A More Topologically Stable Locally Linear Embedding Algorithm Based on R*-Tree*

Tian Xia[1,2], Jintao Li[1], Yongdong Zhang[1], and Sheng Tang[1]

[1] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
[2] Graduate University of the Chinese Academy of Sciences, Beijing 100039, China
{txia,jtli,zhyd,ts}@ict.ac.cn

Abstract. Locally linear embedding is a popular manifold learning algorithm for nonlinear dimensionality reduction. However, the success of LLE depends greatly on an input parameter - neighborhood size, and it is still an open problem how to find the optimal value for it. This paper focuses on this parameter, proposes that it should be self-tuning according to local density not a uniform value for all the data as LLE does, and presents a new variant algorithm of LLE, which can effectively prune "short circuit" edges by performing spatial search on the R*-Tree built on the dataset. This pruning leads the original fixed neighborhood size to be a self-tuning value, thus makes our algorithm have more topologically stableness than LLE does. The experiments prove that our idea and method are correct.

Keywords: LLE, Manifold Learning, Nonlinear Dimensionality Reduction, R*-Tree, Neighborhood size.

1 Introduction

Dimensionality reduction is introduced as a way to overcome the curse of dimensionality when dealing with high-dimensional data and as a modeling tool for such data. There are usually two kinds of methods for dimensionality reduction: linear and nonlinear methods. Linear subspace methods are the most popular linear methods, including PCA (Principle Component Analysis), FLA (Fisher Linear Analysis), and ICA (Independent Component Analysis). However, we will only concentrate on the nonlinear methods in this paper because these methods pay more attention to nonlinearity in the dataset, and nonlinearity is more universal than linearity in the real world.

Recently, some manifold learning methods have been proposed to perform nonlinear dimensionality reduction, including LLE (Locally Linear Embedding) [1][8], ISOMAP

* This work was supported in part by the National Basic Research Program of China (973 Program, 2007CB311100), the National High Technology and Research Development Program of China (863 Program, 2007AA01Z416), the Knowledge Innovation Project of The Institute of Computing Technology, Chinese Academy of Sciences (20076031).

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 803–812, 2008.

[2], and Eigenmaps [3]. All the above nonlinear algorithms share the same framework, consisting of three steps:

1. Constructing a K nearest neighborhood graph over the dataset.
2. Constructing a "normalized" matrix M.
3. Calculating spectral embedding based on the eigenvectors of M.

The neighborhood size K which has to be specified manually in the first step plays an important role in constructing a reasonable neighborhood graph for nonlinear dimensionality reduction. A large value of K tends to introduce "short circuit" edges [4] into the neighborhood graph, while a too small one may lead to an unconnected graph, both cases will distort the results of nonlinear dimensionality reduction. [5] and [6] tried to select the optimal value for K automatically based on a predefined cost function. Two main issues arise in this approach. First, it is an enumeration method in fact, and very time consuming. Second, the cost function is very difficult to define, and we do not think the cost functions used in [5] and [6] are reasonable and effective. We will give our explanation and proof in Section 3 and Section 5.

In this paper, we focus on the selection of neighborhood size in manifold learning methods for nonlinear dimensionality reduction, and concentrate on LLE without loss of generality. We propose that K should be self-tuning according to local density not a uniform value for all the data, and present a new variant algorithm of LLE, which can effectively prune "short circuit" edges by performing spatial search on the R*-Tree [7] built on the dataset. This pruning leads the original fixed neighborhood size to be a self-tuning value, thus makes our algorithm have more topologically stableness than LLE does.

2 Background

2.1 LLE Algorithm

Locally Linear Embedding [1][8] tries to find meaningful low-dimensional structure hidden in high-dimensional data. It maps a dataset $X = \{\vec{X}_1,...,\vec{X}_N\}$, $\vec{X}_i \in \mathbb{R}^D$, to a data set $Y = \{\vec{Y}_1,...,\vec{Y}_N\}$, $\vec{Y}_i \in \mathbb{R}^d$, where $d \ll D$. Formally the algorithm consists of three steps:

Step 1. For each data point $\vec{X}_i$, find its K nearest neighbors set $NE_i = \{\vec{X}_j \mid j \in J_i, |J_i| = K\}$.

