Topology Learning Embedding: A Fast and Incremental Method for Manifold Learning

Tao Zhu, Furao Shen(✉), Jinxi Zhao, and Yu Liang

National Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology,
Collaborative Innovation Center of Novel Software Technology and Industrialization,
Nanjing University, Nanjing, People's Republic of China
tao144@outlook.com, {frshen,jxzhao}@nju.edu.cn,
mg1533023@smail.nju.edu.cn

Abstract. In this paper, we propose a novel manifold learning method named topology learning embedding (TLE). The key issue of manifold learning is studying data's structure. Instead of blindly calculating the relations between each pair of available data, TLE learns data's internal structure model in a smarter way: it constructs a topology preserving network rapidly and incrementally through online input data; then with the Isomap-based embedding strategy, it achieves out-of-sample data embedding efficiently. Experiments on synthetic data and real-world handwritten digit data demonstrate that TLE is a promising method for dimensionality reduction.

Keywords: Dimensionality reduction · Manifold learning · Incremental learning · Neural network · SOINN

1 Introduction

A manifold $\mathcal{M}$ is a topological space that is locally Euclidean, meaning that each point of $\mathcal{M}$ has a neighbourhood that is homeomorphic to an open subset of Euclidean space. Since the late 1990s, manifold learning has attracted great interest and many new methods have been proposed for dimensionality reduction (DR) [1–5].

The key problem of DR is finding the meaningful low-dimensional structures hidden in the high-dimensional observations [1]. The basic assumption of manifold learning is that the data of interest lie on a low dimensional manifold $\mathcal{M}$. As $\mathcal{M}$ locally resembles Euclidean space near each point, theoretically, its local structure can be learned by extracting the linear relations between the data that are close enough to each other. Unfortunately, due to the curse of dimensionality [6], in the absence of simplifying assumptions, if the neighborhood radius is fixed, the ideal number of data samples required for manifold learning should grow rapidly as the number of dimensions increases. In other word, it is difficult to collect enough samples for perfect manifold learning.

D. Liu et al. (Eds.): ICONIP 2017, Part I, LNCS 10634, pp. 43–52, 2017.
https://doi.org/10.1007/978-3-319-70087-8_5

As a result of that, almost all the manifold learning algorithms that have the ability of nonlinear DR learn manifold structure in a trivial way: For all the N available data, each of them is employed as a representative node and the $O(N^2)$ pair-wise relations between them are calculated. Combining the N representative nodes and the relations between them, a neighborhood graph contains N nodes can be generated to approximate the structure of $\mathcal{M}$. Then the mapping is done based on the collected structure information to obtain the data's low-dimensional representations. The above strategy is so natural that people often tend to turn a blind eye to its obvious deficiency: the huge storage space cost and computational complexity may become a bottleneck for its applications in large scale problems. Usually, typical nonlinear manifold learning methods are only applied on the dataset whose size is less than 10K.

When N is large but the computing resources are limited, as the large number of representative nodes is main cause of the great cost of nonlinear manifold learning, we believe that trying to find a much smaller set of suitable representative nodes and employ them to approximate $\mathcal{M}$'s structure is a reasonable choice. Though it may inevitably cause loss of precision, but it makes impossible tasks possible.

In this paper, by inheriting and developing the work of [7], we propose a novel DR method named topology learning embedding (TLE). Given an arbitrary large scale data set that assumed to lay on hidden manifold $\mathcal{M}$, instead of exhaustively calculating the pair-wise relations between all the data, based on the technology of self-organizing incremental neural network (SOINN) [8], TLE learns $\mathcal{M}$'s structure by constructing a topology preserving network $\mathfrak{G}$ that consists of a small number of automatically generated nodes and the adaptively established connections between them. Inspired by [9], each node of $\mathfrak{G}$ is represented as a isotropic multivariate normal distribution and stores the data's local statistical information. Network $\mathfrak{G}$ is updated incrementally through learning from the online data input in one pass. Once $\mathfrak{G}$ is obtained, by employing the Isomap-based "out-of-sample" data embedding strategy, TLE maps the data to their low-dimensional representations efficiently. Therefore, as a manifold learning method, TLE achieves incremental manifold topology structure learning and out-of-sample data embedding while enjoying the advantage of low computational and space complexity.

