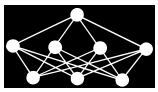


MANIFOLD LEARNING AND VISUALIZATION BASED ON DYNAMIC SELF-ORGANIZING MAP

Chao Shao[*], *Chunhong Wan*[†], *Haitao Hu*[‡]

Abstract: For the data sampled from a low-dimensional nonlinear manifold embedded in a high-dimensional space, such as Swiss roll and S-curve, Self-Organizing Map (SOM) tends to get stuck in local minima and then yield topological defects in the final map. To avoid this problem and obtain more faithful visualization results, a variant of SOM, i.e. Dynamic Self-Organizing Map (DSOM), was presented in this paper. DSOM can dynamically increase the map size, as the training data set is expanded according to its intrinsic neighborhood structure, starting from a small neighborhood in which the data points can lie on or close to a linear patch. According to the locally Euclidean nature of the manifold, the map can be guided onto the manifold surface and then the global faithful visualization results can be achieved step by step. Experimental results show that DSOM can discover the intrinsic manifold structure of the data more faithfully than SOM. In addition, as a new manifold learning method, DSOM can obtain more concise visualization results and be less sensitive to the neighborhood size and the noise than typical manifold learning methods, such as Isometric Mapping (ISOMAP) and Locally Linear Embedding (LLE), which can also be verified by experimental results.

Key words: *Manifold learning, self-organizing map, topological defect, neighborhood structure, robustness*

Received: August 21, 2012 **DOI:** 10.14311/NNW.2015.25.009
Revised and accepted: April 10, 2015

1. Introduction

As the combination of vector quantization and nonlinear dimensionality-reduction mapping, Self-Organizing Map [3] (SOM) can map high-dimensional data onto a low-dimensional regular lattice of neurons, while preserving the topological relationship between data points as faithfully as possible, which makes it a popular clustering, visualization and abstraction tool.

[*]Chao Shao – Corresponding Author, School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450002, China, Tel.: +8618503885361, E-mail: sc_flying@163.com

[†]Chunhong Wan, School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450002, China, Tel.: +8618503885508, E-mail: scwch@huel.edu.cn

[‡]Haitao Hu, School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450002, China, Tel.: +8618503885399, E-mail: frank.h@163.com

For the data sampled from a low-dimensional nonlinear manifold embedded in a high-dimensional space, such as Swiss roll and S-curve, typical manifold learning methods, such as Isometric Mapping [13] (ISOMAP), Locally Linear Embedding [7] (LLE) and Laplacian Eigenmap [1] (LE), can discover its intrinsic manifold structure nicely, but cannot summarize it, which hampers the effective analysis of mass data sets [11]. Compared with these typical manifold learning methods, SOM can obtain more concise visualization results due to its vector quantization feature, which makes it very suitable for visualizing mass data sets. However, when the nonlinear structure of the data cannot simply be regarded as a perturbation from a linear approximation, the iterative approach used by SOM has a tendency to get stuck in local minima and then yield topological defects in the final map [4, 5, 10, 11], which makes SOM not faithfully discover the intrinsic manifold structure of the data sampled from a low-dimensional nonlinear manifold embedded in a high-dimensional space [11, 12].

Based on the locally Euclidean nature of the manifold, this paper presents a variant of SOM, i.e. Dynamic Self-Organizing Map (DSOM). DSOM can dynamically increase the map size, as the training data set is expanded according to its intrinsic neighborhood structure, starting from a small neighborhood in which the data points can lie on or close to a linear patch. In this way, DSOM can guide the map onto the manifold surface instead of getting stuck in local minima, and then can avoid the topological defect problem and faithfully discover the intrinsic manifold structure of the data in the end.

The rest of the paper is organized as follows. Section 2 briefly reviews SOM and the related methods. Section 3 describes DSOM in detail. Section 4 uses two widely-used data sets to verify the effectiveness of DSOM. Section 5 gives the conclusion of the paper.

2. SOM and the related methods

SOM consists of a single layer of neurons located on a low-dimensional regular lattice, usually 1-D or 2-D for visualization. Each neuron i is represented by a d-dimensional weight vector $\mathbf{w}_i = \{w_{i1}, \cdots, w_{id}\}$, where d is the dimensionality of the data. On each training step, a data point $\mathbf{x}$ is selected randomly, and its best-matching unit (BMU), denoted as $c(\mathbf{x})$, is selected according to the following rule:

$$c(\mathbf{x}) = \arg\min_i ||\mathbf{w}_i - \mathbf{x}||. \tag{1}$$

After that, a neighborhood learning is adopted, that is, the weight vectors of $c(\mathbf{x})$ and its neighbors, defined by a neighborhood kernel $h_{c(\mathbf{x})\cdot}(t)$ in the lattice, are updated towards $\mathbf{x}$ according to the following rule:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) \times h_{c(\mathbf{x})i}(t) \times (\mathbf{x} - \mathbf{w}_i(t)), \tag{2}$$

where $\alpha(t)$ is the learning rate and $h_{c(\mathbf{x})i}(t)$ is the neighborhood kernel which usually takes the following form:

$$h_{c(\mathbf{x})i}(t) = \mathrm{e}^{-\frac{||\mathbf{p}_i - \mathbf{p}_{c(\mathbf{x})}||^2}{2\delta(t)^2}}, \tag{3}$$

where $\mathbf{p}_{c(\mathbf{x})}$ and $\mathbf{p}_i$ are the position vectors of $c(\mathbf{x})$ and its neighbor i in the lattice respectively, and $\delta(t)$ is the width of the neighborhood kernel. To ensure the convergence, both $\alpha(t)$ and $\delta(t)$ should decrease monotonically with time t.

