Information Processing Letters 115 (2015) 492–501

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

An improved incremental nonlinear dimensionality reduction for isometric data embedding ☆

Xiaofang Gao [a,*], Jiye Liang [a,b]

[a] College of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi 030006, China
[b] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Taiyuan, Shanxi 030006, China

ARTICLE INFO

Article history:
Received 10 March 2014
Received in revised form 16 November 2014
Accepted 8 December 2014
Available online 16 December 2014
Communicated by X. Wu

Keywords:
Manifold learning
Nonlinear dimensionality reduction
Incremental learning
ISOMAP
Design of algorithms

ABSTRACT

Manifold learning has become a hot issue in the field of machine learning and data mining. There are some algorithms proposed to extract the intrinsic characteristics of different type of high-dimensional data by performing nonlinear dimensionality reduction, such as ISOMAP, LLE and so on. Most of these algorithms operate in a batch mode and cannot be effectively applied when data are collected sequentially. In this paper, we proposed a new incremental version of ISOMAP which can use the previous computation results as much as possible and effectively update the low dimensional representation of data points as many new samples are accumulated. Experimental results on synthetic data as well as real world images demonstrate that our approaches can construct an accurate low-dimensional representation of the data in an efficient manner.

1. Introduction

Data from the real world is of a high-dimensional nature, and so it is very difficult to understand and analyze. Some linear dimensionality reduction techniques attempted to solve this problem by lowering the data dimensionality, such as PCA [8], MDS [18]. But they only can solve linear data. Since 2000, manifold learning has become a hot issue in the field of machine learning and data mining. Its main goal is to find a smooth low-dimensional manifold embedded in nonlinear high-dimensional data space. There are some algorithms proposed to extract the intrinsic characteristics of different types of high-dimensional data, such as ISOMAP [19], LLE [17] and so on. These algorithms aim to presever different geometrical properties of the data manifold, and formally transform the dimensionality reduction problem into an eigen-problem of matrices. Therefore, they are often mentioned as spectral embedding methods [21].

Most of these manifold learning algorithms operate in a batch mode, meaning that they have no incremental ability and all data points are need to be available during training [12]. However, in applications like video surveillance, and speech recognition, where data come sequentially, the batch methods seem clumsy: running them repeatedly is not only time consuming, but also wasteful to discard previous results [5]. So it is urgently necessary to develop incremental methods to efficiently find intrinsic properties of high-dimensional data. As more and more data points are obtained, the evolution of data manifold can reveal interesting properties of the data stream [12].

There have been some attempts to create incremental manifold algorithms, which can be roughly categorized into two groups. One group, known as out-of-sample

☆ This work was supported by the National Natural Science Foundation of China (Nos. 61303091, 61202018 and 61201453), Specialized Research Fund for the Doctoral Program of Higher Education (Nos. 20131401120004), the Natural Science Foundation of Shanxi (Nos. 2013021018-2 and 2012011014-4).

* Corresponding author.
E-mail addresses: gxfhtp@sxu.edu.cn (X. Gao), ljy@sxu.edu.cn (J. Liang).

http://dx.doi.org/10.1016/j.ipl.2014.12.004
0020-0190/

extension, attempts to parameterize new observations based on the assumption that all the known results are correct. Out-of-sample extensions for LLE, ISOMAP, LE are given by Bengio et al. [2], using kernel tricks to reformulate these algorithms. But the method may fail if the data manifold is non-uniformly sampled [16]. The another group tries to give more credible results, not only embedding new points but also updating the known results, such as incremental LLE [16], incremental Isomap [12], incremental LTSA [11], incremental LE [6], etc. All the recent methods in the latter group are restricted to dealing with only one new point per running, and thus they are forced to rerun as many times as the number of new data points. The total cost of the time complexity and memory requirement is high and even higher than those of re-running the original algorithms. As a further imperfection, the geometric structure of the manifold may be destroyed if the new sample does not lie in original sampled area.

