Distance-Preserving SOM: A New Data Visualization Algorithm

Chao Shao
School of Computer & Information Engineering, Henan University of Economics and Law, Zhengzhou 450002, China
Email: shaochao051227@gmail.com

Yongqiang Yang
School of Computer & Information Engineering, Henan University of Economics and Law, Zhengzhou 450002, China
Email: yyqndsc@sina.com

***Abstract*—As the combination of topology-preserving dimensionality reduction and vector quantization, Self-Organizing Map (SOM) is suitable for visualizing the structure of high-dimensional mass data, which can be used to select more suitable algorithms for subsequent data analysis/processing. However, due to the fixed regular lattice of neurons, SOM has to require some color-coding scheme such as U-matrix to imprint the inter-neuron distance information on the lattice for the aim of visualization. Even so, the structure of the data may often appear in a distorted and unnatural form. In order for the map to visualize the structure of the data faithfully and naturally, the similarity/dissimilarity information should be preserved on the map directly. To do this, a novel variant of SOM, i.e. Distance-Preserving SOM (DPSOM), was presented in this paper. DPSOM can adjust the positions of neurons on the map according to the corresponding distances in the data space, and thus preserve the distance information on the map directly, as Multidimensional Scaling (MDS) does. What's the most important, DPSOM can automatically avoid the excessive contraction of neurons to one point without any additional parameter, which makes it advantageous over those existing position-adjustable SOMs. Finally, DPSOM can be verified by experimental results well.**

***Index Terms*—data visualization, Self-Organizing Map (SOM), Himberg's contraction model, Multi-Dimensional Scaling (MDS), the gradient descent**

I. INTRODUCTION

Nowadays, the explosive growth in the amount of data and their dimensionality makes data visualization more and more important in data mining process. For high-dimensional mass data, the useful structure information, which can be used to select more suitable algorithms for subsequent data analysis/processing, cannot be seen by eyes directly, but can be obtained by data visualization approaches easily.

During the last hundreds of years, lots of approaches to visualize high-dimensional mass data have been emerged, most of which fall into the following five categories:

1) Several sub-windows are used to visualize the data in different subsets of dimensions respectively, such as scatterplot matrices[1] and pixel-oriented techniques[2];

2) The dimension axes are rearranged in the low-dimensional space, such as parallel coordinates[3] and star coordinates[4];

3) The dimensions of the data are embedded to partition the low-dimensional space hierarchically, such as dimensional stacking[5] and treemap[6];

4) Certain objects with several visual features are used to represent the high dimensional data, in which each visual feature stands for one dimension of the data, such as Chernoff-faces[7] and stick figures[8];

5) The dimensionality of the data is reduced to two or three by dimensionality reduction techniques, such as PCA (Principal Component Analysis)[9], MDS (Multidimensional Scaling)[10], SOM (Self-Organizing Map)[11], ISOMAP (Isometric Mapping)[12], LLE (Locally Linear Embedding)[13] and Laplacian Eigenmap[14], *etc*.

Unlike the other approaches, dimensionality reduction techniques try to preserve the high-dimensional relationship between the data in the low-dimensional space directly, which can represent visually the structure of the data well. In addition, dimensionality reduction techniques can be used to avoid "the curse of dimensionality" and improve the efficiency and performance of the subsequent data analysis/processing algorithms.

As a non-linear dimensionality reduction technique, SOM is a singular-layer neural network model based on competitive learning. As the combination of topology-preserving dimensionality reduction and vector quantization[15], SOM can map the high-dimensional mass data onto a low-dimensional regular lattice, i.e. a fixed grid of neurons, while preserving the topological relationship between the data as faithfully as possible, which makes it a popular clustering, visualization and abstraction tool.

However, due to the fixed regular lattice of neurons, the inter-neuron distances has to be expressed indirectly on the lattice by some color-coding scheme such as U-matrix[15] for visualizing the structure of the data. Even

Manuscript received February 12, 2011; revised June 5, 2011; accepted June 15, 2011

doi:10.4304/jsw.7.1.196-203

so, the structure of the data may often appear in a distorted and unnatural form[16].

In order for the map to visualize the structure of the data faithfully and naturally, the similarity/dissimilarity information should be preserved on the map directly[16]. To do this, the positions of neurons on the map should be adjusted according to the corresponding similarities/ dissimilarities during or after the learning process. Taking the concision of the model and the amount of computing time into consideration, we prefer adjusting the positions of neurons synchronously during the learning process. We name such variants of SOM as position-adjustable SOMs[17], such as the Grouping Neuron SOM (GNSOM) algorithm[18], the Adaptive Coordinate SOM (ACSOM) algorithm[19], the Double SOM (DSOM) algorithm[20], and the Position-Adjustable SOM (PASOM) algorithm[17]. Being add-ins to the standard Kohonen's SOM, the robustness of these methods can be assured[19]. In addition, the adjustment rule is relatively simple. However, all these methods fall into the category of Himberg's contraction model, which is inevitably confronted with the problem of the excessive contraction to one point[17]. To avoid this problem, these methods have to use some quite complex initialization procedure or additional parameters which are difficult to control in reality. In this paper, we present a novel variant of SOM, i.e. Distance-Preserving SOM (DPSOM), which can adaptively adjust the positions of neurons on the map according to the corresponding distances in the data space, as MDS does. Unlike those existing position-adjustable SOMs, DPSOM can automatically avoid the excessive contraction of neurons to one point without any additional parameter.

