J Math Model Algor (2008) 7:351–370
DOI 10.1007/s10852-008-9092-y

Topology Representing Networks for the Visualization of Manifolds

Agnes Vathy-Fogarassy · Agnes Werner-Stark · Janos Abonyi

Received: 26 April 2008 / Accepted: 23 October 2008 /
Published online: 19 November 2008

Abstract In practical data mining tasks, high-dimensional data has to be analyzed. In most of the cases it is very informative to map and visualize the hidden structure of a complex data set in a low-dimensional space. In this paper a new class of mapping algorithms is defined. These algorithms combine topology representing networks and different nonlinear mapping algorithms. While the former methods aim to quantify the data and disclose the real structure of the objects, the nonlinear mapping algorithms are able to visualize the quantized data in the low-dimensional vector space. In this paper, techniques based on these methods are gathered and the results of a detailed analysis performed on them are shown. The primary aim of this analysis is to examine the preservation of distances and neighborhood relations of the objects. Preservation of neighborhood relations was analyzed both in local and global environments. To evaluate the main properties of the examined methods we show the outcome of the analysis based both on synthetic and real benchmark examples.

Keywords Mapping algorithm · Topology representing networks · Visualization of manifolds

A. Vathy-Fogarassy (✉) · A. Werner-Stark
Department of Mathematics and Computing, University of Pannonia,
P.O.Box 158, Veszprem, 8201 Hungary,
e-mail: vathya@almos.uni-pannon.hu

J. Abonyi
Department of Process Engineering, University of Pannonia,
P.O.Box 158, Veszprem, 8201 Hungary
e-mail: abonyij@fmt.uni-pannon.hu
URL: http://www.fmt.uni-pannon.hu/softcomp

1 Introduction

In practical data mining problems high-dimensional data has to be analyzed. Objects to be analyzed are characterized with n variables, and each variable corresponds to a dimension of the vector space. Because humans can not see high-dimensional data, it is useful to map the high-dimensional data points into a low-dimensional vector space.

The goal of *dimensionality reduction* is to map a set of observations from a high-dimensional space (D) into a low-dimensional space (d, $d \ll D$) preserving as much of the intrinsic structure of the data as possible. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ be a set of the observed data, where $\mathbf{x}_i$ denotes the i-th observation ($\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]^T$). Each data object is characterized by D dimensions, so $x_{i,j}$ yields the j-th ($j = 1, 2, \ldots, D$) attribute of the i-th ($i = 1, 2, \ldots, N$) data object. Dimensionality reduction techniques transform data set $\mathbf{X}$ into a new data set $\mathbf{Y}$ with dimensionality d ($\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$, $\mathbf{y}_i = [y_{i,1}, y_{i,2}, \ldots, y_{i,d}]^T$).

Three types of dimensionality reduction methods can be distinguished: (i) metric methods that try to preserve the distances of the data defined by a metric, (ii) non-metric methods that try to preserve the global ordering relations of the data, and (iii) other methods that differ from the previously introduced two groups.

One of the most widely applied dimensionality reduction methods is *Principal Component Analysis* (PCA) [13]. The PCA algorithm is also known as Hotteling or as Karhunen-Loéve transform ([12, 13]). PCA differs from the metric and non-metric dimensionality reduction methods, because instead of the preservation of the distances or the global ordering relations of the objects it tries to preserve the variance of the data. PCA represents the data as linear combinations of a small number of basis vectors. This method finds the projection that stores the largest variance possible in the original data and rotates the set of the objects such that the maximum variability becomes visible. Geometrically, PCA transforms the data into a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on. *Independent Component Analysis* (ICA) [4] is similar to PCA, except that it tries to find components that are independent.

Multidimensional scaling (MDS) [3] refers to a group of unsupervised data visualization techniques. There are two types of MDS: (i) *metric MDS* and (ii) *non-metric MDS*. As MDS has a metric and a non-metric variant, it can also preserve the pairwise distances or the rank ordering among the data objects.

The *metric MDS* discovers the underlying structure of data set by preserving similarity information (pairwise distances) among the data objects. The metric multidimensional scaling tries to optimize the cost function that describes how well the pairwise distances in a data set are preserved. If the square-error cost is used, the objective function (stress) to be minimized can be written as:

$$E_{metric\_MDS} = \frac{1}{\sum_{i<j}^{N} d_{i,j}^{*2}} \sum_{i<j}^{N} \left(d_{i,j}^{*} - d_{i,j}\right)^2, \tag{1}$$

where $d_{i,j}^{*}$ denotes the distance between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and $d_{i,j}$ between $\mathbf{y}_i$ and $\mathbf{y}_j$ respectively. N yields the number of the objects to be mapped.

The metric MDS algorithm is an algebraic method that rests on the fact that matrix $\mathbf{Y}$ containing the output coordinates can be derived by eigenvalue decomposition from the scalar product matrix $\mathbf{B} = \mathbf{Y}\mathbf{Y}^T$. Matrix $\mathbf{B}$ can be found from the known distances using the Young-Householder process [29].

In *non-metric MDS*, only the ordinal information of the proximities is used for constructing the spatial configuration, thereby the non-metric MDS attempts to preserve the rank order among the dissimilarities. The non-metric MDS finds a configuration of points whose pairwise Euclidean distances have approximately the same rank order as the corresponding dissimilarities of the objects. Equivalently, the non-metric MDS finds a configuration of points, whose pairwise Euclidean distances approximate a monotonic transformation of the dissimilarities. These transformed values are known as the disparities. The non-metric MDS stress can be formulated as follows:[1]

$$E_{nonmetric\_MDS} = \sqrt{\sum_{i<j}^{N} \left(\widehat{d}_{i,j} - d_{i,j}\right)^2 / \sum_{i<j}^{N} d_{i,j}^2}, \tag{2}$$

where $\widehat{d}_{i,j}$ yields the disparity of $\mathbf{x}_i$ and $\mathbf{x}_j$, and $d_{i,j}$ denotes the distance between the vectors $\mathbf{y}_i$ and $\mathbf{y}_j$.

The main steps of the non-metric MDS algorithm are given in Algorithm 1.

Algorithm 1 Non-metric MDS algorithm

Step 1	Find a random configuration of points in the output space.
Step 2	Calculate the distances between the points.
Step 3	Find the optimal monotonic transformation of the proximities in order to obtain the disparities.
Step 4	Minimize the non-metric MDS stress function by finding a new configuration of points.
Step 5	Compare the stress to some criteria. If the stress is not enough small then go back to Step 2.

It can be shown, that the metric and non-metric MDS mappings are substantially different methods. While the metric MDS algorithm is an algebraic method, the non-metric MDS is an iterative mapping process.

Sammon mapping (SM) [22] is a nonlinear mapping method, that is closely related to the metric MDS version described above. The aim of Sammon mapping is to find the low-dimensional presentation of the input data set, such a way that the corresponding distances approximate the original ones as well as possible. The Sammon stress function can be written as:

$$E_{SM} = \frac{1}{\sum_{i<j}^{N} d_{i,j}^*} \sum_{i<j}^{N} \frac{\left(d_{i,j}^* - d_{i,j}\right)^2}{d_{i,j}^*}, \tag{3}$$

[1] Traditionally, the non-metric MDS stress is often called Stress-1 due to Kruskal [16].

where N is the number of the objects to be mapped, $d^*_{i,j}$ denotes the distance between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and $d_{i,j}$ respectively for $\mathbf{y}_i$ and $\mathbf{y}_j$. The only difference between the stress functions of the Sammon mapping and the metric MDS (see Eq. 1) is, that the errors in distance preservation are normalized by the distances of the input data objects. Because of this normalization the Sammon mapping emphasizes the preservation of small distances.