In this paper, an improved incremental version of ISOMAP is proposed, which can use the previous computation results as much as possible and effectively update the low dimensional representation of data points as many new samples are collected simultaneously. The algorithm not only is more fit to the cognitive mechanism in our brain, but also improves the efficiency while the accuracy of the embedding results are not be decreased obviously. The experimental results on both synthetic "Swiss-roll" data set and two real images data sets show that the algorithm is feasible.

The corresponding works (main contributions) of our approaches include

- An effective method to update the neighborhood graph and geodesic distances matrix. Different from ISOMAP [19] and its incremental versions [12], the method does not re-compute and update k-NN neighborhood graph. It keeps the previous neighborhood relations as much as possible, only adds the new neighborhood relations related to some new points and deletes the original links leading to short circuits. And the method also does not update the geodesic distances one by one. It only updates the distances of two kinds of paths: the paths leading to the conflicting predecessor matrix; the paths including short circuits.
- A simple method to detect the short circuits in the neighborhood graph. The method re-estimates all weights of the edges in the original neighborhood graph in view of the newest "geodesic distance" between new points and all points (including all the original points and new points). At the same time, the thresholds of the weights are also estimated by computing the maximum distances of two neighborhood pitches. If its weight is larger than its threshold, the edge can be marked as a short circuits edge.
- A better solution of the incremental eigen-decomposition problem with increasing matrix size, which computes eigen-values and eigenvectors by subspace iteration with Rayleigh–Ritz acceleration. This differs from previous incremental ISOMAP version [12] where only one new sample is increased and its coordinate is directly estimated.

The rest of the paper is organized as follows: Section 2 reviews the related works. Section 3 describes the proposed incremental version of ISOMAP. Section 4 shows the complexity of the proposed algorithm and compares it with those of ISOMAP and law-IISOMAP. Section 5 presents the experimental results and finally Section 6 gives a conclusion.

2. Related works

Suppose that $M \subset R^D$ is a smooth manifold. A set of data points $\{x_1, ..., x_n\}$ is sampled from it. ISOMAP assumes that the data lie on a (Riemannian) manifold and maps x_i to its d-dimensional representation y_i in such a way that the geodesic distance between x_i and x_j is as close to the Euclidean distance between y_i and y_j in R^d as possible.

ISOMAP algorithm has three steps:

i. Constructing the neighborhood graph. ISOMAP requires specifying a parameter of the neighborhood: k-nearest neighbors (k-NN) or ε-hyper sphere. The k-NN version is more common since the sparseness of the resulting structures is guaranteed. The weighted undirected neighborhood graph $NG = (V, E)$ is constructed with the vertex $v_i \in V$ corresponding to x_i. An edge $e(i, j)$ between v_i and v_j exists if x_i is a neighbor of x_j. The weight of $e(i, j)$, denoted by w_{ij}, is the value of the Euclidean distance. If the set of the k-NN neighborhood of x_i is denoted by $knn(i)$ and the set of indices of the vertices adjacent to v_i in G is denoted by $adj(i)$, then $adj(i)$ is corresponding to $knn(i) \bigcup \{v_j | v_i \subset knn(j)\}$.

ii. Estimating the geodesic distances. The key assumption is that the geodesic between two points on the manifold can be approximated by the shortest path between the corresponding vertices in the neighborhood graph. Let g_{ij} denote the length of the shortest path $sp(i, j)$ between v_i and v_j. The shortest paths can be found by the Dijkstra's algorithm with different source vertices. The shortest paths can be stored efficiently by the predecessor matrix Π, where $\pi_{ij} = k$ if v_k is immediately before v_j in $sp(i, j)$. Since g_{ij} is the approximate geodesic distance between x_i and x_j, we shall call g_{ij} the geodesic distance. So the geodesic distance matrix $G = \{g_{ij}\}$ is symmetric.