This paper is organized as follows: In Section II, we recall those existing position-adjustable SOMs and MDS briefly. In Section III, we describe and analyze the DPSOM algorithm in detail. Finally, experimental results and conclusions are given in Section IV and V respectively.

II. Position-Adjustable SOMs and MDS

A. The Standard Kohonen's SOM

The standard Kohonen's SOM consists of a singular-layer of neurons located on a low-dimensional regular lattice, usually 1-D or 2-D for the aim of visualization. In addition to the fixed position vector on the lattice, each neuron k is also represented by an n-dimensional weight vector $w_k = \{w_{k1}, \cdots, w_{kn}\}$ in the data space, where n is the dimensionality of the data. SOM adopts neighborhood learning to update the weight vectors of neurons fixed on the lattice, which makes all the neurons representatives of the data and arranged by topological order in the data space finally.

In each learning step, a data sample x is randomly selected at first, and then the best-matching neuron (BMU) v is found according to the following rule:

$$v = \arg\min_k \|w_k - x\|. \quad (1)$$

After that, SOM adopts neighborhood learning to update the weight vectors of the neurons belonging to a given neighborhood of the BMU v on the lattice according to the following learning rule:

$$w_k(t+1) = w_k(t) + \alpha(t) h_{vk}(t)(x - w_k(t)). \quad (2)$$

Where $\alpha(t)$ is the learning rate at the t-th iteration, and $h_{vk}(t)$ is the neighborhood kernel at the t-th iteration, which is defined on the lattice and usually takes the following form:

$$h_{vk}(t) = e^{\frac{\|v-k\|^2}{2\sigma(t)^2}}. \quad (3)$$

Where $\|v-k\|$ represents the distance between the BMU v and neuron k on the lattice, and $\sigma(t)$ represents the neighborhood radius at the t-th iteration. To assure the convergence of the learning process, both the learning rate $\alpha(t)$ and the neighborhood radius $\sigma(t)$ decrease monotonically with time.

Because not only the BMU but also its neighbors on the lattice are updated in the same direction, the weight vectors of neighboring neurons resemble each other[2]. Consequently, the BMUs of similar data samples are close to each other on the lattice, which is so-called SOM's topological ordering or topological preserving. For the sake of visual comparison and easy implementation, the neurons are fixed onto a regular lattice, usually rectangular or hexagonal. However, the inter-neuron distances are indirectly visible, and thus the structure of the data may often be distorted and unnatural.

B. Position-Adjustable SOMs

In order for the map to visualize the structure of the data faithfully and naturally, several variants of SOM were presented, such as GNSOM, ACSOM, DSOM and PASOM. The same of them is that they can adjust the positions of neurons on the map (concretely speaking, contract the neurons) according to the corresponding similarities, and thus the similarity information can be preserved on the map directly, which makes these variants of SOM obtain better visualization than the standard Kohonen's SOM, so we name them position-adjustable SOMs.

Let m be the dimensionality of the low-dimensional regular lattice or the map, and $p_k = \{p_{k1}, \cdots, p_{km}\}$ be the position vector of neuron k on the map, then the adjustment rules of these position-adjustable SOMs can be described as follows respectively:

$$\text{GNSOM:} \quad p_k(t+1) = p_k(t) + a(t) \cdot b_{vk}(t) \cdot (p_v(t) - p_k(t)) \quad (4)$$

$$\text{ACSOM:} \quad p_k(t+1) = p_k(t) + \Delta Dist_k(t+1) \cdot (p_v(t) - p_k(t)) \quad (5)$$

$$\text{DSOM:}\quad p_k(t+1) = p_k(t) + \eta_{vk}(t)\cdot h'_{vk}(t)\cdot(p_v(t)-p_k(t)) \tag{6}$$

$$\text{PASOM:}\quad p_k(t+1) = p_k(t) + cf\cdot\alpha(t)\cdot h^p_{vk}(t)\cdot (p_v(t)-p_k(t)),\ t > threshold. \tag{7}$$

Obviously, $b_{vk}(t)$ in (4) and $h'_{vk}(t)$ in (6) can be thought as the similarity between the BMU *v* and neuron *k* in the data space, because both of them decrease as the distance between these two neurons in the data space increases; $\eta_{vk}(t)$ in (6) can be thought as the similarity between the BMU *v* and neuron *k* on the map, because it decreases as the distance between these two neurons on the map increases.

Theorem 1. $\Delta Dist_k(t+1)$ in (5) can be thought as the similarity between the BMU *v* and neuron *k* on the map.