Step 2. Compute the weights W_{ij} that best reconstruct each data point $\vec{X}_i$ from its K nearest neighbors by solving a least squares problem as follows.

$$\min\ \varepsilon(W_{N\times N}) = \sum_{i=1}^{N} \left| \vec{X}_i - \sum_{j\in J_i} W_{ij}\vec{X}_j \right|^2 \text{, s.t. } \sum_{j\in J_i} W_{ij} = 1,\ W_{ij} = 0 \text{ if } j \notin J_i . \quad (1)$$

Step 3. For each data point $\vec{X}_i$, compute the vector $\vec{Y}_i$ which best fits the reconstruction weights by solving the optimization problem as follows.

$$\min\ \Phi(Y_{d\times N}) = \sum_{i=1}^{N}\left|\vec{Y}_i - \sum_{j\in J_i} W_{ij}\vec{Y}_j\right|^2 \text{, s.t. } \sum_{i=1}^{N}\vec{Y}_i = \vec{0},\ \frac{1}{N}\sum_{i=1}^{N}\vec{Y}_i\vec{Y}_i^T = I. \quad (2)$$

This optimization problem can be solved by calculating the non-zero bottom d eigenvectors of matrix $M = (I-W)^T(I-W)$, these eigenvectors form rows of matrix Y. We also list two properties the matrix M satisfies as follows, and these properties will help us to understand some results of LLE in Section 3.

1. M is symmetric and positive semi-definite.
2. M has N non-negative, real-valued eigenvalues $\lambda_N \geq ... \geq \lambda_2 \geq \lambda_1 = 0$, and the corresponding eigenvector to 0 is the constant one vector $\vec{1}$.

2.2 R*-Tree

Since 1984 when Antonin Guttman first proposed R-Tree, it has become one of the most popular spatial index mechanisms which can help retrieve spatial data more efficiently according to their spatial locations. During these years, researchers and practitioners have applied R-Tree everywhere, from CAD and GIS to Multimedia Information Retrieval, and have made many variations including R+-Tree, R*-Tree, TV-Tree, X-Tree, Pyramid-Tree. Details about R-Tree and its variations can be found in [9].

R*-Tree [7] which is used in our topologically stable LLE algorithm, is also a variant of R-Tree. It deals with the problem as follows. Given a spatial dataset: $S = \{s_1,...,s_n\}$, $s_i \in \mathbb{R}^m$, and a m-dimensional bounding box represented by $I = (I_1, I_2, ..., I_m)$, I_i is a closed bounded interval $[a_i, b_i]$, using what spatial index mechanism can we retrieve the data $R = \{r_i \mid r_i \in S, r_i \text{ locates in } I\}$ quickly and precisely? Based on the R*-Tree built on S, we can do the retrieval very easily. Particularly, we construct I in our algorithm as follows. Given any location $l_k = (l_{k1},...,l_{km}) \in \mathbb{R}^m$ and a positive real-valued range ε, I_i is a closed bounded interval $[l_{ki} - \varepsilon, l_{ki} + \varepsilon]$.

3 Problem Formation and Related Work

As we can see, LLE has one free parameter -K- the number of neighbors used in Step 1. K controls the range of neighbors based on which we reconstruct a data point, and we think the optimal K should satisfy two requirements at the same time:

1. The local linearity is preserved in K nearest neighbors.
2. K is as large as possible.

The first requirement is easy to understand, as for requirement 2, we can see that the larger K is, the more information the neighbors can provide for reconstruction in Step 2. But it is very difficult to select the optimal K satisfying the two requirements manually. A small K can divide a continuous manifold into unconnected sub-manifolds. Fig. 1a-c illustrate the result of LLE performing on modified S-Curve dataset with a small K.

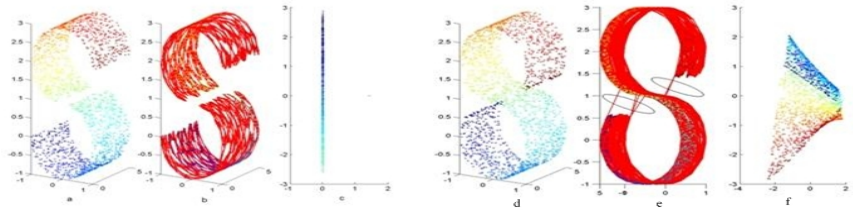

Fig. 1. (a) A broken S-Curve dataset with 2000 points. (b) The corresponding neighborhood graph on the broken S-Curve where K=7. (c) The result of LLE over the broken S-Curve where K=7. (d) A S-Curve dataset with 2000 points. (e) The corresponding neighborhood graph on S-Curve where K=20, the edges circled by a ring are "short circuit" edges. (f) The result of LLE over S-Curve where K=20.