2 Topology Learning Embedding

Figure 1 illustrates the process of the proposed TLE method. $\mathcal{X}$ is the data set that is available for training. The low-dimensional representations of the data in $\mathcal{X}'$ are what we wanted. As TLE aims to achieve "out-of-sample" learning, $\mathcal{X}'$ may be not a subset of $\mathcal{X}$, but both of $\mathcal{X}$ and $\mathcal{X}'$ are assumed to have the same distribution and the elements of them lie on the same manifold $\mathcal{M}$. $\mathcal{Y}'$ is the set of $\mathcal{X}'$'s low-dimensional representations.

TLE contains two main steps:

1. Topology learning: constructing topology preserving network $\mathfrak{G}$ through learning from online input $\mathcal{X}$. This step achieves the goal of manifold structure extraction and is the main contribution of this work.
2. Data embedding: according to the relations between $\mathcal{X}'$ and $\mathfrak{G}$, employing Isomap-based strategy to obtain $\mathcal{Y}'$.

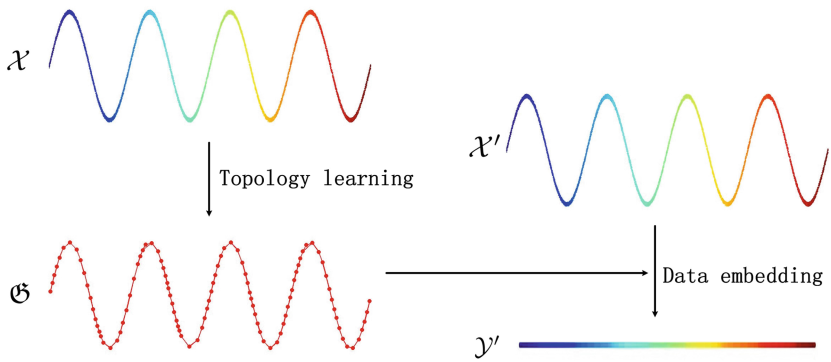


Fig. 1. The process of TLE

2.1 Topology Learning

For TLE, the extracted network $\mathfrak{G}$ is stored by the node set $\mathcal{S}$ and connection matrix $\boldsymbol{C}$.

Node set $\mathcal{S} = \{s_1, \ldots, s_n\}$ contains the n extracted nodes. For each s_i, we define it as a triple $\langle N_i,\ \boldsymbol{\mu}_i,\ D_i\rangle$. N_i denotes the number of times that node s_i has been updated. $\boldsymbol{\mu}_i \in \mathbb{R}^d$ is s_i's coordinate. D_i stores s_i's local information, it is employed in s_i's activation judgment and neighborhood determining.

$n \times n$ connection matrix $\boldsymbol{C}$ stores the neighborhood relations between the n extracted nodes. If nodes s_i and s_j are connected (they are neighbors), we have $[\boldsymbol{C}]_{i,j} = [\boldsymbol{C}]_{j,i} = 1$; otherwise, we have $[\boldsymbol{C}]_{i,j} = [\boldsymbol{C}]_{j,i} = 0$.

The process of manifold topology learning is shown in Fig. 2.

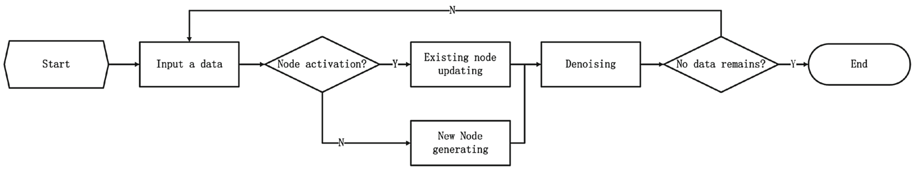


Fig. 2. The flowchart of the topology learning process

When a data $\boldsymbol{x}_t \in \mathbb{R}^d$ is input, node activation judgment is done. We try to find the winner node $s_{i_t^*}$ and the second winner node $s_{j_t^*}$ that are the two closest nodes to $\boldsymbol{x}_t$:

$$i_t^* = \arg_i \min dist(\boldsymbol{x}_t, s_i) \quad s_i \in \mathcal{S} \tag{1}$$

$$j_t^* = \arg_j \min dist(\boldsymbol{x}_t, s_j) \quad s_j \in \mathcal{S}/\{s_{i_t^*}\} \tag{2}$$