Because not only the BMU but also its neighbors are updated towards the same direction, the weight vectors of neighboring neurons resemble each other [14]. Consequently, the BMUs of similar data points are close to each other in the lattice, which is the SOM's topology preserving property. As a nonlinear dimensionality-reduction method, SOM uses iterative optimization to fit a regular lattice with a predefined low-dimensional topology to the training data, and thus tends to get stuck in local minima depending on the initial conditions and yield topological defects in the final map, especially for highly nonlinear data sets [11, 12].

For the data sampled from a low-dimensional nonlinear manifold embedded in a high-dimensional space, such as Swiss roll and S-curve, several variants of SOM, such as M-SOM [11], GDBSOM [10] and ISOSOM [2], were presented to avoid the topological defect problem and then faithfully discover its intrinsic manifold structure; however, they need to run a certain typical manifold learning method, such as LLE or ISOMAP, beforehand to obtain the internal coordinates of data points on the manifold or geodesic distances between them, so these variants of SOM have the same practical limits as typical manifold learning methods, for example, they need a predefined suitable neighborhood size which is difficult to select efficiently [9].

3. DSOM

As indicated in [10–12], when the structure of the data is linear or almost linear, the iterative approach used by SOM can easily avoid getting stuck in local minima and then yield a global or almost global solution. So, for the data sampled from a low-dimensional nonlinear manifold embedded in a high-dimensional space, to avoid getting stuck in local minima and then faithfully discover its intrinsic manifold structure, based on the locally Euclidean nature of the manifold, the map should be trained first within a small neighborhood, in which the data points can be ensured to lie on or close to a linear patch, and then the first faithful solution can be achieved easily. After that, as the training data set is expanded gradually according to its intrinsic neighborhood structure, the global faithful solution can be achieved step by step. However, the map has to be trained multiple times, which makes this method very time-consuming. To enhance the efficiency of this method, the map size should also be increased, starting from a small map size, synchronously as the training data set is expanded, which is the thinking of DSOM.

To do this, unlike those variants of SOM described in Section 2, DSOM needs only to adopt the k-nearest neighbors method (the same as in typical manifold learning methods) to obtain the neighborhood structure of the data, while not running the other more time-consuming parts of typical manifold learning methods, such as shortest path computation and the eigenvalue decomposition.

After obtaining the neighborhood structure of the data, DSOM adopts the two-order neighborhood of a certain data point (including this data point, its neighbors and the neighbors of its neighbors) as the first training data set. To ensure that the first training data set lies on or close to a linear patch, DSOM computes the PCA

(Principal Component Analysis) reconstruction error of the two-order neighborhood of each data point and selects the one with the minimal PCA reconstruction error as the first training data set. Formally, let $\mathbf{N}_k(\mathbf{x})$ be the set of the neighbors of data point $\mathbf{x}$ (including $\mathbf{x}$ itself) in the data space with the neighborhood size being k, then $\mathbf{N}_k(\mathbf{N}_k(\mathbf{x}))$ is the two-order neighborhood of data point $\mathbf{x}$, and let $\epsilon(\mathbf{X})$ be the PCA reconstruction error of the data set $\mathbf{X}$, then the first training data set, denoted as $\mathbf{D}(0)$, can be obtained according to the following rule:

$$\mathbf{D}(0) = \mathbf{N}_k(\mathbf{N}_k(\mathbf{x}_s)), \tag{4}$$

where $\mathbf{x}_s = \arg\min_{\mathbf{x}} \epsilon(\mathbf{N}_k(\mathbf{N}_k(\mathbf{x})))$.

To ensure that the training data set is expanded according to its intrinsic neighborhood structure and make DSOM less sensitive to the neighborhood size, DSOM adopts a simple method that new data point and a certain data point in the current training data set should be in the neighborhood of each other [6]. Formally, let $\mathbf{D}_{\text{old}}$ be the current training data set, then the set of new data points, denoted as $\mathbf{D}_{\text{new}}$, can be described as follows:

$$\mathbf{D}_{\text{new}} = \{\mathbf{x} | \mathbf{x} \in \mathbf{N}_k(\mathbf{y}) \wedge \mathbf{y} \in \mathbf{N}_k(\mathbf{x}) \wedge \mathbf{x} \notin \mathbf{D}_{\text{old}} \wedge \mathbf{y} \in \mathbf{D}_{\text{old}}\}. \tag{5}$$