Proof. $\Delta Dist_k(t+1)$ means the relative change of the distances between neuron *k* and the randomly selected data sample *x* before and after the *t*-th iteration, that is,

$$\Delta Dist_k(t+1) = \frac{Dist_k(t) - Dist_k(t+1)}{Dist_k(t)}. \tag{8}$$

$$\begin{aligned}
\because Dist_k(t+1) &= \|x - w_k(t+1)\| \\
&= \|x - (w_k(t) + \alpha(t)h_{vk}(t)(x - w_k(t)))\| \\
&= \|(1-\alpha(t)h_{vk}(t))(x - w_k(t))\| \\
&= (1-\alpha(t)h_{vk}(t))\|x - w_k(t)\| \ (\because 0 < \alpha(t)h_{vk}(t) \le 1) \\
&= (1-\alpha(t)h_{vk}(t))Dist_k(t) \\
\therefore \Delta Dist_k(t+1) &= \frac{Dist_k(t) - Dist_k(t+1)}{Dist_k(t)} \\
&= \alpha(t)h_{vk}(t)
\end{aligned}$$

So, $\Delta Dist_k(t+1)$ can be thought as the similarity between the BMU *v* and neuron *k* on the map like the neighborhood kernel $h_{vk}(t)$.

Due to SOM's topological ordering or topological preserving, these similarity measures are similar on the final map. So we can say that all the former three methods contract the neurons on the map according to the corresponding similarities because all these similarity measures are always positive like $h_{vk}(t)$, which is so-called Himberg's contraction model[17][21], and its online learning version can be expressed as in (7), where $h^p_{vk}(t)$ is the generalized similarity between the BMU *v* and neuron *k*, which can be any one of the above similarity measures or their combinations, and where the contraction factor *cf* and the beginning iteration of adjustment *threshold* are two additional parameters to avoid the excessive contraction of neurons to one point.

With no additional parameter, Himberg's contraction model is inevitably confronted with the problem of the excessive contraction to one point, because the similarity measure used in Himberg's contraction model are always positive. To avoid this problem, these position-adjustable SOMs have to use some quite complex initialization procedure or additional parameters, such as the contraction rate $a(t)$ in (4), *cf* and *threshold* in (7), and those implied in the above similarity measures, which are difficult to control in reality.

C. MDS

As a traditional method related to dimensionality reduction and data visualization, MDS tries to project data samples into a low-dimensional (usually 2-D) space by preserving the inter-sample distances (usually Euclidean distances) as closely as possible. So the objective function or the so-called stress function of MDS usually takes the following form:

$$S = \frac{1}{2}\sum_i\sum_{i<j}(\delta_{ij} - d_{ij})^2. \tag{9}$$

Where δ_{ij} is the distance (usually Euclidean distance) between the *i*-th and *j*–th data samples in the data space, and d_{ij} is the corresponding Euclidean distance (for the aim of visualization) in the low-dimensional space. If we still use $w_k = \{w_{k1}, \cdots, w_{kn}\}$ and $p_k = \{p_{k1}, \cdots, p_{km}\}$ to represent the *k*-th data sample in the data space and its mapping in the low-dimensional space respectively, δ_{ij} and d_{ij} can be defined as follows:

$$\begin{aligned}
\delta_{ij} &= \sqrt{\sum_{i=1}^{n}(w_{ik} - w_{jk})^2} \\
d_{ij} &= \sqrt{\sum_{i=1}^{m}(p_{ik} - p_{jk})^2}.
\end{aligned} \tag{10}$$

By minimizing the stress function in (9), MDS can realize a point-to-point dimensionality reduction mapping, which preserves the inter-sample distances in the low-dimensional space directly, and thus can visualize the structure of the data more faithfully and naturally than those SOMs.

There are many optimization techniques to minimize the stress function in (9), such as the gradient descent method described in (11).

$$\begin{aligned}
p_{ik} &= p_{ik} - \alpha\cdot\frac{\partial S}{\partial p_{ik}} \\
&= p_{ik} - \alpha\cdot\sum_{j\ne i}\frac{\partial S}{\partial d_{ij}}\cdot\frac{\partial d_{ij}}{\partial p_{ik}} \\
&= p_{ik} - \alpha\cdot\sum_{j\ne i}(d_{ij} - \delta_{ij})\cdot\frac{p_{ik} - p_{jk}}{d_{ij}},\ k = 1,2,\cdots,m
\end{aligned} \tag{11}$$

However, these optimization techniques require all the data to obtain the gradient, i.e. $\sum_{j\ne i}(d_{ij} - \delta_{ij})\cdot\frac{p_{ik} - p_{jk}}{d_{ij}}$, used in each learning step, and thus cannot be suitable for online learning tasks.

Similar to SOM, in each learning step, the mappings of the other data samples can be adjusted only based on that of one selected data sample (similar to the BMU *v* in

SOM, e.g. the *j*-th data sample), where the stress function to be minimized can be simplified to its part associated with the selected data sample (e.g. $S_j = \sum_{i \neq j} S_{ij} = \frac{1}{2}\sum_{i \neq j}(\delta_{ij} - d_{ij})^2$), and then the corresponding gradient can be easily obtained only with one pair of data samples.