The minimization of the Sammon stress is an optimization problem. When the gradient-descent method is applied to search for the minimum of Sammon stress, a local minimum can be reached. Therefore a significant number of runs with different random initializations may be necessary.

In the literature there are several *neural networks* proposed to visualize high-dimensional data in low-dimensional space. The *Self-Organizing Map* (SOM) [15] is one of the most popular artificial neural networks. The main disadvantage of SOM is that it maps the data objects into a topological ordered grid, thereby it is needed to utilize complementary methods (coloring scheme such as U-matrix) to visualize the relative distances between data points on the map.

The *Visualization Induced Self-Organizing Map* (ViSOM) [28] is an effective extension of SOM. ViSOM is an unsupervised learning algorithm, which is proposed to directly preserve the local distance information on the map. ViSOM preserves the inter-point distances as well as the topology of data, therefore it provides a direct visualization of the structure and distribution of the data. ViSOM constrains the lateral contraction forces between neurons and hence regularizes the interneuron distances so that distances between neurons in the data space are in proportion to those in the input space [28]. The motivation of the development of the ViSOM algorithm was similar to the motivation of our work, but here the improvement of the Topology Representing Network based data visualization techniques are in focus.

Dimensionality reduction methods in many cases are confronted with *low-dimensional structures nonlinearly embedded in the high-dimensional space*. In these cases the Euclidean distance is not suitable to compute distances among the data points. The *geodesic distance* [2] is more suitable to catch the pairwise distances of objects lying on a manifold, because it is computed in such a way that it always goes along the manifold. To compute the geodesic distances a graph should be built on the data.

There are two basic variations to construct the *neighborhood graphs* of the objects: (i) ϵ-neighboring and (ii) k-neighboring. In the case of the first approach objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected by an edge if $d(\mathbf{x}_i, \mathbf{x}_j) < \epsilon$, where the distance of the objects $\mathbf{x}_i$ and $\mathbf{x}_j$ is calculated as the Euclidean distance, and ϵ is a small real number. In the second case objects $\mathbf{x}_i$ and $\mathbf{x}_j$ are connected by an edge if $\mathbf{x}_i$ is among the k-nearest neighbors of $\mathbf{x}_j$ or $\mathbf{x}_j$ is among the k-nearest neighbors of $\mathbf{x}_i$. The edges of the graph are weighted with their Euclidean distance, so the geodesic distance is obtained as the shortest path for each pair of points: given a set of objects ($\mathbf{X}$), the geodesic distance between two objects $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ is the sum of the length of the shortest paths of $\mathbf{X}$ joining $\mathbf{x}_i$ and $\mathbf{x}_j$.

Although most of the algorithms utilize the neighborhood graphs for the construction of the representative graph of the data set (e.g. Isomap [24], Curvilinear Component Analysis (CCA) [5], Curvilinear Distance Analysis (CDA) [17]), there are other possibilities to disclose the topology of the data. *Topology representing networks* refers to a group of methods that generate compact, optimal topology preserving maps for different data sets. Topology representative methods combine

the neural gas (NG) [19] vector quantization method and the competitive Hebbian learning rule [11]. For a given data distribution, first a cloud is created by running the neural gas algorithm and then the topology is generated by the competitive Hebbian learning rule. The methods generate their topology map as a result.

There are many methods published in the literature proposing to capture the topology of the given data set. Martinetz and Shulten [20] showed how the simple competitive Hebbian rule forms Topology Representing Network (TRN). Dynamic Topology Representing Networks (DTRN) were introduced by Si at al. [23]. In their method the topology graph incrementally grows by adding and removing edges and vertices. Weighted Incremental Neural Network (WINN) [21] produces a weighted connected net. This net consists of weighted nodes connected by weighted edges.

The aim of this paper is to analyze the different topology representing network based data visualization techniques. Although topology representing networks are able to disclose the structure of the data set to be analyzed, they are not able to visualize the disclosed structure in a vector space with a dimension lower than the initial dimension of the objects. Therefore, the topology representing network based data visualization methods include two major parts: (1) they create a topology network for disclosing the data structure, and (2) they utilize a dimensionality reduction method for the low-dimensional representation. These two steps can be performed simultaneously and sequentially, as well. In this paper we summarize topology representing networks based visualization techniques, we specify their implementation methods and perform an analysis on them. The analysis compares the mapping qualities in the local environment of the objects and the global mapping properties.

The organization of this paper is as follows. Section 2 introduces the Topology Representing Network, and Section 3 gives an overview of the related mapping methods. Section 4 introduces the measurement of the mapping quality and gives application examples to show the results of the analysis. Section 5 concludes the paper.

2 Topology Representing Network

Although, the Topology Representing Network (TRN), Dynamic Topology Representing Network (DTRN) and Weighted Incremental Neural Network (WINN) algorithms are quite similar, the TRN algorithm [20] gives the most robust representation of the data. In the following we introduce this method in detail.

Given are a set of data ($\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$) and a set of codebook vectors ($\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$, $\mathbf{w}_i \in \mathbb{R}^D$, $i = 1, \ldots, n$) ($N > n$). The algorithm of the Topology Representing Network (TRN) [20] distributes the pointers $\mathbf{w}_i$ between the data objects by the neural gas algorithm, and forms connections between them by applying the competitive Hebbian rule. The TRN algorithm firstly selects some random points (units) in the input space. The number of units (n) is a predefined parameter. The algorithm then iteratively selects an object from the input data set randomly and moves all units closer to this pattern. After this step, the two units closest to the randomly selected input pattern will be connected. Finally, edges

exceeding a predefined age (iteration count) are removed. This iteration process is continued until a termination criterion is satisfied. The run of the algorithm results in a Topology Representing Network that means a graph $G = (\mathbf{W}, \mathbf{C})$, where $\mathbf{W}$ denotes the nodes (codebook vectors, neural units, representatives) and $\mathbf{C}$ yields the set of edges between them. The detailed description of the TRN algorithm can be found in Algorithm 2.

Algorithm 2 TRN algorithm

Step 1 Initialize the codebook vectors $\mathbf{w}_j$ $(j = 1, \ldots, n)$ randomly. Set all connection strengths $c_{i,j}$ to zero. Set $t = 0$.

Step 2 Select an input pattern $\mathbf{x}_i(t)$, $(i = 1, \ldots, N)$ with equal probability for each $\mathbf{x} \in \mathbf{X}$.

Step 3 Determine the ranking $r_{i,j} = r(\mathbf{x}_i(t), \mathbf{w}_j(t)) \in \{0, 1, \ldots, n-1\}$ for each codebook vector $\mathbf{w}_j(t)$ with respect to the vector $\mathbf{x}_i(t)$ by determining the sequence $(j_0, j_1, \ldots, j_{n-1})$ with

$$\|\mathbf{x}_i(t) - \mathbf{w}_{j_0}(t)\| < \|\mathbf{x}_i(t) - \mathbf{w}_{j_1}(t)\| < \ldots < \|\mathbf{x}_i(t) - \mathbf{w}_{j_{n-1}}(t)\|. \tag{4}$$

Step 4 Update the codebook vectors $\mathbf{w}_j(t)$ according to the neural gas algorithm by setting

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \varepsilon \cdot e^{-\frac{r(\mathbf{x}_i(t), \mathbf{w}_j(t))}{\lambda(t)}} \left(\mathbf{x}_i(t) - \mathbf{w}_j(t)\right), \quad j = 1, \ldots, n \tag{5}$$

Step 5 If a connection between the first and the second closest codebook vector to $\mathbf{x}_i(t)$ does not exist already ($c_{j_0,j_1} = 0$), create a connection between them by setting $c_{j_0,j_1} = 1$ and set the age of this connection to zero by $t_{j_0,j_1} = 0$. If this connection already exists ($c_{j_0,j_1} = 1$), set $t_{j_0,j_1} = 0$, that is, refresh the connection of the codebook vectors j_0 - j_1.