On each expansion of the training data set, the map size is increased according to the distribution of new data points. In the DSOM, the neurons are located on a rectangular lattice with four boundaries, and a row or column of new neurons is added outside the boundary along which the average number of new data points is maximal. Formally, let $\mathbf{NoB}(i)$ be the set of neurons on the i-th boundary where i can be set from 1 to 4 (representing four boundaries of the lattice), then the sets of the current training data points and new data points along the i-th boundary, denoted as $\mathbf{D}_{\text{old}}^{(i)}$ and $\mathbf{D}_{\text{new}}^{(i)}$ respectively, can be described as follows:

$$\mathbf{D}_{\text{old}}^{(i)} = \{\mathbf{x} | c(\mathbf{x}) \in \mathbf{NoB}(i) \wedge \mathbf{x} \in \mathbf{D}_{\text{old}}\}, \tag{6}$$

$$\mathbf{D}_{\text{new}}^{(i)} = \{\mathbf{x} | \mathbf{x} \in \mathbf{N}_k(\mathbf{y}) \wedge \mathbf{y} \in \mathbf{N}_k(\mathbf{x}) \wedge \mathbf{x} \notin \mathbf{D}_{\text{old}} \wedge \mathbf{y} \in \mathbf{D}_{\text{old}}^{(i)}\}, \tag{7}$$

and then a row or column of new neurons is added outside the b-th boundary where

$$b = \arg\max_i \frac{|\mathbf{D}_{\text{new}}^{(i)}|}{|\mathbf{NoB}(i)|}, \tag{8}$$

$|\mathbf{D}_{\text{new}}^{(i)}|$ and $|\mathbf{NoB}(i)|$ are the number of elements in the sets $\mathbf{D}_{\text{new}}^{(i)}$ and $\mathbf{NoB}(i)$ respectively. To make these new neurons close to the manifold surface, we initialize their weight vectors, denoted as $\mathbf{w}_{\text{newn}}$, according to the following rule:

$$\mathbf{w}_{\text{newn}} = \mathbf{w}_{\mathbf{NoB}(b)} + \lambda \times (\mathbf{mv}(\mathbf{D}_{\text{new}}^{(b)}) - \mathbf{mv}(\mathbf{D}_{\text{old}}^{(b)})), \tag{9}$$

where $\mathbf{mv}(\mathbf{D}_{\text{new}}^{(b)})$ and $\mathbf{mv}(\mathbf{D}_{\text{old}}^{(b)})$ are the mean vectors of data points in the sets $\mathbf{D}_{\text{new}}^{(b)}$ and $\mathbf{D}_{\text{old}}^{(b)}$ respectively, and then $\mathbf{mv}(\mathbf{D}_{\text{new}}^{(b)}) - \mathbf{mv}(\mathbf{D}_{\text{old}}^{(b)})$ represents the expansion direction of the training data set on the b-th boundary, λ is a little coefficient.

During the first several expansions (decided by a threshold of DSOM which is denoted as f) of the training data set, the training data set is still limited within a

relatively small neighborhood and can still be regarded to lie on or close to a linear patch, so DSOM can still be run in the same way as SOM, that is, the BMU can still be selected based on Euclidean distance according to Eq. (1), and the neurons can still be updated according to Eq. (2).

After that, the training data set can no longer be ensured to lie on or close to a linear patch, so the BMU shouldn't be selected based on Euclidean distance according to Eq. (1) any longer, but based on geodesic distance, because only geodesic distance is meaningful along the manifold [8]; however, as addressed in [11], selection of the BMU based on geodesic distance is difficult to realize practically.

To avoid the topological defect problem and then faithfully discover the intrinsic manifold structure of the data, in the DSOM, the BMU of each data point is selected according to its neighborhood structure. Concretely speaking, the BMU of data point $\mathbf{x}$, i.e. $c(\mathbf{x})$, is selected among the neighbors (in the lattice, decided by another threshold of DSOM which is denoted as r) of the BMUs of the neighbors (in the data space, decided by the k-nearest neighbors method) of $\mathbf{x}$. Formally, let $\mathbf{N}'_r(i)$ be the set of the neighbors of neuron i (including i itself) in the lattice with the neighborhood size being r, then $c(\mathbf{x})$ is selected according to the following rule:

$$c(\mathbf{x}) = \underset{i \in \mathbf{N}'_r(c(\{\mathbf{y}|\mathbf{y}\in\mathbf{N}_k(\mathbf{x})\wedge\mathbf{x}\in\mathbf{N}_k(\mathbf{y})\wedge\mathbf{y}\in\mathbf{D}_{\text{old}}\}))}{\arg\min} ||\mathbf{w}_i - \mathbf{x}||. \tag{10}$$

Based on the locally Euclidean nature of the manifold, only each data point and its neighbors can be meaningfully regarded to lie on or close to a linear patch. Similarly, in the DSOM, on each training step with the selected data point $\mathbf{x}$, only $c(\mathbf{x})$ (the representative of $\mathbf{x}$ in the lattice) and its neighbors (in the lattice, decided by the threshold r too. Due to the SOM's topology preserving property, they should lie on or close to a linear patch around $\mathbf{x}$ or its BMU $c(\mathbf{x})$ in the final map) can be updated towards $\mathbf{x}$ according to Eq. (2), that is:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) \times h_{c(\mathbf{x})i}(t) \times (\mathbf{x} - \mathbf{w}_i(t)), i \in \mathbf{N}'_r(c(\mathbf{x})). \tag{11}$$