According to this strategy, the gradient descent method to minimize the stress function of MDS can be described as follows:

$$\begin{aligned} p_{ik} &= p_{ik} - \alpha \cdot \frac{\partial S_j}{\partial p_{ik}} \\ &= p_{ik} - \alpha \cdot \frac{\partial E_j}{\partial d_{ij}} \cdot \frac{\partial d_{ij}}{\partial p_{ik}} \\ &= p_{ik} - \alpha \cdot \frac{\partial E_{ij}}{\partial d_{ij}} \cdot \frac{\partial d_{ij}}{\partial p_{ik}} \\ &= p_{ik} - \alpha \cdot (d_{ij} - \delta_{ij}) \cdot \frac{(p_{ik} - p_{jk})}{d_{ij}} \\ &= p_{ik} - \alpha \cdot (1 - \frac{\delta_{ij}}{d_{ij}}) \cdot (p_{ik} - p_{jk}), \forall i \neq j,\ k = 1, \cdots, m. \end{aligned} \tag{12}$$

III. DPSOM AND ITS ANALYSIS

A. DPSOM

Unlike the above four position-adjustable SOMs, DPSOM adjusts the positions of neurons on the map not based on Himberg's contraction model, but based on the consistency between the corresponding distances of neurons in the data space and on the map, the goal of which is to express the distance information of neurons (and then the data) in the data space by their positions on the map directly, and then visualize the structure of the data faithfully and naturally.

Formally, in each learning step, the positions of neurons on the map are adjusted according to the following adjustment rule:

$$\begin{aligned} p_k(t+1) = p_k(t) + \\ \alpha'(t) \cdot (1 - \frac{\delta_{vk}}{d_{vk}}) \cdot (p_v(t) - p_k(t)), \forall k \neq v. \end{aligned} \tag{13}$$

Where $\alpha'(t)$ is the adjustment rate at the *t*-th iteration, which can be equal to the learning rate $\alpha(t)$ in (2) simply, δ_{vk} and d_{vk} are the Euclidean distances between the BMU *v* and neuron *k* in the data space and on the map respectively, which are defined as follows:

$$\begin{aligned} \delta_{vk} &= \sqrt{\sum_{i=1}^{n} (w_{vi}(t) - w_{ki}(t))^2} \\ d_{vk} &= \sqrt{\sum_{i=1}^{m} (p_{vi}(t) - p_{ki}(t))^2}. \end{aligned} \tag{14}$$

The adjustment rule in (13) adjusts the position of neuron *k* on the map, i.e. $p_k(t)$, based on that of the BMU *v*, i.e. $p_v(t)$, to minimize the difference between δ_{vk} and d_{vk}, the goal of which is to express or preserve δ_{vk} by d_{vk} as closely as possible. $(1 - \frac{\delta_{vk}}{d_{vk}})$ in (13) is similar to the above similarity measures, but the difference is that $(1 - \frac{\delta_{vk}}{d_{vk}})$ is not always positive, sometimes negative, which means that the neurons are not always contracted to each other, so DPSOM is not a contraction model. In addition, the adjustment of the positions of neurons on the map is based on the corresponding distances in the data space (i.e. δ_{vk}), so any two neurons cannot be excessively contracted to one point unless the distance between them in the data space is zero.

Consequently, DPSOM can be described briefly as follows:

1) Specify the lattice structure of neurons, usually rectangular or hexagonal;
2) Initialize the position vectors of all the neurons identical to their positions on the lattice;
3) Initialize the weight vectors of all the neurons randomly;
4) *t*=0;
5) Specify $\alpha(0)$ and $\sigma(0)$;
6) While the stop condition is not met
   a) Initialize *D* as a set of all the data samples;
   b) While *D* is not an empty set
      i) Select a data sample *x* from *D* randomly;
      ii) *D*=*D*-{*x*};
      iii) Find the BMU *v* according to (1);
      iv) Update the weight vectors of the BMU *v* and its neighbors according to (2);
      v) Adjust the position vectors of all the neurons but the BMU *v* according to (13);
   c) End
   d) *t*=*t*+1;
   e) Decrease $\alpha(t)$ and $\sigma(t)$;
7) End

As described above, DPSOM is also a type of position-adjustable SOMs, so it has the advantages of other position-adjustable SOMs too, such as the robustness and simplicity[17].

B. Theoretical Analysis

DPSOM can preserve the distance information of neruons (and then the data) on the map directly as MDS does, because the adjustment rule of DPSOM in (13) can be thought as the gradient descent method to minimize the stress function of MDS in (9).

Theorem 2. the adjustment rule of DPSOM can be thought as the gradient descent method to minimize the stress function of MDS.