Step 6 Increment the age of all connections of $\mathbf{w}_{j_0}(t)$ by setting $t_{j_0,l} = t_{j_0,l} + 1$ for all $\mathbf{w}_l(t)$ with $c_{j_0,l} = 1$.

Step 7 Remove those connections of codebook vector $\mathbf{w}_{j_0}(t)$ the age of which exceed the parameter T by setting $c_{j_0,l} = 0$ for all $\mathbf{w}_l(t)$ with $c_{j_0,l} = 1$ and $t_{j_0,l} > T$.

Step 8 Increase the iteration counter $t = t + 1$. If $t < t_{max}$ go back to Step 2.

The algorithm has many parameters. The number of the iterations (t_{max}) and the number of the codebook vectors (n) are determined by the user. Parameter λ, step size ε and lifetime T are dependent on the number of the iterations. This time dependence can be expressed by the following general form:

$$g(t) = g_i \left(\frac{g_f}{g_i}\right)^{t/t_{max}} \tag{6}$$

where g_i denotes the initial value of the variable, g_f denotes the final value of the variable, t denotes the iteration counter, and t_{max} denotes the maximum number of iterations. (For example for parameter λ it means: $\lambda(t) = \lambda_i(\lambda_f/\lambda_i)^{t/t_{max}}$.) Paper [20] gives good suggestions to tune these parameters.

3 Topology Representing Network Based Mapping Algorithms

In the literature, there are only a few methods published that utilize the topology representing networks to visualize the data set in the low-dimensional vector space. Online Visualization Neural Gas (OVI-NG) [7] is a nonlinear projection method, in which the codebook positions are adjusted in a continuous output space by using an adaptation rule that minimizes a cost function that favors the local distance preservation. As OVI-NG utilizes Euclidean distances to map the data set, it is not able to disclose the nonlinearly embedded data structures. The Geodesic Nonlinear Projection Neural Gas (GNLP-NG) [8] algorithm is an extension of OVI-NG, that uses geodesic distances instead of the Euclidean ones. Abreast with these algorithms, the authors of this paper have developed a new group of the mapping methods, called Topology Representing Network Map (TRNMap) [25, 26]. TRNMap also utilizes the Topology Representing Network and the resultant graph is mapped by MDS into a low-dimensional vector space. Hence TRNMap utilizes geodesic distances during the mapping process, it is a nonlinear mapping method, which focuses on the global structure of data. As the OVI-NG is not able to disclose the nonlinearly embedded manifolds, in the following, we will not deal with this method.

3.1 Geodesic Nonlinear Projection Neural Gas

While the TRN algorithm creates only the topology representing network of the objects, the GNLP-NG method maps this representative graph as well. The GNLP-NG algorithm is a nonlinear mapping procedure, which includes the following two major steps: (1) creating a topology representing network to depict the structure of the data set, and (2) mapping this approximate structure into a low-dimensional vector space.

The first step utilizes a modified TRN algorithm for building the connectivity graph of the representative codebook vectors. The applied combination of the neural gas method and the Hebbian rule differs slightly from the TRN algorithm: it connects not only the first and the second closest codebook vectors to the randomly selected input pattern ($\mathbf{x}_i(t)$) (Step 5 in the TRN algorithm (see Algorithm 2)), but it creates other edges, as well. The establishment of these complementary edges can be formalized in the following way:

– for $k = 2, \ldots, k_{max}$
  Create a connection between the k-th nearest unit ($\mathbf{w}_{j_k}$) and the $k+1$-th nearest unit ($\mathbf{w}_{j_{k+1}}$), if it does not exist already, and the following criterion is satisfied:

$$\|\mathbf{w}_{j_k} - \mathbf{w}_{j_{k+1}}\| < \|\mathbf{w}_{j_0} - \mathbf{w}_{j_{k+1}}\| \tag{7}$$

  Else create a connection between codebook vectors $\mathbf{w}_{j_0}$ and $\mathbf{w}_{j_{k+1}}$.
  Set the ages of established connections to zero. If the connection already exists, refresh the age of the connection by setting its age to zero.

The parameter k_{max} is an accessory parameter compared to the TRN algorithm. In [8] it is suggested to set $k_{max} = 2$. This extra step amends the 5-th step of the TRN algorithm.

Furthermore GNLP-NG increments not only the ages of all connections of $\mathbf{w}_{j_0}$ (see Step 6 in Algorithm 2), but it also extends this step to the k-th nearest unit as follows:

– Increment the age of all edges emanating from $\mathbf{w}_{j_k}$, for $k = 1, \ldots, k_{max}$:

$$t_{j_0,l} = t_{j_0,l} + 1, \forall l \in N_{\mathbf{w}_{j_k}}, \tag{8}$$

where $N_{\mathbf{w}_{j_k}}$ is the set of all direct topological neighbors of $\mathbf{w}_{j_k}$.

This extra step amends the 6-th step of the TRN algorithm (see Algorithm 2).

During the mapping process (second major part of the GNLP-NG algorithm) the GNLP-NG algorithm applies an adaptation rule for the codebook positions in the projection space. It minimizes the following cost function:

$$E_{GNLP-NG} = \frac{1}{2} \sum_{j=1}^{n} \sum_{k \neq j} \left(d_{j,k} - \delta_{j,k}\right)^2 e^{-\left(\frac{\bar{r}_{j,k}}{\sigma(t)}\right)^2}, \tag{9}$$

where $\bar{r}_{j,k} = \bar{r}(\mathbf{x}_j, \mathbf{w}_k) \in \{0, 1, ..., n-1\}$ denotes the rank of the k-th codebook vector with respect to the $\mathbf{x}_j$ using geodesic distances, and σ is a width of the neighborhood surround. $d_{j,k}$ denotes the Euclidean distance of the codebook positions $\mathbf{y}_j$ and $\mathbf{y}_k$ defined in the output space, $\delta_{j,k}$ yields the geodesic distance between codebook vectors $\mathbf{w}_j$ and $\mathbf{w}_k$ measured in the input space.

According to the previously presented overview, the GNLP-NG first determines the topology of the data set by the modified TRN algorithm and then maps this topology based on the graph distances. The whole process is summarized in Algorithm 3.

Algorithm 3 GNLP-NG algorithm

Step 1 Determine the topology of the data set based on the modified TRN algorithm.

Step 2 Compute the geodesic distances between the codebook vectors based on the connections ($c_{i,j}$) of the previously calculated topology representing network. Set $t = 0$.

Step 3 Initialize the codebook positions $\mathbf{y}_j$, randomly.

Step 4 Select an input pattern $\mathbf{x}_i(t)$ with equal probability for each $\mathbf{x} \in \mathbf{X}$.

Step 5 Find the codebook vector $\mathbf{w}_{j_0}(t)$ in the input space that is closest to $\mathbf{x}_i(t)$.

Step 6 Generate the ranking using geodesic distances in the input space $\bar{r}_{j_0,j} = \bar{r}(\mathbf{w}_{j_0}(t), \mathbf{w}_j(t)) \in \{0, 1, \ldots, n-1\}$ for each codebook vector $\mathbf{w}_j(t)$ with respect to $\mathbf{w}_{j_0}(t)$.

