. Satisfying General Proximity/Similarity Queries with Metric Trees // Information Processing Letters, 40 (4). - 1991.

[Van13] van der Maaten L.J.P. Barnes-Hut-SNE // In Proceedings of the International Conference on Learning Representations. - 2013.

[Vla14] Vladymyrov M., Carreira-Perpiñán, M.Á. Linear-time training of nonlinear low-dimensional embeddings // 17th International Conference on Artificial Intelligence and Statistics (AISTATS 2014) – 2014. - pp. 968-977.

[Yan13] Z. Yang, J. Peltonen, and S. Kaski. Scalable optimization of neighbor embedding for visualization // In Proc. of the Int. Conf. on Machine Learning. - 2013.

[Yia93] Yianilos Data structures and algorithms for nearest neighbor search in general metric spaces // Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms. - 1993. - pp. 311–321.