Except the neighboring neurons of $c(\mathbf{x})$, the other neurons can no longer be ensured to lie on or close to a linear patch around $\mathbf{x}$ or its BMU $c(\mathbf{x})$, they shouldn't be updated in the same way as the above any longer, in the DSOM, they are updated towards their direct neighbors to $c(\mathbf{x})$ in the lattice respectively according to the following rule:

$$\begin{aligned} \mathbf{w}_i(t+1) &= \mathbf{w}_i(t) \\ &+ \alpha(t) \times h_{c(\mathbf{x})i}(t) \times (\mathbf{w}_{i_1}(t) - \mathbf{w}_i(t))/d_1 \\ &+ \alpha(t) \times h_{c(\mathbf{x})i}(t) \times (\mathbf{w}_{i_2}(t) - \mathbf{w}_i(t))/d_2, i \notin \mathbf{N}'_r(c(\mathbf{x})), \end{aligned} \tag{12}$$

where i_1 and d_1 are the direct neighbor along the row of i to $c(\mathbf{x})$ and their distance along the row in the lattice, and i_2 and d_2 are the direct neighbor along the column of i to $c(\mathbf{x})$ and their distance along the column in the lattice. If i and $c(\mathbf{x})$ lie on the same column, the second part of the right side of Eq. (12) will be omitted; and if i and $c(\mathbf{x})$ lie on the same row, the third part of the right side of Eq. (12) will be omitted.

Consequently, DSOM can be described briefly as follows:

1) Obtain the neighborhood structure of the data using the k-nearest neighbors method with a predefined neighborhood size k, as typical manifold learning methods do;

2) Obtain the first training data $\mathbf{D}(0)$ according to Eq. (4);

3) Specify the number of rows and columns of neurons in the lattice at first, denoted as $r_1(0)$ and $r_2(0)$ respectively;

4) Initialize all the weight vectors with small random values;

5) Specify three parameters described in the above, i.e. f, r and λ;

6) Specify the initial learning rate $\alpha(0)$ and the initial width of the neighborhood kernel $\sigma(0)$;

7) $j = 0$; $t = 0$;

8) While $\mathbf{D}(j)$ is not the whole data set or the stop condition is not met

   a) Select a data point $\mathbf{x}$ from $\mathbf{D}(j)$ randomly;

   b) If $j \leq f$

      Select $c(\mathbf{x})$ according to Eq. (1);

   Else

      Select $c(\mathbf{x})$ according to Eq. (10);

   End

   c) For each neuron i

      If $j \leq f$

         Update $\mathbf{w}_i(t)$ according to Eq. (2);

      Else

         If $i \in \mathbf{N}'_r(c(\mathbf{x}))$

            Update $\mathbf{w}_i(t)$ according to Eq. (11);

         Else

            Update $\mathbf{w}_i(t)$ according to Eq. (12);

         End

      End

   End

   d) $t = t + 1$;

   e) Decrease $\alpha(t)$ and $\sigma(t)$;

   f) If $\mathbf{D}(j)$ is not the whole data set and the stop condition is met

      i) $\mathbf{D}(j+1) = \mathbf{D}(j) \cup \mathbf{D}_{\text{new}}$ according to Eq. (5);

ii) Add a row (i.e. $r_1(j+1) = r_1(j) + 1$) or column (i.e. $r_2(j+1) = r_2(j) + 1$) of new neurons according to Eq. (8);

iii) Initialize the weight vectors of new neurons, i.e. $\mathbf{w}_{\text{newn}}$, according to Eq. (9);

iv) $j = j + 1$; $t = 0$;

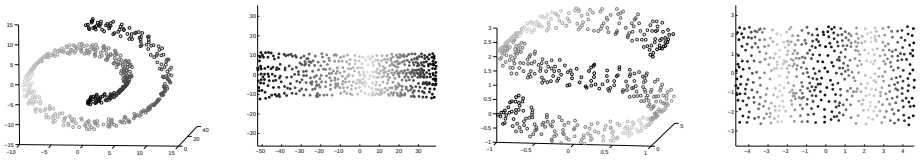
End

End

On each expansion of the training data set, new data points should be trained first several times (e.g. 10 times in the below experiments) to alleviate the imbalance among the training steps of all the data points.

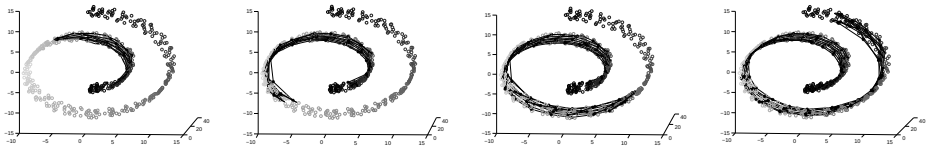
4. Experimental results

In the experiments, we use two widely-used data sets, i.e. Swiss roll and S-curve with the randomly selected 2000 data points, out of which $n = 500$ representative data points are selected using the K-means method in Matlab v7.0 toolboxes, as shown in Fig. 1(a) and Fig. 1(c) (their intrinsic manifold structures are shown in Fig. 1(b) and Fig. 1(d) respectively, represented in black and gray). Some snapshots during the map formation process of DSOM on these two data sets are shown in Fig. 2 and Fig. 3 respectively, from which we can see that the map of DSOM can be guided onto the manifold surface instead of getting stuck in local minima, and then can avoid the topological defect problem in the end (as shown in Fig. 6(a) and Fig. 7(a)).