Step 7 Update the codebook positions in the output space:

$$\mathbf{y}_j(t+1) = \mathbf{y}_j(t) + \alpha(t) e^{-\left(\frac{\bar{r}_{j_0,j}}{\sigma(t)}\right)^2} \frac{(d_{j_0,j} - \delta_{j_0,j})}{d_{j_0,j}} (\mathbf{y}_{j_0}(t) - \mathbf{y}_j(t)) \tag{10}$$

Step 8 Increase the iteration counter $t = t + 1$. If $t < t_{max}$ go back to Step 4.

In this algorithm the parameter α is the learning rate, σ is the width of the neighborhood, and they typically decrease with the number of iterations, t, in the same way as Eq. 6. $D_{j,k}$ denotes the Euclidean distance of the codebook positions $\mathbf{z}_j$ and $\mathbf{z}_k$ defined in the output space. The codebook positions mean the mapped codebook vectors in the output space. $\delta_{j,k}$ yields the geodesic distance between codebook vectors $\mathbf{w}_j$ and $\mathbf{w}_k$ measured in the input space, and m_j yields the ranking value of the codebook vector $\mathbf{w}_j$. Paper [8] gives an extension to the GNLP-NG, to tear or cut the graphs with non-contractible cycles.

3.2 Topology Representing Network Map

Topology Representing Network Map (TRNMap) refers to a group of nonlinear mapping methods, which combines the TRN algorithm and the MDS method to visualize the data structure to be analyzed. It results in a new visualization map, called Topology Representing Network Map (TRNMap). TRNMap is a self-organizing model with no predefined structure which provides an expressive presentation of high-dimensional data in low-dimensional vector space. The dimensionality of the input space is not restricted. Although we can get arbitrary dimensional output map as a result, in this paper the 2-dimensional output map is recommended.

The method for constructing the TRNMap (TRNMap algorithm) is based on graph distances, therefore it is able to handle a set of data lying on a low-dimensional manifold that is nonlinearly embedded in a higher-dimensional input space. For the preservation of the intrinsic data structure, TRNMap computes the dissimilarities of the data points based on the graph distances. To compute the graph distances, the set of data is replaced by the graph resultant of the TRN algorithm applied on the data set. The edges of the graph are labeled with their Euclidean length and Dijkstra's algorithm [6] is run on the graph, in order to compute the shortest path for each pair of points. The TRNMap algorithm utilizes the group of multidimensional scaling mapping algorithms to give the low-dimensional representation of the data set. If the aim of the mapping is the visualization of the distances of the objects or their representatives, the TRNMap utilizes the metric MDS method. On the other hand, if the user is only interested in the ordering relations of the objects, the TRNMap uses non-metric MDS for the low-dimensional representation. As a result it gives compact low-dimensional topology preserving feature maps to explore the hidden structure of data. In the following the TRNMap algorithm is introduced in detail.

To avoid the influence of the range of the attributes, a normalization procedure is suggested as a preparing step (Step 0). After the normalization, the algorithm creates the topology representing network of the input data set (Step 1). It is achieved by the use of the Topology Representing Network proposed by Martinetz and Shulten [20]. If the resultant graph is unconnected, the algorithm connects the subgraphs by linking the closest elements (Step 2). Then the pairwise graph distances are calculated between every pair of representatives (Step 3). Next, the original topology representing network is mapped into a 2-dimensional graph (Step 4). The mapping method utilizes the similarity of the data points provided by the previously calculated graph distances. This mapping process can be carried out by the use of metric or non-metric multidimensional scaling, as well. For the expressive visualization, component planes are also created by the D-dimensional representatives (Step 5). A component

plane displays the value of one component of each node. If the input data set has D attributes, the TRNMap component plane includes D different maps according to the D components. The structure of these maps is the same as the TRNMap map, but the nodes are represented in greyscale.

Algorithm 4 Topology Representing Network Map algorithm

Step 0 Normalize the input data set $\mathbf{X}$.

Step 1 Create the Topology Representing Network of $\mathbf{X}$ by the use of the TRN algorithm [20]. Yield $M^{(D)} = (\mathbf{W}, \mathbf{C})$ the resulting graph, let $\mathbf{w}_i \in \mathbf{W}$ be the representatives (codebook vectors) of $M^{(D)}$. If exists an edge between the representatives $\mathbf{w}_i$ and $\mathbf{w}_j$ ($\mathbf{w}_i, \mathbf{w}_j \in \mathbf{W}, i \neq j$), $c_{i,j} = 1$, otherwise $c_{i,j} = 0$.

Step 2 If $M^{(D)}$ is not connected, connect the subgraphs in the following way:

While there are unconnected subgraphs ($m_i^{(D)} \subset M^{(D)}, i = 1, 2, \ldots$):

(a) Choose a subgraph $m_i^{(D)}$.

(b) Let the terminal node $\mathbf{t}_1 \in m_i^{(D)}$ and its closest neighbor $\mathbf{t}_2 \notin m_i^{(D)}$ from:
$$\|\mathbf{t}_1 - \mathbf{t}_2\| = min\|\mathbf{w}_j - \mathbf{w}_k\|, \quad \mathbf{t}_1, \mathbf{w}_j \in m_i^{(D)}, \mathbf{t}_2, \mathbf{w}_k \notin m_i^{(D)}$$

(c) Set $c_{\mathbf{t}_1,\mathbf{t}_2}$=1.

End while

Yield $M^{*(D)}$ the modified $M^{(D)}$.

Step 3 Calculate the geodesic distances between all $\mathbf{w}_i, \mathbf{w}_j \in M^{*(D)}$.

Step 4 Map the graph $M^{(D)}$ into a 2-dimensional vector space by metric or non-metric MDS based on the graph distances of $M^{*(D)}$.

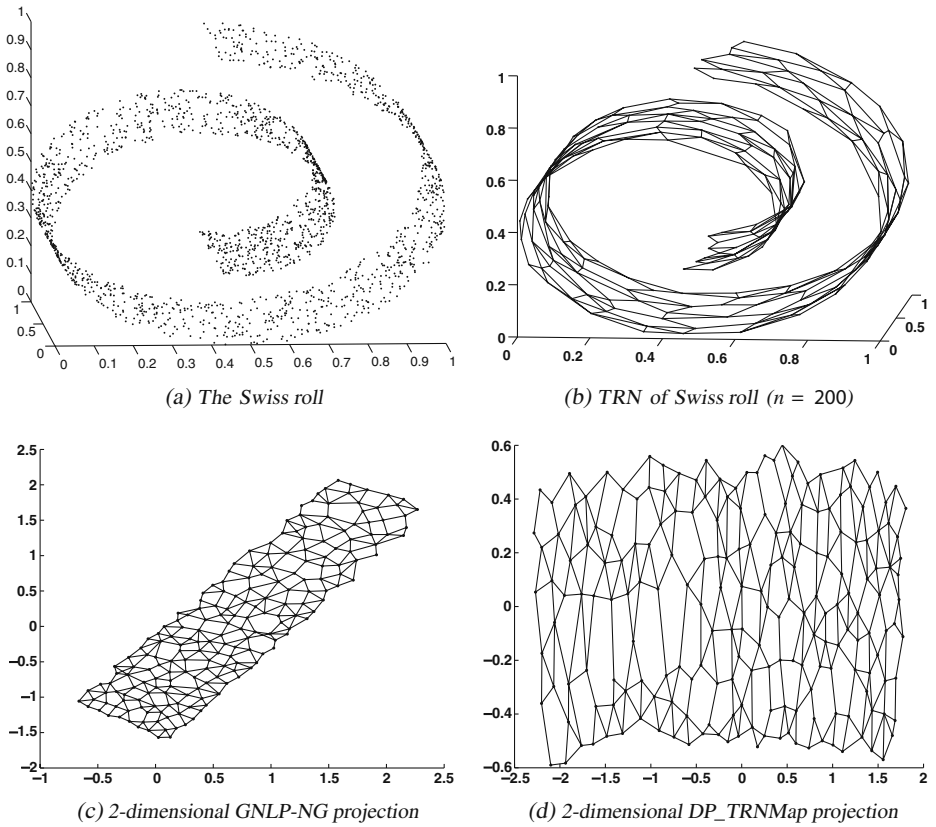
Step 5 Create component planes for the resulting Topology Representing Network Map based on the values of $\mathbf{w}_i \in M^{(D)}$.