Fig. 1a-c show that LLE performs very badly when K is not large enough to make the entire graph a connected one. We can interpret this situation when considering the properties of matrix M which are mentioned in Section 2.1. Considering M consists of C connected components (C=2 in Fig. 1), without loss of generality, we assume that the data points are ordered according to the connected components they belong to. In this case, M has a block diagonal form: $M = diag(M_1,...,M_C)$. Note that each block M_i has the same properties as M. Thus, we know that M has eigenvalue 0 with multiplicity C, and the corresponding eigenvectors are the indicator vectors of the connected components with 1 at the positions of one block and 0 at the positions of the other blocks. LLE assumes the multiplicity of eigenvalue 0 of M is 1, so it performs badly when the multiplicity is lager than 1.

In contrast, a large K tends to violate the requirement 1 through introducing "short circuit" edges [4] into the neighborhood graph. Fig. 1 d-f show the failure caused by "short circuit" edges.

Some work [5][6] focused on choosing the optimal K automatically. They usually choose an interval of possible values of K firstly, then determine the optimal K by calculating the predefined cost function for each candidate in the interval. These methods are computationally demanding as a result of enumerating every candidates of K. And the cost function measuring the quality of input-output mapping is hard to define. [2] proposed to use the residual variance to evaluate the mapping. The residual variance in [2] is defined as $1-\rho^2_{\widehat{D}_M D_Y}$, where D_Y is the matrix of Euclidean distances in the low-dimensional embedding, $\widehat{D}_M$ is a best estimate of the intrinsic manifold distances D_M, and ρ is the standard linear correlation coefficient. But how to compute $\widehat{D}_M$ which is a good estimate of real manifold distances D_M, especially for data in real world, is a very hard problem. [5] and [6] used $K_{opt} = \arg\min_K(1-\rho^2_{D_X D_Y})$ to specify the optimal K. where D_X is the matrix of Euclidean distances in the high-dimensional input space. It is obviously wrong to take D_X as an estimate of D_M, it even doesn't work on S-Curve and Swiss-Roll datasets, we will demonstrate this by experimental results in Section 5.

In fact, we can calculate the exact D_M on some artificial datasets, such as S-Curve and Swiss-Roll. We ignore the dimensionality of height, and only discuss the computation of arc length on two-dimensional S-Curve and Swiss-Roll. The two dimensional S-Curve consists of connected two pieces of circular curves with opposite rotation orientations, the arc length between two points with angles θ_1 and θ_2 (measured in radians) is simply: $s = r * |\theta_1 - \theta_2|$, where r is the radius.

In polar coordinates, two-dimensional Swiss-Roll is defined as $r = \theta$. So the arc length is:

$$s = \int_{\theta_1}^{\theta_2} \sqrt{r^2 + \left(\frac{dr}{d\theta}\right)^2}\, d\theta = \frac{\theta}{2}\sqrt{\theta^2+1} + \frac{1}{2}\ln\left|\theta + \sqrt{\theta^2+1}\right|\Bigg|_{\theta_1}^{\theta_2} \tag{3}$$

We use $1 - \rho^2_{D_M D_Y}$ to evaluate LLE over artificial datasets in the following sections.

4 Our Solution Based on R*-Tree

As we can see in Section 3, it is not only difficult to specify the optimal K automatically, but also unreasonable to assume that each data point share the same number of neighbors. In fact, LLE ignores the effect of local scaling [11], when the dataset includes subsets with different statistics there may not be a single value of K that works well for all the data.

Actually, every data point should have its own optimal K, not a uniform value. The individual optimal K should subject to the two constraints proposed in Section 3, it means that for each data point, K should be as large as possible without introducing the "short circuit" edges into the neighborhood graph. Investigating into the "short circuit" edge, we can find that it usually passes by a low-density area in which very sparse data points are located. We can detect and prune the "short circuit" edges based on this property. [10] proposed the pruned-ISOMAP algorithm, which first constructs an neighborhood graph with a relative large K, then prunes "short circuit" edges existed possibly in the graph based on kernel density estimation. But the pruning based on kernel density estimation in [10] is computationally demanding, and needs to specify manually two elaborate parameters which depict the kernel function. In fact, if we just want to prune "short circuit" edge, we need not to know the exact local density at the edge, the computation of which is very time consuming. Based on a spatial index built on the dataset, we can do the pruning more efficiently. In this paper we use R*-Tree to do the pruning.