Here, $dist(\boldsymbol{x}_t, s_i) = \|\boldsymbol{x}_t - \boldsymbol{\mu}_i\|_2$. Then, the result of node activation determines how to update $\mathfrak{G}$.

Node Activation. We stipulate that when n the number of existing nodes is not less than 2 and current $\boldsymbol{x}_t$ is close enough to $s_{i_t^*}$, $\boldsymbol{x}_t$ successfully activates $s_{i_t^*}$.

For convenience, we assume that the data represented by s_i follow the distribution $\mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2 \boldsymbol{I}_d)$, $\boldsymbol{I}_d$ is the $d \times d$ identity matrix. For each $\boldsymbol{x} \in \mathcal{X}_i$, $\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_i\|_2^2}{\sigma_i^2}$ follows χ_d^2 distribution. Assume node s_i have been updated by N_i data $\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,N_i}$ $(N_i > 1)$, we estimate its $\boldsymbol{\mu}_i$ and σ_i^2 by:

$$\boldsymbol{\mu}_i = \frac{\sum_{l=1}^{N_i} \boldsymbol{x}_{i,l}}{N_i} \tag{3}$$

$$\sigma_i^2 = \varepsilon \frac{D_i}{d}, \quad D_i = \frac{\sum_{l=1}^{N_i} \|\boldsymbol{x}_{i,l} - \hat{\boldsymbol{\mu}}_i\|_2^2}{N_i}. \tag{4}$$

ε is the corrective parameter, with $f_d(u)$ that is the probability density function of χ_d^2 distribution, we manually set it as: $\varepsilon = 1.01 \frac{\int_0^\infty u f_d(u) du}{\frac{1}{p}\int_0^{F_d^{-1}(p)} u f_d(u) du}$. According to the property of χ_d^2 distribution, when d is large, ε can be directly set as 1.01.

Given an arbitrary $\boldsymbol{x}_t$, by pre-determining confidence p, we can determine whether $\boldsymbol{x}_t$ is likely to belong to $\mathcal{X}_{i_t^*}$ by checking whether

$$\frac{\|\boldsymbol{x}_t - \boldsymbol{\mu}_{i_t^*}\|_2^2}{\sigma_{i_t^*}^2} < F_d^{-1}(p) \tag{5}$$

is satisfied. $F_d(.)$ is the cumulative distribution function of χ_d^2 distribution. In this paper, p is set to be 0.99.

(5) is employed in $s_{i_t^*}$'s activation judgment. If (5) cannot be satisfied or n is less than 2, we believe $s_{i_t^*}$ cannot be activated by $\boldsymbol{x}_t$, and a new node s_{n+1} should be generated.

New Node Generating. The initialization of $s_{n+1} =< N_{n+1},\ \boldsymbol{\mu}_{n+1},\ D_{n+1} >$ is as follows:

$$N_{n+1} = 1,\ \boldsymbol{\mu}_{n+1} = \boldsymbol{x}_t,\ D_{n+1} = D_0. \tag{6}$$

D_0 is the initial setting about s_{n+1}'s local information and it defines what is "close" in the beginning. Obviously, for different data sets, it should be set respectively. Sometimes, we can set D_0 by analyzing a small subset of the whole large scale data set.

Existing Node Updating. Once winner $s_{i_t^*}$ has been successfully activated, firstly, we check whether the second winner node $s_{j_t^*}$ is close enough to $\boldsymbol{x}_t$ too $(\frac{\|\boldsymbol{x}_t-\boldsymbol{\mu}_{j_t^*}\|_2^2}{\sigma_{j_t^*}^2} < F_d^{-1}(p))$. If the answer is yes, we make sure that $s_{i_t^*}$ and $s_{j_t^*}$ are connected.