The parameters of the TRNMap algorithm are the same as those of the Topology Representing Networks algorithm. The number of the nodes of the output graph (n) is determined by the user. The bigger the n the more detailed the output map will be. We suggest the choice $n = 0.2N$. If the number of the input data elements is high, it can result in numerous nodes. In these cases, it is practical to decrease the number of the representatives and iteratively run the algorithm to capture the structure more precisely. Values of the other parameters of TRN (λ, the step size ε, and the threshold value of edge's ages, T) can be the same as proposed by Martinetz & Schulten [20].

The TRNMap algorithm has different variations based on the mapping used. If the applied MDS is a metric MDS method, the mapping process will preserve the pairwise distances of the objects. On the other hand, if the TRNMap algorithm applies a non-metric MDS, the resultant map tries to preserve the global ordering relations of the data.

Table 1 Systematic overview of the Topology Representing Network based mapping methods

Algorithm	Topology learning	Distance measure	Mapping
GNLP_NG	Modified TRN	Geodesic	Iterative adaptation rule
DP_TRNMap	TRN	Geodesic	Metric MDS
NP_TRNMap	TRN	Geodesic	Non-metric MDS

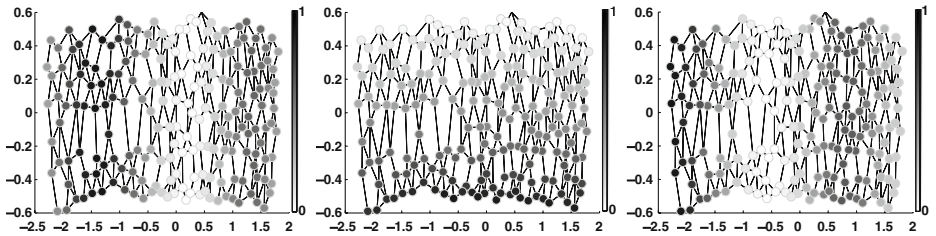


(a) The Swiss roll *(b) TRN of Swiss roll ($n = 200$)*

(c) 2-dimensional GNLP-NG projection *(d) 2-dimensional DP_TRNMap projection*

Fig. 1 The Swiss roll data set, its TRN and its GNLP-NG and DP_TRNMap projections (**a**–**d**)

Table 1 gives a systematic overview of GNLP-NG, metric TRNMap (DP_-TRNMap, DP from distance preserving) and non-metric TRNMap (NP_TRNMap, NP from neighborhood preserving). This table summarizes the applied topology learning methods, distance measures, and mapping techniques.



(a) Dimension 1 *(b) Dimension 2* *(c) Dimension 3*

Fig. 2 Component planes of the metric MSD based Topology Representing Network Map of the Swiss roll data set (**a**–**c**)

Table 2 Values of Sammon stress, metric MDS stress and residual variance of GNLP-NG and TRNMap algorithms on the Swiss roll data set

Algorithm	Sammon stress	Metric MDS stress	Residual variance
GNLP-NG	0.00103	0.00055	0.00170
DP_TRNMap	0.00096	0.00043	0.00156
NP_TRNMap	0.00095	0.00045	0.00155

4 Analysis of the Topology Representing Network Based Mapping Methods

The aim of this section is to analyze the Topology Representing Network based mapping methods that are able to unfold the nonlinearly embedded manifolds. The mapping qualities are analyzed based on the following aspects:

– preservation of distance and neighborhood relations of data, and
– preservation of local and global geometry of data.

The distance preservation of the methods to be analyzed is measured by the metric MDS stress function, Sammon stress function and residual variance. The neighborhood preservation and the local and global mapping qualities are measured by functions of trustworthiness and continuity.

4.1 Mapping Quality

The metric MDS stress function (see Eq. 1) and Sammon stress function (see Eq. 3) have been shown in Section 1. The *residual variance* is defined as:

$$1 - \mathbf{R}^2 \left(\mathbf{D}_X^*, \mathbf{D}_Y \right), \tag{11}$$

where $\mathbf{D}_Y$ denotes the matrix of Euclidean distances in the low-dimensional output space ($\mathbf{D}_Y = [d_{i,j}]$), and $\mathbf{D}_X^* = [d_{i,j}^*]$ is the best estimation of the distances of the data to be projected. $\mathbf{R}$ is the standard linear correlation coefficient, taken over all entries of $\mathbf{D}_X^*$ and $\mathbf{D}_Y$.

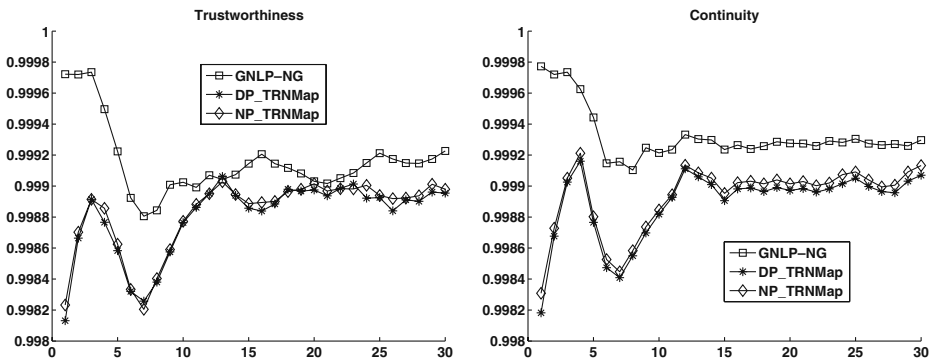


Fig. 3 Trustworthiness and continuity of GNLP-NG and TRNMap methods as a function of the number of neighbors k, for the Swiss roll data set (**a**, **b**)

Table 3 CPU time for different mappings of the Swiss roll data set ($n = 200$)

Algorithm	CPU time (s)
GNLP-NG	140.9313
DP_TRNMap	18.2989
NP_TRNMap	18.2209

A projection is said to be *trustworthy* [14, 27] if the nearest neighbors of a point in the reduced space are also close in the original vector space. Let n be the number of the objects to be mapped, $U_k(i)$ be the set of points that are in the k size neighborhood of the sample i in the visualization display but not in the original data space. The measure of trustworthiness of visualization can be calculated in the following way:

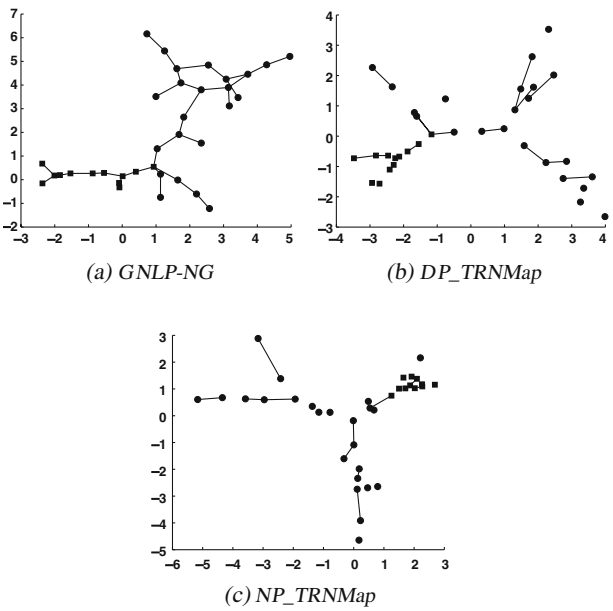
$$M_1(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in U_k(i)} (r(i, j) - k), \tag{12}$$

where $r(i, j)$ denotes the ranking of the objects in the input space.