4.1 "Short Circuit" Edge Pruning Based on R*-Tree

As we discussed in Section 2.2, given a data point and a range, R*-Tree helps us retrieve its neighbors efficiently. So we can detect "short circuit" edge by counting the number of its neighbors based on R*-Tree. As for an edge $E(X_i, X_j)$ connecting two data points X_i and X_j, and $X_i, X_j \in \mathbb{R}^n$. We sample the bisecting point on the edge for the consideration of computing efficiency: $X_{ij} = (X_i + X_j)/2$. For the sample point

$X_{ij} = (x_1, ..., x_n)$, we perform a R*-Tree search for the neighbors of X_{ij} in a n-dimensional bounding box represented by $I = (I_1, I_2, ..., I_n)$, where I_m is a closed bounded interval $[x_m - \varepsilon_{ij}, x_m + \varepsilon_{ij}]$. ε_{ij} is an important value that will be discussed later in this section. Now we denote the number of neighbors retrieved by the R*-Tree search as C_{ij} , $C_{ij} \in \{0,1,2,...\}$, and we define a metric $E_{ij} = C_{ij}$ on the edge $E(X_i, X_j)$. A small E_{ij} means there are sparse data points located in the neighborhood of the edge, and it is very likely to be a "short circuit" edge, while a large E_{ij} means the opposite. E_{ij} is a simple and effective metric which helps us prune the "short circuit" edges simply by setting a threshold on it. We set 0 as the threshold in our algorithm, it means edge $E(X_i, X_j)$ is recognized as a "short circuit" edge if E_{ij} equals 0, a normal edge if E_{ij} is larger than 0. Fig. 2 illustrates the histograms of E_{ij} on two neighborhood graphs, and shows that it is suitable for us to set the threshold to be 0.

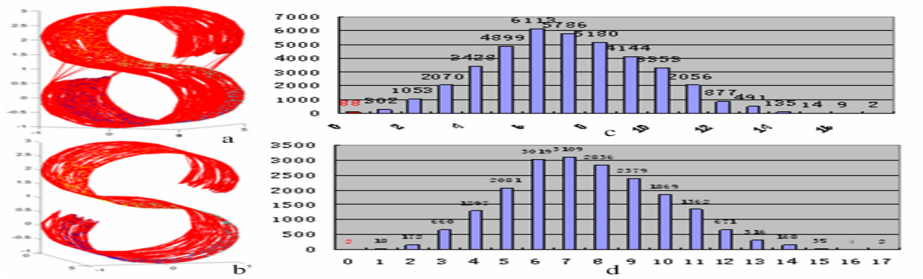

Fig. 2. (a) Neighborhood graph on S-Curve with N=2000, K=20. (b) Neighborhood graph on S-Curve with N=2000, K=10. (c) Histogram of E_{ij} on neighborhood graph in a. (d) Histogram of E_{ij} on neighborhood graph in b.

As for ε_{ij} which confines the searching range, it should deal with the effect of local scaling [11], otherwise it tends to recognize the edges in large scale as "short circuit" edges. It means that ε_{ij} should be self-tuning according to the local scales, dense distribution fits to a small ε_{ij} , while sparse distribution fits to a large one. We define ε_{ij} for edge $E(X_i, X_j)$ as follows.

$$\varepsilon_{ij} = \min(S(X_i), S(X_j)), \quad S(X_i) = \frac{1}{n} \sum_{X_k \in NE_n(X_i)} \left\| X_i - X_k \right\| \tag{4}$$

Where $NE_n(X_i)$ is the n nearest neighbors set of X_i , and n=2 in our algorithm.

4.2 The More Topologically Stable LLE Algorithm

In summary, we propose a more topologically stable LLE algorithm as follows.

1. Choose a relative large K with which the neighborhood graph is connected at least (1%-5% of total number of data points is recommended). Then construct a K nearest neighborhood graph on the dataset based on Euclidean metric.
2. Prune the "short circuit" edges.
   2.1. Create an R*-Tree on the dataset.
   2.2. For every edge $E(X_i, X_j)$ in the K nearest neighborhood graph.
      2.2.1 Specify the bisecting point X_{ij}.
      2.2.2 Calculate the searching range ε_{ij} in Eq. (4).
      2.2.3 Compute E_{ij} by performing a R*-Tree search for the neighbors of X_{ij} within ε_{ij}.
      2.2.4 Prune the edge if its E_{ij} equals 0.
3. Run LLE on the pruned neighborhood graph.

5 Experimental Results

In this section, we present several examples to illustrate the performance of our algorithm that we name as R*-Tree LLE for brevity. We give both subjective and objective results: visualization of output data and residual variance metric which is discussed in Section 3. The test datasets include S-Curve and Swiss-Roll [8].