Proof. According to the adjustment rule of DPSOM in (13), the positions of the other neurons, i.e. $p_k(t)$, are adjusted only based on that of the BMU *v*, i.e. $p_v(t)$,

where the related part of the stress function of MDS is only $S_v = \sum_{k \neq v} S_{vk} = \frac{1}{2}\sum_{k \neq v}(\delta_{vk} - d_{vk})^2$, the gradient of which is described as follows:

$$\begin{aligned}\frac{\partial S_v}{\partial p_k(t)} &= \frac{\partial S_v}{\partial d_{vk}} \cdot \frac{\partial d_{vk}}{\partial p_k(t)} = \frac{\partial S_{vk}}{\partial d_{vk}} \cdot \frac{\partial d_{vk}}{\partial p_k(t)} \\ &= (d_{vk} - \delta_{vk}) \cdot \frac{(p_k(t) - p_v(t))}{d_{vk}} \\ &= (1 - \frac{\delta_{vk}}{d_{vk}}) \cdot (p_k(t) - p_v(t)).\end{aligned} \tag{15}$$

So, the adjustment rule of DPSOM in (13) can be described as follows:

$$p_k(t+1) = p_k(t) - \alpha'(t) \cdot \frac{\partial S_v}{\partial p_k(t)}, \forall k \neq v. \tag{16}$$

According to (16), we can say that the adjustment rule of DPSOM in (13) can be thought as the gradient descent method to minimize the stress function of MDS. Consequently, DPSOM can be thought as the combination of SOM and MDS to a certain extent.

IV. Experimental Results

In this section, we will apply the standard Kohonen's SOM, PASOM (the additional parameters, *cf* and *threshold*, of PASOM are easier to control than those of the other existing position-adjustable SOMs such as GNSOM, ACSOM and DSOM[17]) and DPSOM on the following three datasets respectively:

1) The butterfly dataset: a two-dimensional dataset with 53 data points (seen in Fig. 1).

2) The IRIS dataset: a well-known four-dimensional dataset with 150 data points, which is divided into three groups equally and two of them are overlapping.

3) The Gaussian5d dataset: a five-dimensional dataset with 180 data points, which is divided into six groups equally and each of them follows the five-dimensional normal distribution with the covariance matrix of I (I is a 5×5 identity matrix). These normal distributions are independent of one another and the means are specified as (0,0,0,0,0), (10,0,0,0,0), (0,10,0,0,0), (0,0,10,0,0), (0,0,0,10,0) and (0,0,0,0,10) respectively.

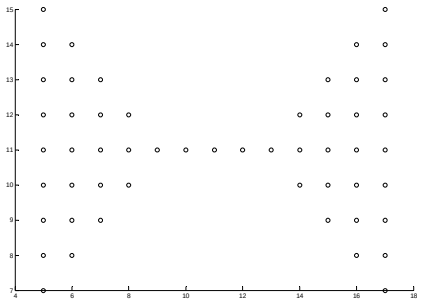

Figure 1. The butterfly dataset.

In the experiments, we specify that the lattice structure is rectangular and the neighborhood kernel $h_{vk}(t)$ is a Gaussian function whose form is given in (3), and the parameters used in SOM are defined as follows: the number of neurons $N = rows \times cols$, where *rows* and *cols* represent the number of rows and columns of neurons respectively and are specified in the brackets of the captions of the corresponding result figures respectively; $\alpha(t) = \max(0.001, 0.9 \cdot e^{-\frac{t}{100}})$, $\sigma(t) = \max(0.0333, \max(rows, cols) \cdot e^{-\frac{t}{100}})$, where $\max(x, y)$ is the larger one between x and y; the learning process stops when the number of iterations reaches 2000 or the square sum of the differences of all the weight vectors between two successive steps is smaller than a threshold $\varepsilon = 0.0001$ and the BMU of each data point is stable. In addition, there are another two parameters in PASOM, i.e. *cf* and *threshold* to avoid the excessive contraction of neurons, which are also specified in the brackets of the captions of the corresponding result figures.

The results of these three algorithms on the butterfly dataset are given in Fig. 2-5, the results of these three algorithms on the IRIS dataset are given in Fig. 6-9, and the results of these three algorithms on the Gaussian5d dataset are given in Fig. 10-13. In these Figures, the neurons onto which there is no data sample mapped is invisible on the map.

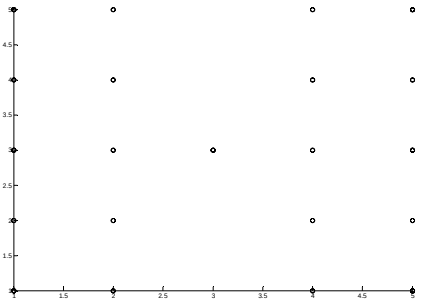

Figure 2. Result of the standard Kohonen's SOM on the butterfly data set (5x5).

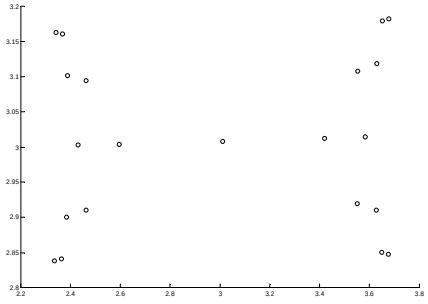

Figure 3. Result of PASOM on the butterfly data set (5x5, *cf*=0.25, *threshold*=150).