The projection onto a lower dimensional output space is said to be *continuous* [14, 27] if points near to each other in the original space are also nearby in the output space. Denote $V_i(k)$ the set of those data points that belong to the k-neighbors of data sample i in the original space, but not in the visualization. The measure of continuity of visualization is calculated by the following equation:

$$M_2(k) = 1 - \frac{2}{nk(2n - 3k - 1)} \sum_{i=1}^{n} \sum_{j \in V_k(i)} (\widehat{r}(i, j) - k), \tag{13}$$

where $\widehat{r}(i, j)$ is the rank of the data sample i from j in the output space.



(a) GNLP-NG *(b) DP_TRNMap*

(c) NP_TRNMap

Fig. 4 GNLP-NG and TRNMap visualizations of the Wisconsin breast cancer data set (**a**–**c**)

Table 4 The values of the Sammon stress, MDS stress and residual variance of different mapping algorithms on the Wisconsin breast cancer data set ($n = 35$)

Algorithm	Sammon stress	MDS stress	Residual variance
GNLP-NG	0.02996	0.02764	0.09733
DP_TRNMap	0.01726	0.01075	0.04272
NP_TRNMap	0.01822	0.01077	0.03790

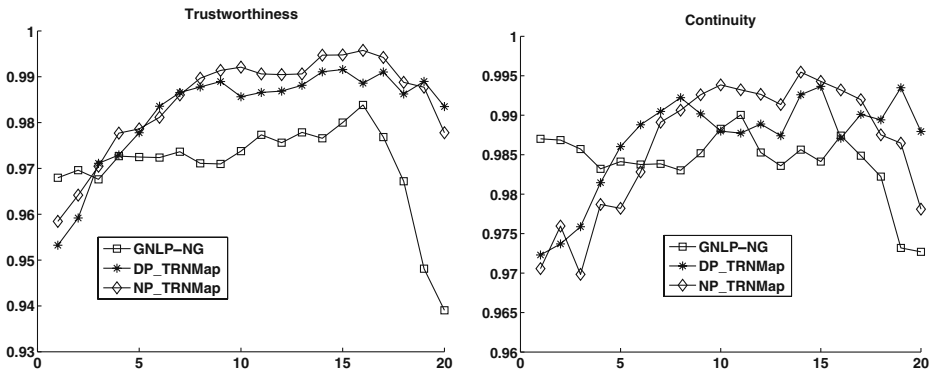
4.2 Analysis of the Methods

In this subsection the mapping qualities of GNLP-NG and TRNMap methods are analyzed. The common parameters of GNLP-NG and TRNMap algorithms were in the simulations set as follows: $t_{max} = 200n$, $\varepsilon_i = 0.3$, $\varepsilon_f = 0.05$, $\lambda_i = 0.2n$, $\lambda_f = 0.01$, $T_i = 0.1n$, $T_f = 0.05n$. The auxiliary parameters of the GNLP-NG algorithm were set as $\alpha_i = 0.3$, $\alpha_f = 0.01$, $k_{max} = 2$, $\sigma_i = 0.7n$ and $\sigma_f = 0.1$.

On visualization results presented in the following the class labels are also presented. The representatives are labeled based on the principle of the majority vote: (1) each data point is assigned to the closest representative; (2) the representatives are labeled with the class label that occurs most often among its assigned data point.

The *Swiss roll data set* (see Fig. 1a) is a typical example of the nonlinearly embedded manifolds. This data set contains a 2-dimensional manifold nonlinearly embedded in the 2-dimensional vector space. All objects belongs to 1 class. In this example the number of data points is 2000 and the number of the representatives was chosen to be $n = 200$.

As both GNLP-NG and TRNMap methods are based on the creation of the Topology Representing Network, Fig. 1b shows the TRN of the Swiss roll data set. In Fig. 1c–d the GNLP-NG projection and the 2-dimensional metric MDS based TRNMap visualization of the Swiss roll data set are shown. As the metric MDS and the non-metric MDS based mappings of the resultant TRN in this case give very similar results in the mapped prototypes, the resultant TRNMap visualizations are not distinguishable by human eye. Thereby Fig. 1d can be seen as the result of the



(a) *(b)*

Fig. 5 Trustworthiness and continuity as a function of the number of neighbors k, for the Wisconsin breast cancer data set ($n = 35$) (**a**, **b**)

Table 5 Values of Sammon stress, MDS stress and residual variance of different mapping algorithms on the Wisconsin breast cancer data set ($n = 70$)

Algorithm	Sammon stress	MDS stress	Residual variance
GNLP-NG	0.06293	0.05859	0.22249
DP_TRNMap	0.01544	0.00908	0.03370
NP_TRNMap	0.02279	0.01253	0.02887

non-metric MDS based TRNMap algorithm as well. It can be seen that GNLP-NG and TRNMap mappings can uncover the structure of data in essence.

The visualization of the Topology Representing Network Map also includes the construction of the component planes. The component planes arising from the metric MDS based TRNMap are shown in Fig. 2. The largest value of the attributes of the representatives corresponds to the black and the smallest value to the white dot surrounded by a grey circle. Figure 2a shows that alongside the manifold the value of the first attribute (first dimension) initially grows to the highest value, then it decreases to the smallest value, after that it grows, and finally it decreases a little. The value of the second attribute is invariable alongside the manifold, but across the manifold it changes uniformly. The third component starts from the highest value, then it falls to the smallest value, following this it increases to a middle value, and finally it decreases a little.

Table 2 shows the error values of distance preservation of different mappings. Although, GNLP-NG and TRNMap methods show similar performances in distance preservation, the TRNMap methods show somewhat better performance.