First, we compare R*-Tree LLE to LLE on uniformly sampled S-Curve and Swiss-Roll under different neighborhood sizes to illustrate the topologically stableness of our algorithm. Fig. 3 and Fig. 6a illustrate the comparison on 2000-point uniformly sampled S-Curve. Fig. 4 and Fig. 6b illustrate the comparison on 2000-point uniformly sampled Swiss-Roll. The residual variance in Fig. 6 is obtained by Eq. (5).

$$\mathrm{RV\_Y1M} = 1 - \rho^2_{D_M D_{Y1}} \quad \mathrm{RV\_Y2M} = 1 - \rho^2_{D_M D_{Y2}} \quad \mathrm{RV\_Y1X} = 1 - \rho^2_{D_X D_{Y1}} \tag{5}$$

Where D_M is the matrix of manifold distances which is discussed in Section 3, D_{Y1} and D_{Y2} are the matrices of Euclidean distances in the output low-dimensional embeddings of LLE and R*-Tree LLE separately. D_X is the matrix of Euclidean distances in the input high-dimensional space. From these figures, we can see R*-Tree LLE has more reasonable visual results and lower residual variances than LLE does, especially when the neighborhood size is large. But we also find that the performances of the two algorithms are similar when the neighborhood size is small. That is because the 2000-point S-Curve and Swiss-Roll are sampled uniformly from two smooth manifolds, thus each data point has nearly the same optimal neighborhood size as it has nearly the same local linearity in its neighborhood. Meanwhile, it is not likely to introduce the "short circuit" edges into the neighborhood graph when the neighborhood size is small. In Fig. 6a, we also show the residual variance RV_Y1X used in [5] and [6], it is obvious to see RV_Y1X is an improper evaluation metric for LLE as we know in Section 3.

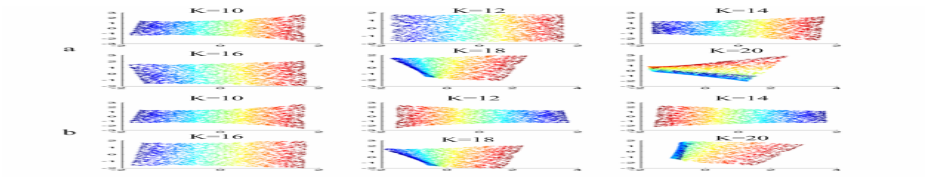

Fig. 3. (a) Results of LLE on 2000-point uniformly sampled S-Curve under different neighborhood sizes. (b) Results of R*-Tree LLE on 2000-point uniformly sampled S-Curve under different neighborhood sizes.

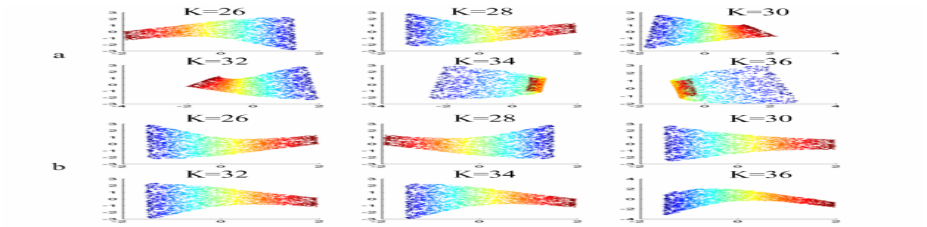

Fig. 4. (a) Results of LLE on 2000-point uniformly sampled Swiss-Roll under different neighborhood sizes. (b) Results of R*-Tree LLE on 2000-point uniformly sampled Swiss-Roll under different neighborhood sizes.

Then, we give the following example to illustrate a self-tuning neighborhood size used in R*-Tree LLE is more reasonable than a uniform neighborhood size used in LLE. We generate the data points from part of Swiss-Roll with a missing rectangle strip, as Fig. 5a illustrates. So the resulting modified Swiss-Roll is not sampled uniformly, and has different densities at different locations. In this case, a self-tuning neighborhood size has more superiorities over a uniform one. From Fig. 5 and Fig. 6c, we can see R*-Tree LLE performs much better than LLE at all the candidate neighborhood sizes, including at the optimal value for LLE.