According to (3) and (4), the triple $\langle N_{i_t^*},\ \boldsymbol{\mu}_{i_t^*},\ D_{i_t^*}\rangle$ is updated by:

$$D_{i_t^*} \leftarrow \frac{N_{i_t^*}}{N_{i_t^*}+1}\left(D_{i_t^*} + \frac{\|\boldsymbol{x}_t-\boldsymbol{\mu}_{i_t^*}\|_2^2}{N_{i_t^*}+1}\right) \tag{7}$$

$$\boldsymbol{\mu}_{i_t^*} \leftarrow \frac{N_{i_t^*}}{N_{i_t^*}+1}\left(\boldsymbol{\mu}_{i_t^*} + \frac{\boldsymbol{x}_t}{N_{i_t^*}}\right) \tag{8}$$

$$N_{i_t^*} \leftarrow N_{i_t^*}+1 \tag{9}$$

Then, all the $K_{i_t^*}$ nodes that connected with $s_{i_t^*}$ are examined: If

$$\|\boldsymbol{\mu}_{i_t^*} - \boldsymbol{\mu}_{i_t^*(q)}\|_2 \geq \sqrt{\varepsilon\frac{F_d^{-1}(p)D_{i_t^*}}{d}} + \sqrt{\varepsilon\frac{F_d^{-1}(p)D_{i_t^*(q)}}{d}}, \tag{10}$$

we believe the formerly learned connection between $s_{i_t^*}$ and its current neighbor $s_{i_t^*(q)}$ is no longer suitable. Therefore, we disconnect $s_{i_t^*}$ from $s_{i_t^*(q)}$.

Denoising. Obviously, inappropriately extracted nodes may seriously influence the performance of manifold learning. Based on the factor that the statistical information obtained from insufficient samples may be unreliable and the assumption that the local area represented by well-generated node should be dense, after every t_0 input data have been learned, we calculate

$$\bar{N} = \frac{1}{n}\sum_{i=1}^{n} N_i, \quad N_e = e\bar{N} \tag{11}$$

and delete the nodes whose number of updating is less than N_e and the corresponding connections. Moreover, all the isolate nodes are eliminated. Here $\bar{N}$ is the mean updated times of all the n existing nodes, and e is the parameter controls the denoising power. In this paper, the default setting is $t_0 = 5000,\ e = 0.25$.

When all the N data have been processed, the denoising is done once more time (if $N \geq t_0$, this time we reuse the N_e calculated at the last time).

Algorithm Summary. The proposed topology learning algorithm is an online algorithm that processes the input data stream in one pass. In the beginning, the first and the second input data are employed to generated two isolate nodes. Then, the nodes and the connections between them are updated according to the input data. TLE's calculation complexity is $O(nNd)$ and its storage load is $O(dn + n^2)$. In the obtained network $\mathfrak{G}$, the number of nodes, the nodes' coordinates and the connections between nodes are all automatically determined.

Especially, the connections are established based on competitive Hebbian learning rule. According to [10], under the assumption of local density, we may have the conclusion that $\mathfrak{G}$ is able to form a topology preserving map of the given manifold and the learned connections forms a path preserving representation.

2.2 Data Embedding

When network $\mathfrak{G}$ has been obtained, we employ the technology introduced in [11, 12] for "out-of-sample" data embedding. To save space, we omit the detailed description of this existing technology.

The main difference between the data embedding of TLE and that of existing technology is that in TLE, geodesic distance from $\boldsymbol{x}' \in \mathcal{X}'$ to the extracted nodes are calculated only by $\boldsymbol{x}'$ itself and the n extracted nodes with no need for the N training data:

We find s_{i^*} that is the nearest node to $\boldsymbol{x}'$:

$$i^* = \arg_i \min dist(\boldsymbol{x}', s_i) \quad s_i \in \mathcal{S} \tag{12}$$

and define that $\boldsymbol{x}'$s neighborhood is the same with the neighborhood of s_{i^*}. We assume that s_{i^*}'s neighborhood contains s_{i^*} itself and its K_{i^*} connected neighbor nodes $s_{i^*(1)}, \ldots, s_{i^*(K_{i^*})}$. We define $s_{i^*(0)} = s_{i^*}$.