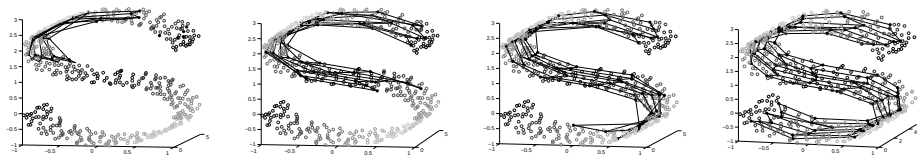
(a) Swiss roll (b) the intrinsic manifold structure of Swiss roll (c) S-curve (d) the intrinsic manifold structure of S-curve

Fig. 1 *Two data sets in the experiments.*



(a) the 11-th count (b) the 18-th count (c) the 25-th count (d) the 32-nd count

Fig. 2 *Some snapshots during the map formation process of DSOM on Swiss roll* ($k = 8$).

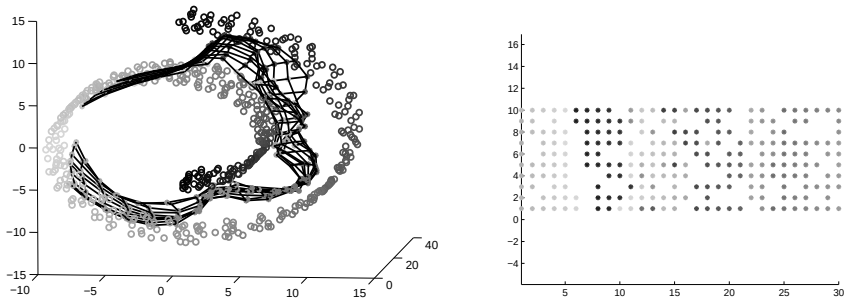


(a) the 6-th count (b) the 9-th count (c) the 12-th count (d) the 15-th count

Fig. 3 *Some snapshots during the map formation process of DSOM on S-curve* ($k = 16$).

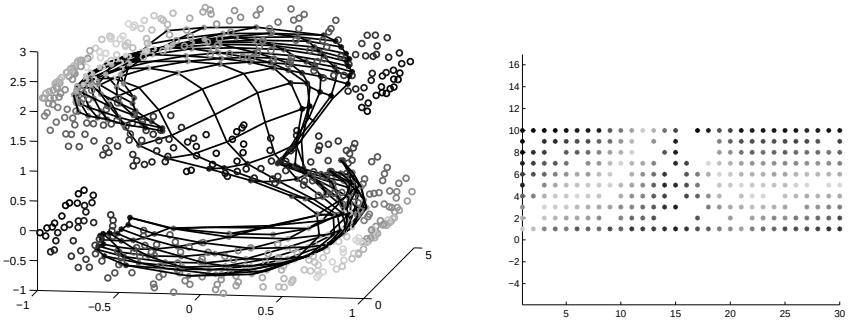
To verify the effectiveness of DSOM, we run SOM, DSOM, ISOMAP and LLE with different neighborhood size k on these two data sets. In the experiments, the parameters of SOM are specified as follows: the number of rows and columns of neurons are set to 10 and 30 respectively, $\alpha(t) = 0.9 \times 0.999^t$, $\sigma(t) = 30 \times 0.999^t$; and the parameters of DSOM are specified as follows: $r_1(0) = r_2(0) = \lfloor \frac{\sqrt{|D(0)|}}{3} \rfloor$, $\alpha(t) = 0.9 \times 0.999^t$, $\sigma(t) = r_1(0) \times 0.999^t$, $f = 5$, $r = 2$, $\lambda = 0.2$. The stop condition on each expansion of the training data set in the DSOM is the same as that in the SOM, in which the maximal number of iterations and the minimal error are set to 100000 and 0.00001 respectively. Besides these parameters, DSOM, ISOMAP and LLE have an additional parameter, i.e. the neighborhood size k, which is listed in the caption of the corresponding figure.

The visualization results of SOM on Swiss roll and S-curve are shown in Fig. 4 and Fig. 5 respectively, from which we can see that there are topological defects in the final maps (as shown in Fig. 4(a) and Fig. 5(a)), and then SOM cannot faithfully discover their intrinsic manifold structure (as shown in Fig. 4(b) and Fig. 5(b), where only the BMUs are displayed and colored in black and gray, according to the positions of data points, which are mapped onto them, in the corresponding manifolds, which can represent the manifold structures spanned by these BMUs, the same below).