iii. Recovering the embedding results $\{y_1, ...y_n\}$ by using the classical MDS on the geodesic distances. Let B be the target inner product matrix, i.e., the matrix of the target inner products between different y_i. If restricting $\sum_i y_i = 0$, B is recovered as $B = -HAH/2$, where $a_{ij} = g_{ij}^2$, $H = I_n - J_n/n$, I_n is an identity matrix and J_n is a matrix with $n \times n$ ones. We seek $Y^T Y$ to be as close to B as possible in the least square sense. Then the embedding result $Y = diag(\sqrt{\lambda_1} ... \sqrt{\lambda_d})[u_1 ... u_d]^T$ is achieved, where $\lambda_1, ..., \lambda_d$ are the d largest eigenvalues of B, with corresponding eigenvectors $u_1, ..., u_d$.

3. Incremental ISOMAP

According to the original ISOMAP algorithm, the main works in incremental algorithms involve three steps: up-

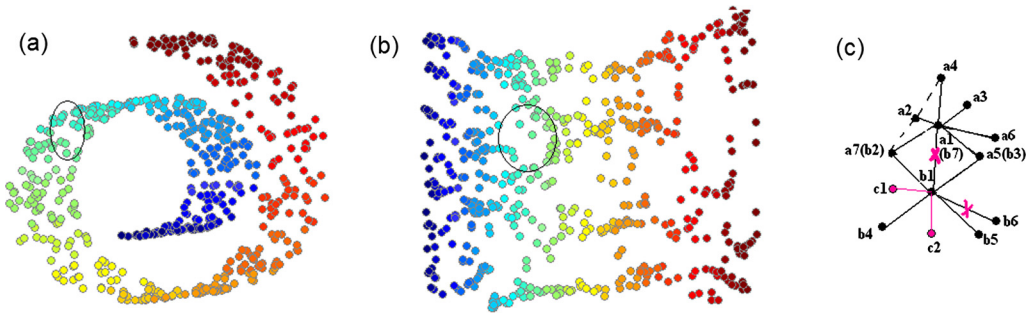

Fig. 1. (a) 600 points sampled randomly from the "Swiss roll"; (b) the 2-dimension embedding results; (c) the neighborhood of a1 and a2. When new samples c1 and c2 are added to be the neighbors of b1, the edges (a1, b1) and (b6, b1) are deleted. The "geodesic distance" between a4 and b4 is enlarged and the accurate is reduces.

dating the neighborhood graph, re-estimating the geodesic distances matrix and solving the incremental eigen-decomposition problem.

Step 1: updating the neighborhood graph.

The construction of the neighborhood is critical step of the manifold learning algorithms, because it can be extended to extract an optimum topological structure from the data by means of Hebbian updates of the neighborhood graph [14]. When new samples $X_1 = \{x_{n+1}, ..., x_{n+m}\}$ arrive, the neighborhood graph is changed because it is influenced in its construction by the parameters of k-NN algorithm and the sampled data. But the topology of manifold would not be changed because it always represents the same manifold.

Suppose that the incremental ISOMAP still use k-NN to update its neighborhood graph. The deleted edges break the existing shortest paths, and the added edges may create the improved shortest paths which should be shorter than the "original shortest paths". This is much more involved than it appears, because the change of a single edge can modify the shortest paths among multiple vertices. The process to update the neighborhood graph is complex. But the accurate of the "geodesic distances" may not be improved. In Fig. 1(c), when new samples c1 and c2 are added to be the neighbors of b1, two neighborhood relationships (a1, b1) and (b6, b1) need to be deleted because of the parameter k. That leads to an enlarged "geodesic distance" between a4 and b4. So the neighborhood size should not be fixed to be a global setting, and the selection of the neighborhood should be data-driven [22]. In fact, the stable parameter k in k-NN is always less than the local optimal neighborhood size, except that the manifold curvature of the point is too high or its sampling density is too low [10]. Some dynamical neighborhood algorithms [22,20, 9,15] have been proposed, in which the neighborhood size of each sample is different. These algorithms enlarge their local neighborhood and the paths between a pair of points in neighborhood graph become more. So the accuracy of the shortest paths and "geodesic distances" are improved.