The geodesic distance from $\boldsymbol{x}'$ to the node s_j is calculated by:

$$dist_G(\boldsymbol{x}', s_j) = \min\{\|\boldsymbol{x}' - \boldsymbol{\mu}_{i^*(l)}\|_2 + [\boldsymbol{W}_G]_{i^*,j}\}_{l=0,1,\ldots,K_{i^*}}. \tag{13}$$

$[\boldsymbol{W}_G]_{i,j}$ denotes the previously calculated geodesic distance between nodes s_i and s_j.

3 Experiment

To demonstrate the effectiveness and efficiency of TLE, we conduct experiments on synthetic data sets and three handwritten digit data sets. All the experiments are run in MATLAB R2013b, RAM 16G, CPU 3.6 GHz.

3.1 Experiment on Synthetic Data

To evaluate the performance of the proposed TLE, we take experiments on four synthetic data sets: Twin Peaks, Swiss Roll, Swiss Hole and Toroidal Helix. For these data sets, D_0 are set as 0.025, 2, 2 and 0.025 respectively.

Firstly, we generate $N = 10000$ data for each data set and employ TLE on them. The subfigures of Fig. 3 show the visualizations of the 3D original data, the learned networks viewed from two angles and the 2D embedding obtained by TLE respectively. Each $\mathcal{G}$ is represented by the n extracted nodes (small empty circles) and the connections between them. As illustrated in these subfigures, TLE automatically learns a small number of representative nodes to approximate

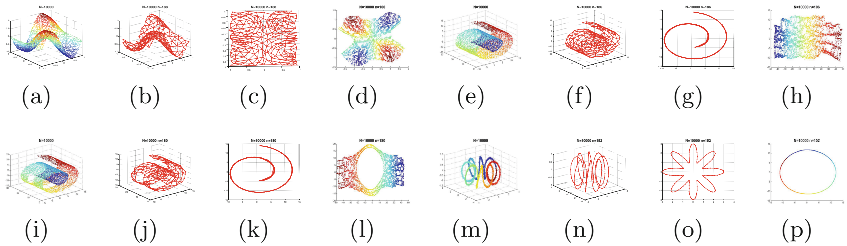


Fig. 3. The results of TLE on synthetic data.

the data's structure well. For comparison, we report the results of Kmeans L-Isomap on synthetic data without noise in Fig. 5(a)–(d). We can find that TLE's performance on these synthetic data is competitive.

The performance of the manifold learning methods are greatly influenced by the quality of the available data. And Isomap-based methods are especially vulnerable to the noise. Therefore, to verify the performance of TLE, we take experiments on Swiss Hole and Toroidal Helix data with uniform distribution noise. The results of TLE are illustrated in Fig. 4. For Swiss Hole data, when $N = 10^4$ and the noise ratio $\frac{N_0}{N} = 5\%$, TLE preserves the basic topology of the manifold well (Fig. 4(a)–(d)). Though when the noise ratio increases to 10%, the disastrous short-circuit errors occur and the obtained 2D embedding is distorted (Fig. 4(e)–(h)), the problem caused by noise is greatly alleviated as N increases to 10^5: the short-circuit errors no longer seriously influence the algorithm's performance (Fig. 4(i)–(l)). For Toroidal Helix data, we find when $N = 10^4$, the 10% noise ratio has little influence to TLE's performance (Fig. 4(m)–(p)). Figure 5(e)–(f) show Kmeans L-Isomap's results on Swiss Hole and Toroidal Helix data with 5% noise. Obviously, Kmeans L-Isomap suffers seriously from the noise and the obtained 2D embedding is greatly distorted. According to the above results, we believe the density-based denoising strategy employed by TLE actually works in these cases.