(a) SOM's neurons in the data space (b) SOM's neurons in the lattice

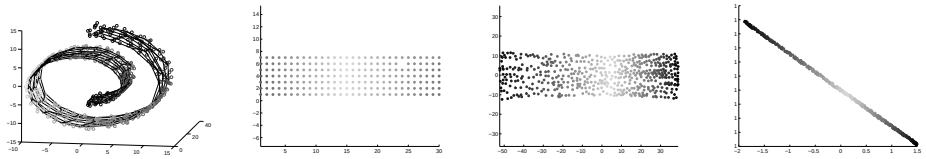
Fig. 4 *Visualization results of SOM on Swiss roll.*



(a) SOM's neurons in the data space (b) SOM's neurons in the lattice

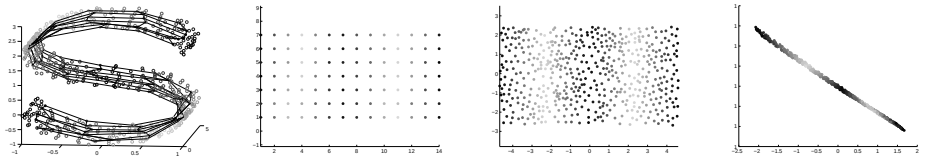
Fig. 5 *Visualization results of SOM on S-curve.*

To verify whether DSOM can faithfully discover the intrinsic manifold structure of the data like ISOMAP and LLE, we run DSOM, ISOMAP and LLE on Swiss roll and S-curve (with the optimal neighborhood size [9]), and the visualization results are shown in Fig. 6 and Fig. 7 respectively. As shown in Fig. 6(a) and Fig. 7(a), the maps are guided onto the manifold surfaces and there are no topological defects in the final maps. Consequently, DSOM can faithfully discover their intrinsic manifold structure (as shown in Fig. 6(b) and Fig. 7(b)) like ISOMAP (as shown in Fig. 6(c) and Fig. 7(c)) and LLE (as shown in Fig. 6(d) and Fig. 7(d)), which can be measured by residual variance to a certain extent; however, the goodness of the



(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 6 *Visualization results of DSOM, ISOMAP and LLE on Swiss roll* ($k = 8$).

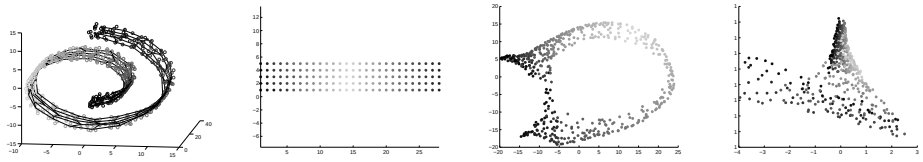


(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 7 *Visualization results of DSOM, ISOMAP and LLE on S-curve* ($k = 16$).

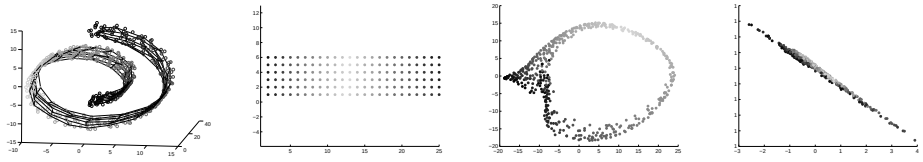
map of DSOM cannot be measured by residual variance like ISOMAP and LLE, because the neurons of DSOM are fixed on a regular low-dimensional lattice like SOM, regardless of the distances between them. In addition, DSOM can obtain more concise visualization results than ISOMAP and LLE, because the number of neurons in the DSOM is less than that of data points.

To verify whether DSOM is sensitive to the neighborhood size like ISOMAP and LLE, we also run DSOM, ISOMAP and LLE with different neighborhood size k on Swiss roll and S-curve, and the visualization results are shown in Fig. 8, Fig. 9, Fig. 10 and Fig. 11 respectively, from which we can see that DSOM can be less sensitive to the neighborhood size than ISOMAP and LLE, because DSOM can still faithfully discover their intrinsic manifold structure when the neighborhood size is too large, such as $k = 10$ and $k = 12$ for Swiss roll (as shown in Fig. 8(b) and Fig. 9(b)), and $k = 18$ and $k = 20$ for S-curve (as shown in Fig. 10(b) and Fig. 11(b)), but ISOMAP and LLE can't do this yet (as shown in Fig. 8(c), Fig. 8(d), Fig. 9(c), Fig. 9(d), Fig. 10(c), Fig. 10(d), Fig. 11(c) and Fig. 11(d)).



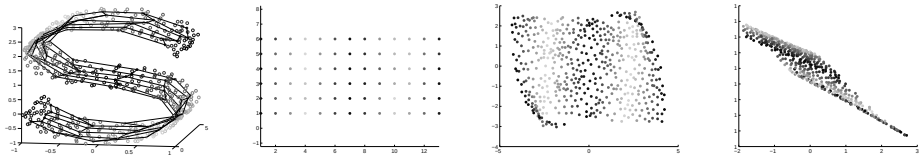
(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 8 *Visualization results of DSOM, ISOMAP and LLE on Swiss roll* ($k = 10$).