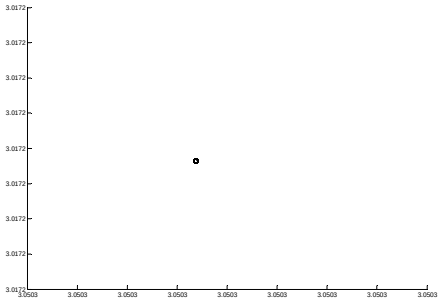

Figure 4. Result of PASOM on the butterfly data set (5x5, *cf*=0.25, *threshold*=50).

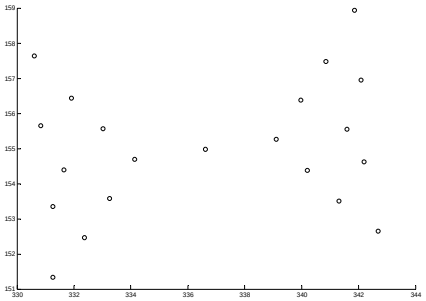

Figure 5. Result of DPSOM on the butterfly data set (5x5).

From Fig. 2, we can see that the standard Kohonen's SOM can keep the shape of the butterfly data set (seen in Fig. 1) to a certain extent, but the shape is visualized distortedly because the neurons are fixed onto the regular lattice. Compared with the standard Kohonen's SOM, PASOM with the proper additional parameters (e.g. *cf*=0.25 and *threshold*=150 in Fig. 3) and DPSOM (seen in Fig. 5) can keep the shape of the butterfly dataset more faithfully and naturally, because the positions of all the neurons on the map are adjusted according to the corresponding similarities. However, the neurons can be contracted excessively in PASOM with the improper additional parameters (e.g. *cf*=0.25 and *threshold*=50 in Fig. 4).

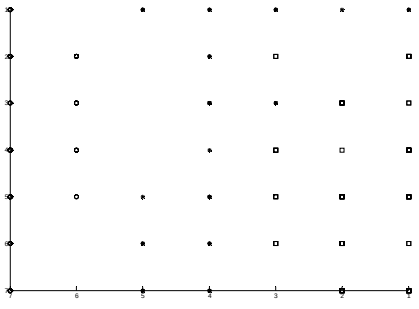

Figure 6. Result of the standard Kohonen's SOM on the IRIS data set (7x7).

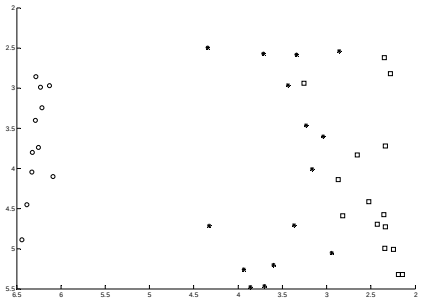

Figure 7. Result of PASOM on the IRIS data set (7x7, *cf*=0.25, *threshold*=500).

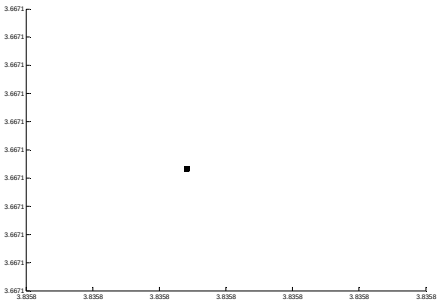

Figure 8. Result of PASOM on the IRIS data set (7x7, *cf*=0.25, *threshold*=50).

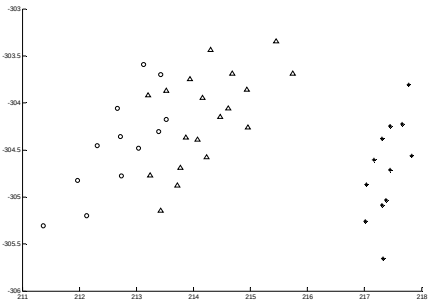

Figure 9. Result of DPSOM on the IRIS data set (7x7).

From Fig. 6, we can see that the standard Kohonen's SOM can map the data samples in the same group onto neighboring neurons and thus visualize the cluster structure of the IRIS dataset to a certain extent (different kinds of markers mean different groups of the data), but the structure is visualized distortedly so that the boundaries of these three groups are blurring, because the neurons are fixed onto the regular lattice. Compared with the standard Kohonen's SOM, PASOM with the proper additional parameters (e.g. *cf*=0.25 and *threshold*=500 in Fig. 7) and DPSOM (seen in Fig. 9) can visualize the cluster structure of the IRIS dataset more faithfully and naturally so that the boundaries of these three groups are relatively clear (the boundary of two groups is still blurring because these two groups are overlapping), because the positions of all the neurons on the map are adjusted according to the corresponding similarities. However, the neurons can be contracted excessively in PASOM with the improper additional parameters (e.g. *cf*=0.25 and *threshold*=50 in Fig. 8).