Our incremental ISOMAP algorithm uses dynamicalKNN algorithm (Algorithm 1) to construct the new neighborhood graph. They are incident to: (1) the new edges are added in the neighborhood graph if and only if they are the newest k nearest neighbors; (2) the original edges are deleted in the neighborhood graph if and only if the topology of the neighborhood graph is destroyed (that is to say, the short circuits[1] occur). In the algorithm, the parameter k is not a fixed parameter any more. It is equal with or greater than k.[2]

The accuracy of "geodesic distances" in incremental ISOMAP algorithm can be improved because the local neighborhood of each sample is enlarged (which is larger than k-NN). The newer k-NN neighbors can be added to original neighborhood graph which is constructed by k-NN algorithm, and the original neighbors do not need to be deleted except that they leading to "short circuits". That is to say, the original neighborhood graph is kept as much as possible. So the efficiency to update the neighborhood graph would be improved greatly.

Algorithm 1 (*dynamicalKNN*). Updating the neighborhood graph.
Input: X, the original dataset; X_1, the set of new samples; adj_{old}, the adjacent relationships matrix; G; Π;
Output: adj_{new}, the new adjacent relationships matrix;
1: computing adj_{new} using k-NN of X and X_1;
2: computing the geodesic distances and predecessor matrix of $x_i \in X_1$ using adj_{new} in Dijkstra's algorithm, then updating G and Π;
3: shortEs := ϕ;
  for $i = 1{:}n$
    shortEs := shortEs $\bigcup$ judgeShortCircuits(i, G, Π, adj_{old}, adj_{new});
  end for;
4: $adj_{new} := adj_{new} \bigcup adj_{old}$; $adj_{new} := adj_{new} - shortEs$;

In dynamicalKNN algorithm, it is simple to add new neighborhood edges in the neighborhood graph while it is difficult to check out the short circuits edges. The main task of the judgeShortCircuits algorithm (Algorithm 2) is to check out the short circuits edges. Firstly, the newest

[1] The short circuits mean that some neighbors of one point come from other different folds so that these neighbors are not the nearest ones on manifold.

[2] In dynamicalKNN algorithm, the newest k-NN of each point are always added to its original neighborhood. Even though some short circuit edges are deleted, the neighbor size of each point is equal with or greater than k.

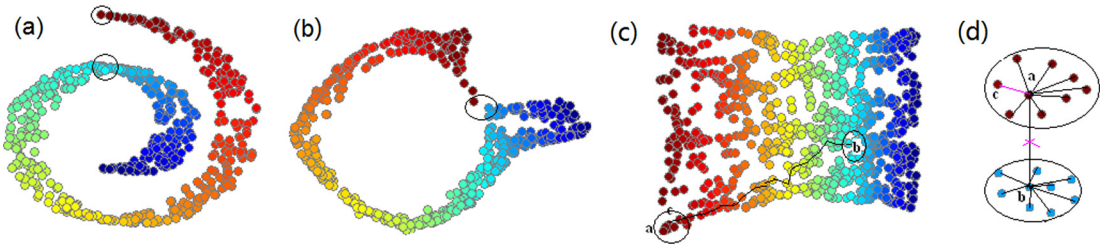

Fig. 2. (a) 600 points sampled randomly from the "Swiss roll"; (b) the 2D embedding results (the short circuits occur); (c) the 2D embedding results after adding 50 new points; (d) the neighborhood of a and b. $e(a,b)$ is short circuit edge. When new sample c is added to the neighborhood of a, the new value of $g_{a,b}$ can be re-estimated to be $g_{a,c}+g_{c,b}$. At the same time, the threshold is estimated to be $\max\{g_{a,adj(a)}\}+g_{a,b}+\max\{g_{b,adj(b)}\}+g_{a,c}$. According to the relationships among the three edges of triangle, if $g_{a,b}$ is larger than its threshold, the edge $e(a,b)$ is judged to be a short circuit edge.