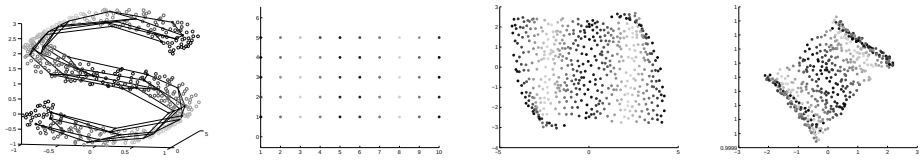
(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 9 *Visualization results of DSOM, ISOMAP and LLE on Swiss roll* ($k = 12$).



(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

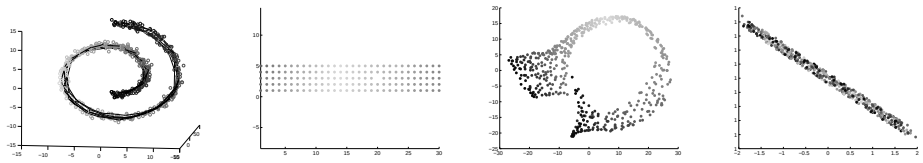
Fig. 10 *Visualization results of DSOM, ISOMAP and LLE on S-curve* ($k = 18$).



(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

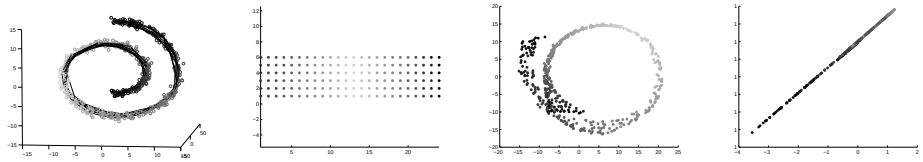
Fig. 11 *Visualization results of DSOM, ISOMAP and LLE on S-curve* ($k = 20$).

To verify whether DSOM is sensitive to the noise like ISOMAP and LLE, we also run DSOM, ISOMAP and LLE with different neighborhood size k on noisy Swiss roll and noisy S-curve, with zero-mean normally distributed noise added to each data point of the corresponding data set, where the standard deviation of the noise is chosen to be 2% of smallest dimension of the bounding box enclosing the data [13]. The visualization results are shown in Fig. 12, Fig. 13, Fig. 14 and Fig. 15 respectively, from which we can see that DSOM can be less sensitive to the noise than ISOMAP and LLE, because DSOM can still faithfully discover their intrinsic manifold structure when $k = 8$ and $k = 12$ for noisy Swiss roll (as shown in Fig. 12(b) and Fig. 13(b)) and $k = 16$ and $k = 20$ for noisy S-curve (as shown in Fig. 14(b) and Fig. 15(b)), but ISOMAP and LLE cannot do this yet, even when $k = 8$ for noisy Swiss roll (as shown in Fig. 12(c) and Fig. 12(d)) and $k = 16$ for noisy S-curve (as shown in Fig. 14(c) and Fig. 14(d)).



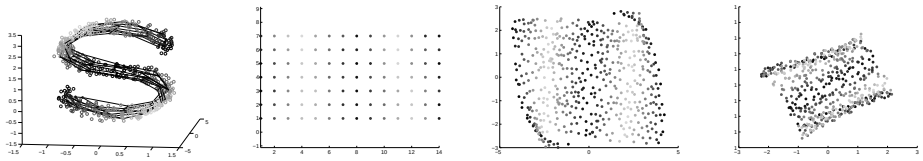
(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 12 *Visualization results of DSOM, ISOMAP and LLE on noisy Swiss roll* ($k = 8$).



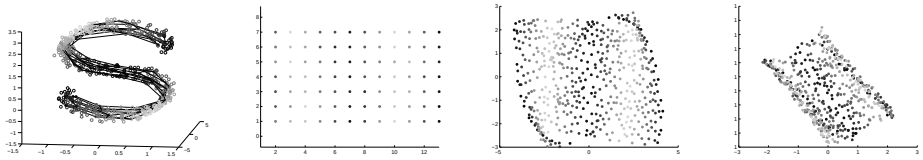
(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 13 *Visualization results of DSOM, ISOMAP and LLE on noisy Swiss roll* ($k = 12$).



(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 14 *Visualization results of DSOM, ISOMAP and LLE on noisy S-curve* ($k = 16$).



(a) DSOM's neurons in the data space (b) DSOM's neurons in the lattice (c) ISOMAP (d) LLE

Fig. 15 *Visualization results of DSOM, ISOMAP and LLE on noisy S-curve* ($k = 20$).