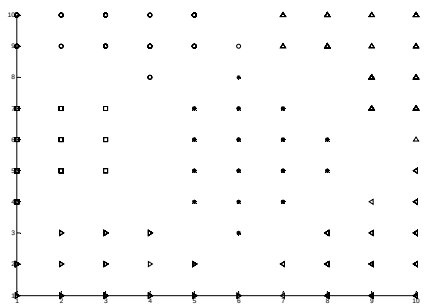

Figure 10. Result of the standard Kohonen's SOM on the Gaussian5d data set (10x10).

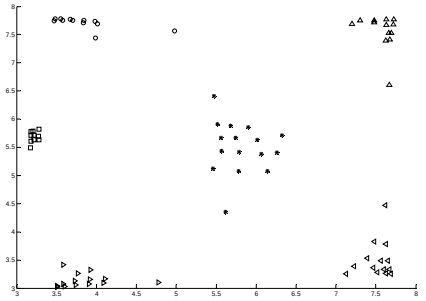

Figure 11. Result of PASOM on the Gaussian5d data set (10x10, *cf*=0.25, *threshold*=200).

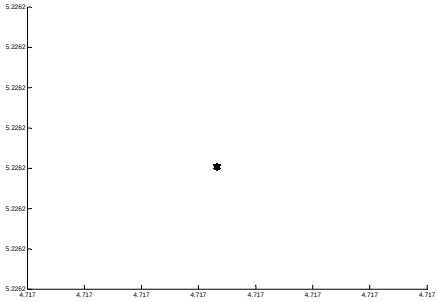

Figure 12. Result of PASOM on the Gaussian5d data set (10x10, *cf*=0.25, *threshold*=50).

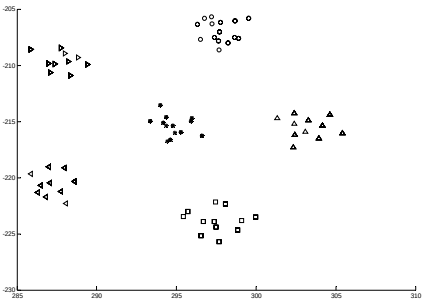

Figure 13. Result of DPSOM on the Gaussian5d data set (10x10).

From Fig. 10, we can see that the standard Kohonen's SOM can map the data samples in the same group onto neighboring neurons and thus visualize the cluster structure of the Gaussian5d dataset to a certain extent (different kinds of markers mean different groups of the data), but the structure is visualized distortedly so that the boundaries of these six groups are blurring, because the neurons are fixed onto the regular lattice. Compared with the standard Kohonen's SOM, PASOM with the proper additional parameters (e.g. *cf*=0.25 and *threshold*=200 in Fig. 11) and DPSOM (seen in Fig. 13) can visualize the structure of the Gaussian5d dataset more faithfully and naturally so that the boundaries of these six groups are relatively clear, because the positions of all the neurons on the map are adjusted according to the corresponding similarities. However, DPSOM can visualize the normal distribution of each group more precisely than PASOM. In addition, the neurons can be contracted excessively in PASOM with the improper additional parameters (e.g. *cf*=0.25 and *threshold*=50 in Fig. 12).

Unlike PASOM which requires proper additional parameters *cf* and *threshold* to obtain relatively good results (seen in Fig. 3 on the butterfly dataset, Fig. 7 on the IRIS dataset and Fig. 11 on the Gaussian5d dataset), DPSOM can visualize the structure of the data very well without any additional parameters (seen in Fig. 5 on the butterfly dataset, Fig. 9 on the IRIS dataset and Fig. 13 on the Gaussian5d dataset). The reason is that DPSOM adjusts the positions of neurons on the map precisely according to the corresponding distances in the data space in the way similar to MDS; however, PASOM uses Himberg's contraction model which contracts the neurons to one point gradually, and thus requires proper additional parameters to avoid the excessive contraction of neurons to one point, and these additional parameters influence the quality of the result maps greatly. Therefore, DPSOM has better controllability than PASOM. In addition, for the same reason, the results of DPSOM are more precise than those of PASOM.

As a variant of SOM, DPSOM can represent the data more concisely than MDS due to SOM's vector quantitation, for example, the number of neurons used in DPSOM can be much less than that of the data samples (also seen in Fig. 14).

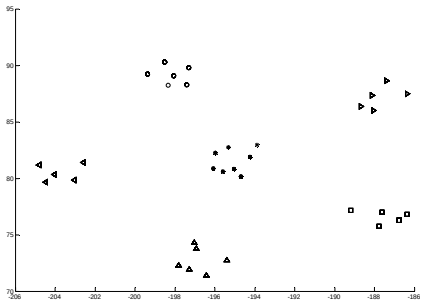

Figure 14. Result of DPSOM on the Gaussian5d data set (7x7).

V. CONCLUSIONS

In this paper, we presented a novel variant of SOM, i.e. Distance-Preserving SOM (DPSOM), which can adaptively adjust the positions of neurons on the map according to the corresponding distances in the data space in the way similar to MDS, the distance information can be preserved on the map relatively precisely, and thus the

structure of the data can be visualized faithfully and naturally.