"geodesic distances" between the new points and all points are computed in view of the new k-NN. Secondly, the weights w_{ij} of $e(i,j)$ in the neighborhood graph are re-estimated if and only if the original edge $e(i,j)$ does not exist in the new k-NN. Thirdly, the algorithm computes the threshold of the original neighborhood edges using the newer neighborhood relationships.

In Fig. 2, the point b is the neighbor of the point a and the $e(a,b)$ is a "short circuit" edge. In Fig. 2(c), the new point c is added into the neighborhood of a and its geodesic distances to all points are computed using the newest k-NN. Then the re-estimated geodesic distance $\tilde{g}_{a,b}=g_{a,c}+g_{c,b}$. Although $\tilde{g}_{a,b}>g_{a,b}$, the difference is significant because $g_{a,c}$ is little. And the threshold of $\tilde{g}_{a,b}$ is defined as follow:

$$threshold(g_{a,b})=d_{max}(N_b,N_c)+g_{c,a} \quad \text{and}$$

$$d_{max}(N_b,N_c) = \max\left(w_{a,i}\middle|i\in adj(a)\right)+g_{a,b}+\max\left(w_{b,j}\middle|j\in adj(b)\right).$$

$d_{max}(N_b,N_c)$ represents the maximum distances of two neighborhood pitches of a and b, because c is a neighbor of the point a.

That is to say, the short circuit edges $e(a,b)$ is judged if its newer weight w_{ab} is larger than the threshold.

Algorithm 2 (*judgeShortCircuits*). Finding the short circuit edges.

Input: i, the index of the existing vertex; G; Π; adj_{old}, the original adjacent relationships matrix; adj_{new}, the new k-NN relationships matrix;

Output: scEdges, the set of the short circuit;

```
1: oldN := adj_old(i); newN := adj_new(i);
   delN := oldN − newN; upN := newN − oldN;
2: for k ∈ delN
   newIK := min{dD|dD = g_{i,j} + g_{j,k}, j ∈ newN};
   threshold := max{g_{newN,i}} + max{g_{adj_new(k),k}} + g_{i,k} +
   max{g_{upN,i}};
   if (newIK > threshold)
     scEdges := scEdges ⋃ e(i,k);
     iN := adj_old(i) − {k}; kN := adj_old(k) − {i};
     temp := {e(a,b)|g_{a,b} < g_{a,i} + g_{i,k} + g_{k,b}, a ∈ iN, b ∈ kN};
     for e(a,b) ∈ temp
       if ((a ∉ adj_new(b)) ∧ (b ∉ adj_new(a)) ∧ (π_{a,b} = a) ∧
       (π_{b,a} = b))
         scEdges := scEdges ⋃ e(a,b);
       end if;
     end for;
   end if;
end for;
```

Step 2: re-estimating the geodesic distances.

ISOMAP algorithm uses the shortest path between the corresponding vertices in the neighborhood graph to approximate the geodesic distance between two points on the manifold. After the neighborhood graph is updated, the geodesic distances also need to be updated.

When the manifolds are sampled enough and each sample has enough neighbors, the geodesic distances can be estimated accurately in view of the neighborhood graph. But the sampling is always sparse and each sample has limited neighbors. In other words, no matter which neighborhood graph is used, the estimated geodesic distances are always not accurate. Since the original neighborhood graph and the newer one represent the topologies of same manifold, the deviation between the original geodesic distances and the new ones should be very small, except that the samples are too sparse or the short circuits occur. That is to say, $G_{new}=\left[\begin{smallmatrix} G+\Delta G & G_1 \\ G_1^T & G_2 \end{smallmatrix}\right]$ and $g_{ij}\gg\Delta g_{ij}$, where G_1 is corresponding to the geodesic distances of the new samples to the existing ones and G_2 is the geodesic distances among new samples. As mentioned in Section 2, $B_{new}=-HA_{new}H/2$, $a_{ij}=g_{ij}^2$ and $H=I_n-J_n/n$. If ΔG is ignored,