As described above, the goodness of the map of DSOM should be better than that of SOM, as shown in Fig. 4(a) and Fig. 6(a), which can be measured by the reconstruction error, as listed in Tab. I. However, the reconstruction errors of DSOM are a bit larger than that of SOM on S-curve and noisy S-curve, the reason for which is that the number of neurons of DSOM are much smaller than that of SOM on S-curve and noisy S-curve, i.e. 98 for DSOM vs 300 for SOM, and then a neuron of DSOM represents much more data points and naturally produces larger reconstruction error than SOM (by contrast, the number of neurons of DSOM are a bit smaller than that of SOM on Swiss roll, i.e. 210 for DSOM vs 300 for SOM).

data set	Swiss roll (k=8)	S-curve (k=16)	noisy Swiss roll (k=8)	noisy S-curve (k=16)
DSOM	761.662	38.140	1009.501	39.034
SOM	2159.165	33.815	1755.952	37.065

Tab. I *The reconstruction errors of DSOM and SOM on different data sets.*

5. Conclusion

To avoid getting stuck in local minima and then faithfully discover the intrinsic manifold structure of the data, this paper presents a variant of SOM, i.e. Dynamic Self-Organizing Map (DSOM), in which the map size can be increased synchronously as the training data set is expanded according to its intrinsic neighbor-

hood structure. Based on the locally Euclidean nature of the manifold, DSOM can guide the map onto the manifold surface and then faithfully discover the intrinsic manifold structure of the data. In addition, DSOM can automatically determine the number and distribution of neurons in the final map. So DSOM has better visualization capabilities than SOM for the data sampled from a low-dimensional nonlinear manifold embedded in a high-dimensional space.

Compared with other variants of SOM, such as M-SOM, GDBSOM and ISOSOM, DSOM needs only to adopt the k-nearest neighbors method to obtain the neighborhood structure of the data, while not running the other more time-consuming parts of typical manifold learning methods, such as shortest path computation and the eigenvalue decomposition. In addition, compared with typical manifold learning methods, such as ISOMAP, LLE and LE, DSOM can obtain more concise visualization results and be less sensitive to the neighborhood size and the noise.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No.61202285, the Research Programme of Henan Fundamental and Advanced Technology of China under Grant No. 112300410201, and the Key Science & Technology Research Programme of Educational Commission of Henan Province of China under Grant No. 14B520020.

References

[1] BELKIN M., NIYOGI P. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*. 2003, 15(6), pp. 1373-1396, doi: 10.1162/089976603321780317.

[2] GUAN H., TURK M. 3D hand pose reconstruction with ISOSOM. In: *Proceedings of the 1st international conference on Advances in Visual Computing (ISVC 2005)*, Lake Tahoe, Nevada: Springer-Verlag, 2005, pp. 630-635.

[3] KOHONEN T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*. 1982, 43(1), pp. 59-69, doi: 10.1007/BF00337288.

[4] MURAKOSHI K., SATO Y. Reducing topological defects in self-organizing maps using multiple scale neighborhood functions. *Biosystems*. 2007, 90(1), pp. 101-104, doi: 10.1016/j.biosystems.2006.07.004.

[5] OTA K., AOKI T., KURATA K., AOYAGI T. Asymmetric neighborhood functions accelerate ordering process of self-organizing maps. *Physical Review E*. 2011, 83(2 Pt 1), pp. 021903-(1-9), doi: 10.1103/PhysRevE.83.021903.

[6] OZAKI K., SHIMBO M., KOMACHI M., MATSUMOTO Y. Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data. In: *Proceedings of the 15th Conference on Computational Natural Language Learning (CoNLL 2011)*, Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 154-162.

[7] ROWEIS S., SAUL L. Nonlinear dimensionality reduction by locally linear embedding. *Science*. 2000, 290(5500), pp. 2323-2326, doi: 10.1126/science.290.5500.2323.

[8] SAXENA A., GUPTA A., MUKERJEE A. Non-linear dimensionality reduction by locally linear isomaps. In: *Proceedings of the 11th International Conference on Neural Information Processing (ICONIP 2004)*, Calcutta, India: Springer-Verlag, 2004, pp. 1038-1043.

[9] SHAO C., WAN C. Selection of the neighborhood size for manifold learning based on Bayesian information criterion. *Journal of Computational Information Systems*. 2012, 8(7), pp. 3043-3050. Available from: `http://www.jofcis.com/publishedpapers/2012_8_7_3043_3050.pdf`

[10] SHI C., ZHANG S., SHI Z.Z. Geodesic distance based SOM for image clustering. In: *Proceedings of the 2006 International Conference on Sensing, Computing and Automation (ICSCA 2006)*, Chongqing, China: Watam, 2006, pp. 2483-2488.

[11] SIMILÄ T. Self-organizing map learning nonlinearly embedded manifolds. *Information Visualization*. 2005, 4(1), pp. 22-31, doi: 10.1057/palgrave.ivs.9500088.

[12] TENENBAUM J. Mapping a manifold of perceptual observations. In: *Proceedings of the 1997 conference on Advances in neural information processing systems (NIPS 1997)*, Cambridge, MA, USA: MIT, 1997, pp. 682-688.

[13] TENENBAUM J., DE SILVA V., LANGFORD J. A global geometric framework for nonlinear dimensionality reduction. *Science*. 2000, 290(5500), pp. 2319-2323, doi: 10.1126/science.290.5500.2319.

[14] VESANTO J. SOM-based data visualization methods. *Intelligent Data Analysis*. 1999, 3(2), pp. 111-126, doi: 10.1016/S1088-467X(99)00013-X.