In addition, as a type of position-adjustable SOMs, DPSOM has the advantages of other position-adjustable SOMs too, such as the robustness and simplicity. What's the most important, DPSOM can automatically avoid the problem of the excessive contraction of neurons to one point without any additional parameter that other position-adjustable SOMs need, which makes DPSOM more advantageous over other position-adjustable SOMs, that is, DPSOM has better controllability than other position-adjustable SOMs. As the combination of SOM and MDS, DPSOM can obtain natural and concise visualization results.

However, due to its intrinsic similarity to MDS, like other position-adjustable SOMs, DPSOM does not work very well on such datasets as Swiss-roll and S-curve, which need the manifold learning methods such as ISOMAP and LLE.

ACKNOWLEDGMENT

This work was supported in part by Research Programme of Henan Fundamental and Advanced Technology of China (No. 112300410201), and the Key Technologies R&D Programme of Henan Province, China (No. 102102210400).

REFERENCES

[1] W. S. Cleveland, *Visualizing Data*. New Jersey, U.S.A.: Hobart Press, 1993.
[2] D. A. Keim, "Designing pixel-oriented visualization techniques: theory and applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 59-78, 2000.
[3] A. Inselberg and B. Dimsdale, "Parallel coordinates: a tool for visualizing multidimensional geometry," in Proc. IEEE Visualization 1990, San Francisco, California, October 23-25, 1990, pp.361-378.
[4] E. Kandogan, "Visualizing multi-dimensional clusters, trends, and outliers using star coordinates," in Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, August 26-29, 2001, pp.107-116.
[5] J. LeBlanc, M. O. Ward, and N. Wittels, "Exploring n-dimensional databases," in Proc. IEEE Visualization 1990, San Francisco, California, October 23-25, 1990, pp.230-239.
[6] B. Johnson, *Visualizing hierarchical and categorical data*. Ph.D. Thesis, Department of Computer Science, University of Maryland, 1993.
[7] E. R. Tufte, *The visual display of quantitative information*. Cheshire, CT: Graphics Press, 1983.
[8] R. M. Pickett and G. G. Grinstein, "Iconographic displays for visualizing multidimensional data," in Proc. 1988 IEEE Conference on Systems, Man and Cybernetics, Beijing and Shenyang, China, August 8-12, 1988, pp.514-519.
[9] J. C. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 296-317, 1995.
[10] A. Naud and W. Duch, "Interactive data exploration using MDS mapping," in Proc. 5th Conference on Neural Networks and Soft Computing, Zakopane, Poland, June 5-10, 2000, pp.255-260.
[11] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biol. Cybern.*, vol. 43, pp. 59-69, 1982.
[12] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 22, pp. 2319-2323, 2000.
[13] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 22, pp. 2323-2326, 2000.
[14] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373-1396, 2003.
[15] J. Vesanto, "SOM-Based Data Visualization Methods," *Intelligent Data Analysis*, Vol. 3, pp. 111-126, 1999.
[16] H. Yin, "ViSOM - a Novel Method for Multivariate Data Projection and Structure Visualization," *IEEE transactions on Neural networks*, vol. 13, pp. 237 - 243, 2002.
[17] C. Shao and H. Huang, "Improvement of Data Visualization Based on SOM," in Proc. 2004 International Symposium on Neural Networks, Dalian, China, August 19-21, 2004, pp. 707-712.
[18] M. Rubio and V. Gimnez, "New Methods for Self-Organising Map Visual Analysis," *Neural Comput & Applic*, vol. 12, pp. 142-152, 2003.
[19] D. Merkl and A. Rauber, "Alternative Ways for Cluster Visualization in Self-Organizing Maps," in Proc. 1997 Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6, 1997, pp. 106-111.
[20] D. Wang, H. Ressom, M. Musavi, and C. Domnisoru, "Double Self-Organizing Maps to Cluster Gene Expression Data," in Proc. 2002 European Symposium on Artificial Neural Networks, Bruges, Belgium, April 24-26, 2002, pp. 45-50.
[21] J. Himberg, "A SOM based cluster visualization and its application for false coloring," in Proc. 2000 International Joint Conference on Neural Networks, Como, Italy, July 24-27, 2000, pp. 587-592.

Chao Shao, male, born in Henan Province, China on December 12, 1977. He received his PH.D. degree in computer application technology from Beijing Jiaotong University, Beijing, China in 2006.

He is currently an ASSOCIATE PROFESSOR of School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou, Henan Province, China. His research fields include artificial neural network, machine learning, data mining and data visualization, *etc*.

Dr. Shao is also a member of China Computer Federation.

Yongqiang Yang, male, born in Henan Province, China on July 12, 1974. He received his Master degree in computer application technology from PLA Information Engineering University, Zhengzhou, Henan Province, China in 2005.

He is currently a LECTURER of School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou, Henan Province, China. His research fields include computer network, grid computing, *etc*.