Figure 3 shows the neighborhood preservation mapping qualities of the methods to be analyzed. The continuity and the trustworthiness of GNLP-NG and TRNMap mappings do not show a substantive difference, the qualitative indicators are greater than 0.998. It means, that all methods to be examined have found the embedded manifold. Trustworthiness and continuity are functions of the number of neighbors k. As small k-nn-s the local reconstruction performance of the model is tested, while

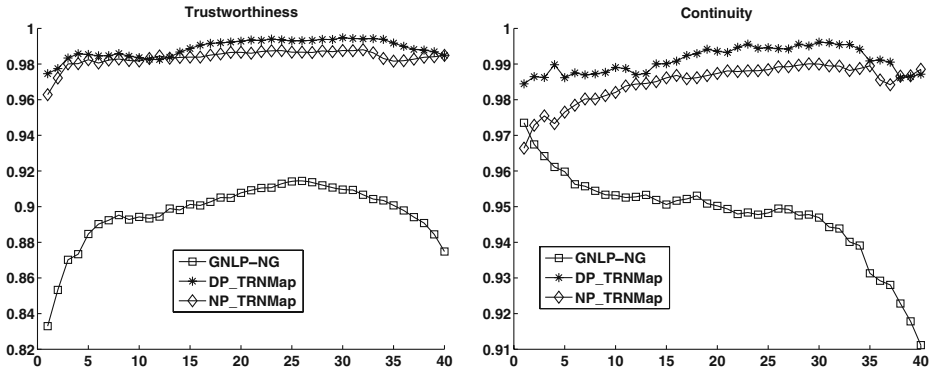


Fig. 6 Trustworthiness and continuity as a function of the number of neighbors k, for the Wisconsin breast cancer data set ($n = 70$) (**a**, **b**)

Table 6 CPU time for different mappings of the Wisconsin breast cancer data set

Algorithm	CPU time (s)	
	$n = 35$	$n = 70$
GNLP-NG	4.3680	14.2117
DP_TRNMap	2.4180	6.0528
NP_TRNMap	2.7924	6.3648

at larger k-nns the global reconstruction is measured. It can be seen, that the GNLP-NG method shows better performance in the local area than the TRNMap mappings.

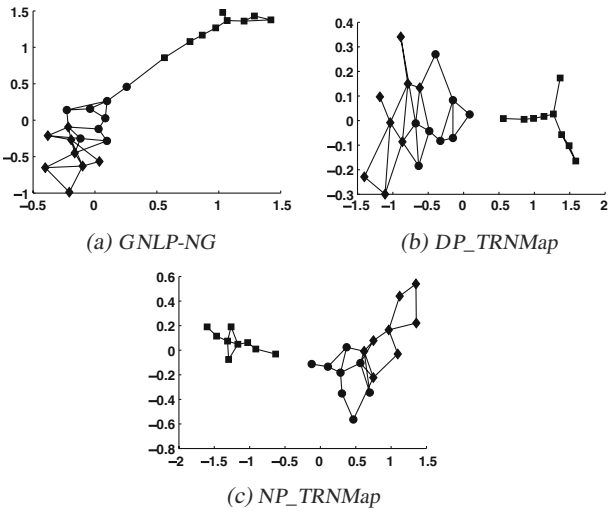
Table 3 shows the CPU time for different mappings of the Swiss roll data set. It can be seen that the TRNMap algorithms produce faster the visualization results than the GNLP-NG method.

In this example, it has been shown that GNLP-NG and TRNMap methods are able to uncover non-linearly embedded manifolds; these methods show good performance both in topology and distance preservation, and the component planes of the TRNMap provide useful facilities to unfold the relations among the features.

The *Wisconsin breast cancer database* is a well-known diagnostic data set for breast cancer compiled by Dr William H. Wolberg, University of Wisconsin Hospitals [10, 18]. This data set contains 9 attributes and class labels for the 683 instances of which 444 are benign and 239 are malignant. The number of the nodes in this case was reduced to $n = 35$ and $n = 70$.

To get a compact representation of the data set to be analyzed, the number of the neurons was initially chosen to be $n = 35$. TRNMap and GNLP-NG visualizations of the Wisconsin breast cancer data set are shown in Fig. 4. The results of the several runs seem to have drawn a fairly wide partition and another compact partition. In these figures the representatives of the benign class are labeled with squares and representatives of the malignant class with circles.

As it can be seen in Fig. 4, the resultant plots suggest that the data can be divided into two or three clusters. Among these clusters one is related to the class benign.



(a) GNLP-NG *(b) DP_TRNMap*

(c) NP_TRNMap

Fig. 7 GNLP-NG and TRNMap visualizations of the Iris data set (**a**–**c**)

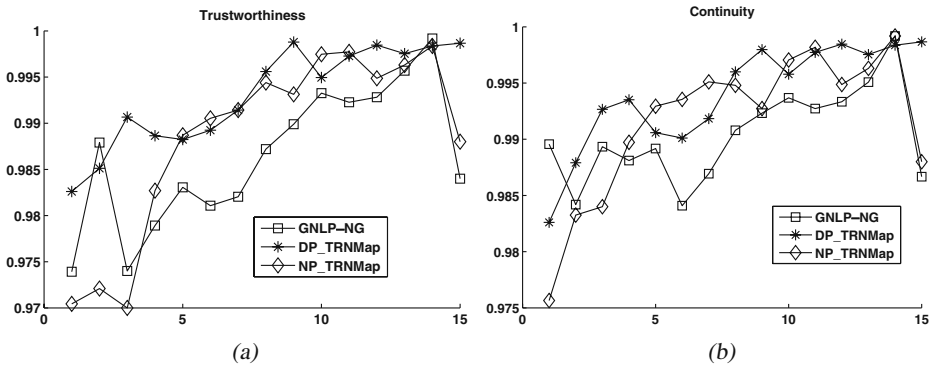


Fig. 8 Trustworthiness and continuity of GNLP-NG and TRNMap methods as a function of the number of neighbors k, for the Iris data set (**a**, **b**)

This information has been confirmed by other data-driven models, like classifiers initialized by decision trees [1].

Table 4 shows the numerical evaluation of the distance preservation capabilities of the mappings. It can be seen that TRNMap mapping methods show better mapping qualities.

The quality of the neighborhood preservation of the mappings is shown in Fig. 5. It can be seen, that the NP_TRNMap and DP_TRNMap methods give better performances at larger k-nn values. On the other hand, in most cases, the GNLP-NG technique gives better performance at the local reconstruction.

To examine the robustness of the methods to be analyzed, another number of the representatives has also been tried. In the second case the number of the representatives was chosen to be $n = 70$. Table 5 and Fig. 6 show the numerical evaluations of the methods in this case (other parameters were not changed). As quality measures of neighborhood preservation of the TRNMap methods move between 0.95 and 1 in both cases ($n = 35$, and $n = 70$), the TRNMap method provides good mapping results in both cases (see Figs. 5 and 6). Compared to this, the GNLP-NG method shows a quite bad mapping result in the case of $n = 70$ (Fig. 6). This fact is confirmed by the error values of the distance preservation, as well (Table 5). Therefore, both the error values and the functions show that the GNLP-NG method has fallen in a local minimum. This could be caused by the fact that GNLP-NG applies a gradient based iterative optimization procedure that can be stuck in local minima. Therefore we can say that TRNMap methods appear to be much less sensitive to the number of the mapped codebooks than the GNLP-GL method.

Table 7 Values of Sammon stress, MDS stress and residual variance of different mapping algorithms on the Iris data set ($n = 70$)

Algorithm	Sammon stress	MDS stress	Residual variance
GNLP-NG	0.00711	0.00324	0.00914
DP_TRNMap	0.00230	0.00062	0.00183
NP_TRNMap	0.00426	0.00163	0.00491

Table 8 CPU time for different mappings of the Iris data set ($n = 200$)

Algorithm	CPU time (s)
GNLP-NG	2.6988
DP_TRNMap	0.9360
NP_TRNMap	1.7472

The CPU time of different mappings have been also analyzed (see Table 6). The DP_TRNMap and NP_TRNMap require significantly shorter calculation than the GNLP-NG method.

To show the correctness of the previous observations, one more application example is presented. The *Iris data set* [9] (http://www.ics.uci.edu) contains measurements on three classes of iris flowers. The data set was made by measurements of sepal length and width and petal length and width for a collection of 150 irises. The data set contains 50 samples from each class of iris flowers (iris setosa, iris versicolor and iris virginica). The goal is to distinguish the three different types. Iris setosa is easily distinguishable from the other two types, but iris versicolor and iris virginica are very similar to each other. This data set has been analyzed many times to illustrate various clustering methods.