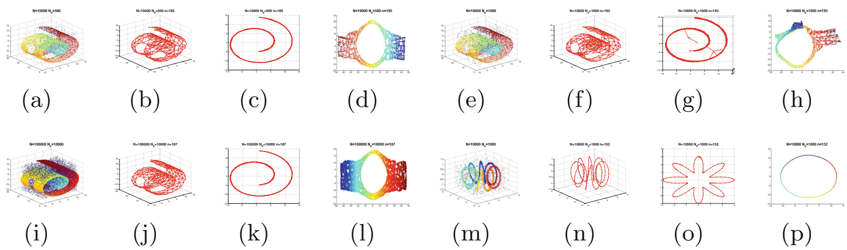
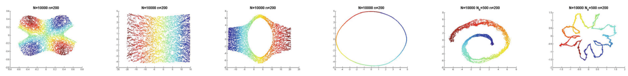


Fig. 4. The results of TLE on Swiss Hole and Toroidal Helix data with noise.



(a) (b) (c) (d) (e) (f)

Fig. 5. The results of Kmeans L-Isomap on synthetic data.

3.2 Experiment on Handwritten Digit Data

In this section, we take experiments on real-world handwritten digit datasets MNIST, optdigits and USPS. In these three datasets, the dimensions of the original data are 400, 64 and 256 respectively; the sizes of training sets are 60000, 3823 and 7291 respectively; the sizes of testing sets are 10000, 1797 and 2007 respectively. In all the experiments, the target dimension is 5, parameter D_0 is set as 100, 1000 and 100 for these three datasets respectively. The quality of the obtained data's low-dimensional representations is validated by performing classification with 1-nearest neighbor classifier.

We employ TLE on the training set to obtain network $\mathcal{G}$, then map the training and testing data into the 5-dimensional space. After that, classification is run. In Table 1, we report the following results of TLE: number of adaptively generated nodes (n), the execution time on the training set (time) and the misclassification rate (MCR). These results are the averages of 10-times executions with different training data input order. As the classical Isomap cannot achieve out-of-sample DR and the Isomap-based methods' storage and computational cost is huge, in Table 1, we report the results of TLOE [7] for comparison and employ PCA's MCR as the benchmark. For TLOE, the initial threshold is the Euclidean distance between the first and the second input data, the other parameters are set as follows: $t_0 = 500$, $e = 0.25$, $age = 100$.

Table 1. Classification results on handwritten digit data sets

Dataset	TLE			TLOE			PCA
	n	Time	MCR	n	Time	MCR	MCR
MNIST	498.8 ± 6.7	135.01s	**10.17%**	503.9 ± 21.1	201.82s	12.85%	30.28%
optdigits	184.4 ± 8.1	1.30s	**4.46%**	440.9 ± 72.5	9.61s	5.41%	12.08%
USPS	304.7 ± 8.1	3.9s	**10.50%**	423.5 ± 65.4	9.99s	11.94%	21.87%

Handwritten digit data are believed to have potential nonlinear structure. As on these data sets, the MCR of PCA is obviously higher than that of TLE and TLOE, we deem that TLE and TLOE are much better at learning these data's structure information. Compared with TLE, TLOE performs a lot more node generation operations during the learning process. To prevent the network from growing too fast, the denoising operations should be performed more frequently. Moreover, the output of TLOE is less stable. From Table 1, we can find that as an improved method, TLE's MCR is lower than that of TLOE when it generates less nodes. High complexity is one of the biggest drawbacks of nonlinear DR algorithms. As reported in Table 1, TLE is able to process 60000 400-dimensional data in minutes. Therefore, as a nonlinear DR method, TLE has good efficiency and its performance is better than that of TLOE.

Then, we compare TLE with L-Isomap and Isomap methods in Table 2. Random L-Isomap, Maxmin L-Isomap and Kmeans L-Isomap are three kinds of L-Isomap methods select their landmarks by randomly selecting, Maxmin algorithm and Kmeans algorithm respectively. Consider the calculation complexity, for MNIST and USPS data, the experiments are taken on a subset size of 5000 from the training set. We run 10-fold cross validation 10 times to obtain the misclassification rate (cvMCR) as evaluation criteria. Each time, n the node number automatically determined by TLE, and the same n is employed as the landmark number of L-Isomap methods. For the L-Isomap and Isomap, neighbor determination is achieved through finding K-nearest neighbor. We execute these methods with setting $K = 5,\ 10,\ 15$ and 20, and only report the optimal results.