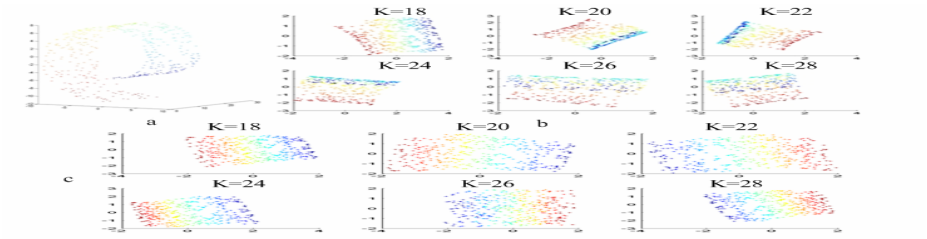

Fig. 5. (a) 450-point non-uniformly sampled Swiss-Roll. (b) Results of LLE on 450-point non-uniformly sampled Swiss-Roll under different neighborhood sizes. (c) Results of R*-Tree LLE on 450-point non-uniformly sampled Swiss-Roll under different neighborhood sizes.

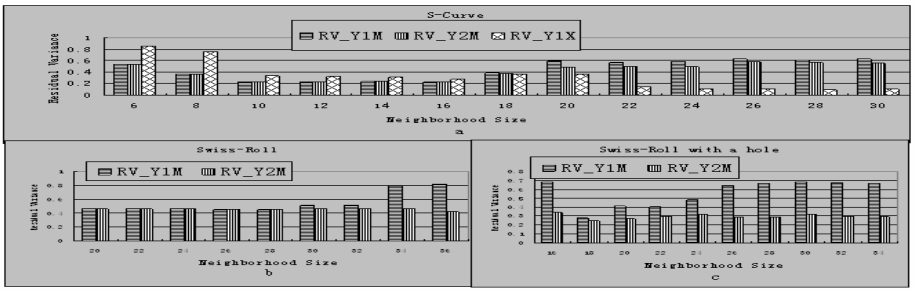

Fig. 6. (a) Residual variances on 2000-point uniformly sampled S-Curve under different neighborhood sizes. (b) Residual variances on 2000-point uniformly sampled Swiss-Roll under different neighborhood sizes. (c) Residual variances on 450-point non-uniformly sampled Swiss-Roll under different neighborhood sizes.

Finally, we give the execution time of the R*-Tree operations in our algorithm which is the necessary time price we pay for higher performance compared to LLE. Implemented in C++ on a personal computer with Pentium-4 3.40 GHz CPU, 1 GB Memory, and MS Windows XP Professional SP2, it takes around 250 ms to create an R*-Tree over 2000 data points, and 0.4 ms to perform a search operation over 2000-point R*-Tree.

6 Conclusion

In this paper, we explore the selection of neighborhood size in LLE, and propose that the neighborhood size should be self tuning according to the local density. Based on this idea, we propose a new variant of LLE which use self tuning K through pruning "short circuit" edges based on R*-Tree.

There are, however, some open problems. In our algorithm, we test every edge while do pruning, in future work, we plan to accelerate the pruning process by a two-step pruning, the first step pruning is based on global information, and the second one depends on local density. The first step is very time efficient, and largely reduces the edges for the second step pruning. We also plan to use other spatial indices, such as TV-Tree and X-Tree, to substitute for R*-Tree, these indices outperforms R*-Tree in indexing high dimensional data. Finally we plan to extend our method to spectral clustering which also needs to construct a neighborhood graph in its algorithm.

References

1. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326 (2000)
2. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)

3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15(6), 1373–1396 (2003)
4. Balasubramanian, M., Schwartz, E.L., Tenenbaum, J.B., de Silva, V., Langford, J.C.: The ISOMAP algorithm and topological stability. Science 295, 5552.7a (2002)
5. Samko, O., Marshall, A.D., Rosin, P.L.: Selection of the optimal parameter value for the ISOMAP algorithm. Pattern Recognition Letters 27, 968–979 (2006)
6. Kouropteva, O., Okun, O., Pietikainen, M.: Selection of the optimal parameter value for the Locally Linear Embedding algorithm. In: Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery, pp. 359–363 (2002)
7. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-Tree: an efficient and robust access method for points and rectangles. In: Proceeding ACM SIGMOD International Conference on Management of Data, NJ, USA, pp. 322–331 (1990)
8. http://www.cs.toronto.edu/~roweis/lle/
9. http://www.rtreeportal.org/
10. Chao, S., Hou-kun, H., Lian-wei, Z.: A more topologically stable ISOMAP algorithm. Journal of Software 18(4), 869–877 (2007)
11. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. Advances in Neural Information Processing Systems 17, 1601–1608 (2005)