The number of the representatives in this case was chosen to be $n = 25$. Results of the different mapping methods can be seen in Fig. 7. Visualization results meet *a priori* expectations. In all the three cases we can see a well separated class (iris setosa), and another group that contains the representatives of the classes iris virginica and iris versicolor. Figure 7 shows that the NP_TRNMap and the DP_TRNMap methods separate the representatives of iris setosa from the other two classes better than the GNLP-NG method. Figure 8 shows the neighborhood preservations of the mapping algorithms. It confirms the previous statement, that the GNLP-NG mapping shows better performance in the local area than TRNMap methods.

Table 7 summarizes the error values of the studied methods. The error values also confirm that the TRNMap methods have better mapping qualities than the GNLP-NG based visualization. Table 8 presents CPU time used for different mappings of the Iris set. Similarly to the previous examples TRNMap methods need shorter time to produce results.

The mapping methods have also been tested on other benchmark examples, and the results confirmed the previous statements.

5 Conclusion

In this paper we have defined a new class of mapping methods, that are based on the topology representing networks. To detect the main properties of the topology representing network based mapping methods, an analysis was performed on them. The primary aim of this analysis was to examine the preservation of distances and neighborhood relations of data. Preservation of neighborhood relations was analyzed both in local and global environments. It has been shown that: (1) Multidimensional scaling is an effective method to form a low-dimensional map of the TRN based on the calculated graph distances. (2) Metric mapping based algorithms (e.g. TRNMap

based on metric MDS) directly minimize the stress functions, so their performance appears to be the best in distance perseveration. (3) In point of neighborhood preservation GNLP-NG mapping shows better performance in the local area than TRNMap methods. (4) MDS-based techniques can be considered as global reconstruction methods, hence in most cases they give better performances at larger k-nn values. (5) Comparing TRNMap and GNLP-NG methods, it can be seen that TRNMap methods are more robust to the initialization parameters. MDS-based techniques are much less sensitive to the number of the mapped codebook vectors than the GNLP-NG technique, which tends to give worse performances when the number of codebook vectors is increased. This could be caused by the fact that GNLP applies a gradient based iterative optimization procedure that can be stuck in local minima. (6) The GNLP-NG needs more computational time, than the MDS based TRNMap methods. (7) The quality of the TRN has a significant role in the exploration of the hidden structure of the data. When classical TRN is used, the user (the knowledge worker) only receives numerical feedback about the result of the algorithm. The main benefit of the proposed visualization based approach is that the resulting figures can serve as a visual feedback for the user. Hence, interactive data mining can be realized as the user can immediately evaluate the resulting figures and see the effects of the parameters. Comparing these plots to each other and to the a priori knowledge of the knowledge worker a proper model can be fine tuned.

As a future topic, we plan to compare the ViSOM and the proposed TRNMap methods.

Acknowledgements The authors acknowledge the financial support of the Cooperative Research Centre (VIKKK, project 2004-I), the Hungarian Research Found (OTKA 49534), the Bolyai Janos fellowship of the Hungarian Academy of Science, and the Öveges fellowship.

References

1. Abonyi, J., Roubos, J.A., Szeifert, F.: Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. Int. J. Approx. Reason. **32**(1) 1–21 (2003)
2. Bernstein, M., de Silva, V., Langford, J.C., Tenenbaum, J.B.: Graph approximations to geodesics on embedded manifolds. Techn. Rep., Stanford Univ. (2000)
3. Borg, I., Groenen, P.: Modern Multidimensional Scaling: Theory and Applications. Springer Series in Statistics. Springer, New York (1997)
4. Comon, P.: Independent component analysis: a new concept? Signal Process. **36**(3), 287–317 (1994)
5. Demartines, P., Herault, J.: Curvilinear component analysis: a selforganizing neural network for nonlinear mapping of data sets. IEEE Trans. Neural Netw. **8**(1), 148–154 (1997)
6. Dijkstra, E.W.: A note on two problems in connection with graphs. Numer. Math. **1**, 269–271 (1959)
7. Estévez, P.A., Figueroa, C.J.: Online data visualization using the neural gas network. Neural Netw. **19**(6), 923–934 (2006)
8. Estévez, P.A., Chong, A.M., Held, C.M., Perez, C.A.: Nonlinear projection using geodesic distances and the neural gas network. ICANN2006 LNCS **4131**, 464–473 (2006)
9. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**, 179–188 (1936)
10. Fryback, D.G., Stout, N.K., Rosenberg, M.A., Trentham-Dietz, A., Kuruchittham, V., Remington, P.L.: Chapter 7: the Wisconsin breast cancer epidemiology simulation model. JNCI Monogr. **36**, 37–47 (2006)
11. Hebb, D.O.: The Organization of Behavior. Wiley, New York (1949)

12. Hoteling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24** (1933) 417–441
13. Jolliffe, I.T.: Principal Component Analysis. Springer, New York (1996)
14. Kaski, S., Nikkilä, J., Oja, M., Venna, J., Törönen, P., Castrén, E.: Trustworthiness and metrics in visualizing similarity of gene expression. BMC Bioinformatics **4**(48), (2003). http://www.biomedcentral.com/1471-2105/4/48
15. Kohonen, T.: Self-organising Maps (2nd edn). Springer, Berlin (1995)
16. Kruskal, J.B.: Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. Psychometrika **29**, 1–29 (1964)
17. Lee, J.A., Lendasse, A., Donckers, N., Verleysen, M.: A robust nonlinear projection method. In: Verleysen, M. (ed.) Proceedings of ESANNŠ2000, 8th European Symposium on Artificial Neural Networks, pp. 13–20. Bruges (2000)
18. Mangasarian, O.L., Wolberg, W.H.: Cancer diagnosis via linear programming. SIAM News **23**(5), 1–18 (1990)
19. Martinetz, T.M., Schulten, K.J.: A neural-gas network learns topologies. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.) Artificial Neural Networks, pp. 397–402. Elsevier, Amsterdam (1991)
20. Martinetz, T.M., Shulten, K.J.: Topology representing networks. Neural Netw. **7**(3), 507–522
21. Muhammed, H.H.: Unsupervised fuzzy clustering using weighted incremental neural networks. Int. J. Neural Syst. **14**(6), 355–371 (2004)
22. Sammon, J.W.: A non-linear mapping for data structure analysis. IEEE Trans. Comput. **C18**(5), 401–409 (1969)
23. Si, J., Lin, S., Vuong, M.-A.: Dynamic topology representing networks. Neural Netw. **13**, 617–627 (2000)
24. Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**, 2319–2323 (2000)
25. Vathy-Fogarassy, A., Werner-Stark, A., Gal, B., Abonyi, J.: Visualization of topology representing networks. In: Lecture Notes in Computer Science: Intelligent Data Engineering and Automated Learning - IDEAL 2007, vol. 4881/2007, pp. 557–566. Springer, Berlin/Heidelberg (2007)
26. Vathy-Fogarassy, A., Kiss, A., Abonyi, J.: Topology representing network map–a new tool for visualization of high-dimensional data. LNCS Trans. Comput. Sci. I **4750/2008**, 61–84 (2008)
27. Venna, J., Kaski, S.: Local multidimensional scaling. Neural Netw. **19**, 889–899 (2006)
28. Yin, H.: ViSOM–a novel method for multivariate data projection and structure visualisation. IEEE Trans. Neural Netw. **13**, 237–243 (2002)
29. Young, G., Householder, A.S.: Discussion of a set of points in terms of their mutual distances. Psychometrika **3**(1), 19–22 (1938)