Table 2. Comparison between TLE and Isomap-based methods

Data set		TLE	Random L-Isomap	Maxmin L-Isomap	Kmeans L-Isomap	Isomap
MNIST $n = 293.7$	K	-	5	5	5	5
	time	**6.99**s	71.16s	75.71s	143.07s	1175.3s
	cvMCR	15.98%	16.88%	16.34%	**14.36**%	16.94%
optdigits $n = 187.6$	K	-	5	5	5	5
	time	**1.16**s	28.80s	29.71s	34.83s	571.32s
	cvMCR	3.32%	2.67%	**2.45**%	2.52%	2.64%
USPS $n = 270.0$	K	-	5	5	5	5
	time	**3.10**s	65.40s	69.29s	107.83s	1185.01s
	cvMCR	7.70%	5.36%	**5.07**%	5.45%	5.36%

According to Table 2, the execution time of TLE is much less than that of L-Isomap and Isomap methods, while TLE's cvMCR is only a little worse than that of those methods. In all cases, L-Isomap and Isomap methods achieve their best performance when $K = 5$. We guess that for these methods, smaller K means smaller neighborhood for each node and the more precise geodesic distances can be obtained. Thus, the accuracy manifold structure leads to a good performance. Inevitably, as TLE enjoys low computational and space complexity cost, the topology structure learned by TLE should suffer loss. However, the cvMCR results imply the quality of the topology structure learned by TLE is still not bad. Therefore, by combing the advantages of low complexity, adaptive neighborhood determination and out-of-sample data embedding, we may declare the proposed TLE is a competitive manifold learning method.

4 Conclusion

In this paper, we propose a novel method named TLE to achieve fast and incremental manifold learning for large scale data. We admit that TLE is not perfect, there are many problems need studying: for example, the setting of the important parameters such as p and D_0, the convergence of the algorithm, even maybe there are other technologies perform much better at topology structure learning. However, we believe the basic idea of TLE is meaningful: the goal of manifold learning can be achieved in a less costly manner.

Acknowledgements. This work is supported in part by the National Science Foundation of China under Grant Nos. (61373130, 61375064, 61373001), and Jiangsu NSF grant (BK20141319).

References

1. Tenenbaum, J.B., Silva, V.D., Langford, J.C.: A Global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
2. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, vol. 14, pp. 586–691. MIT Press (2001)
4. He, X.F., Niyogi, P.: Locality preserving projections. Adv. Neural Inf. Process. Syst. **45**(1), 186–197 (2005)
5. Wang, J.Z.: Local tangent space alignment. In: Geometric Structure of High-Dimensional Data and Dimensionality Reduction, pp. 211–234. Springer, Berlin, Heidelberg (2012). doi:10.1007/978-3-642-27497-8_11
6. Bellman, R.: Adaptative Control Processes: A Guided Tour. Princeton University, Princeton (1961)
7. Gan, Q., Shen, F.R., Zhao, J.X.: Improved Manifold Learning with competitive Hebbian rule. In: International Joint Conference on Neural Networks 2015, pp. 1–6 (2015)
8. Shen, F.R., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Networks **19**(1), 90–106 (2006)
9. Xing, Y.L., Shi, X.F., Shen, F.R., Zhou, K., Zhao, J.X.: A self-organizing incremental neural network based on local distribution learning. Neural Networks **84**, 143–160 (2016)
10. Martinetz, T., Schulten, K.: Topology Representing Networks. Neural Networks **7**(3), 507–522 (1994)
11. Silva, V.D., Tenenbaum, J.B.: Sparse Multidimensional Scaling using Landmark Points (2004)
12. Bengio, Y., Paiement, J.F., Vincent, P., Delalleau, O., Roux, N.L., Ouimet, M.: Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. Adv. Neural Inf. Process. Syst. **16**, 177–184 (2004)