$$B=-\frac{1}{2}\begin{bmatrix} H_0 & H_1 \\ H_1^T & H_2 \end{bmatrix}\begin{bmatrix} A & A_1 \\ A_1^T & A_2 \end{bmatrix}\begin{bmatrix} H_0 & H_1 \\ H_1^T & H_2 \end{bmatrix} \quad \text{and}$$

$$\Delta B=-\frac{1}{2}\begin{bmatrix} H_0 & H_1 \\ H_1^T & H_2 \end{bmatrix}\begin{bmatrix} \Delta A & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} H_0 & H_1 \\ H_1^T & H_2 \end{bmatrix}.$$

It is obviously that $\Delta b_{ij}\ll b_{ij}$. So ΔG can be ignored and the original "geodesic distances" do not need to be updated one by one except two kinds of paths must be updated: the paths leading to conflicted predecessor matrix; the paths including short circuits.

The information of paths is kept in the predecessor matrix. If there are conflicts in paths, the predecessor matrix cannot remember the right path, the incremental algorithm cannot continue. So the original paths must be updated. And if there are "short circuits" edges in the neighborhood graph, the topological structure of the manifold would be destroyed. So the "short circuits" must be found.

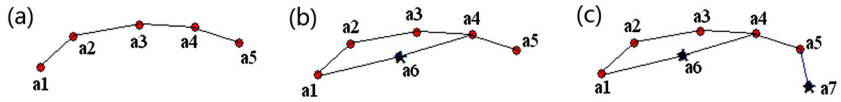

Fig. 3. (a) The shortest paths $sp(a1, a5) = [a1\ a2\ a3\ a4\ a5]$ and $sp(a5, a1) = [a5\ a4\ a3\ a2\ a1]$. (b) After adding new point a6 to the neighborhood graph, the shortest paths are changed to be $sp(a1, a5) = [a1\ a6\ a4\ a5]$ and $sp(a5, a1) = [a5\ a4\ a6\ a1]$. But the original geodesic distances and predecessors are not updated. The predecessor of a4 in $sp(a1, a5)$ is a3. (c) After adding new point a7 to the neighborhood graph, the predecessor matrix produces the conflicting data. The predecessor of a1 in $sp(a7, a1)$ is a6. So $sp(a1, a7) = [a1\ a2\ a3\ a4\ a5\ a7]$ and $sp(a7, a1) = [a7\ a5\ a4\ a6\ a1]$.

Algorithm 3 and Algorithm 4 would solve the two questions.

In general, the conflicts of paths are always related to new samples (see Fig. 3). So the updateConflictPaths algorithm (Algorithm 3) checks and updates the paths including new samples one by one: (1) If the weight of a new path is less than the original geodesic distance, the shortest path should be updated to shorten the geodesic distances; (2) If the new paths include the path such as $\ldots \to v_i \to v_{n+k} \to v_j \ldots$ or $\ldots \to v_i \to v_{n+k1} \to \ldots \to v_{n+k2} \to v_j \to \ldots$, the vertex pair (v_i, v_j) which is adjacent to the new vertices is propagated to the whole updated graph for checking and updating their shortest paths.

Algorithm 3 (*updateConflictPaths*). Updating the geodesic distances matrix and the predecessor matrix.
Input: i, the index of the new vertex; G; Π;
Output: G; Π;
1: $List := \Phi$;
2: for all $s \in adj(i)$
  $\delta_s := \Phi$;
  if s is not a new sample then $\delta_s := \{s\}$; else
    $R := \{s\}$; $P := \{i\}$;
    while $(R \neq \Phi)$
      $a :=$ the first element of R; $N := adj(a)$;
      $P := P \bigcup \{a\}$; $N := N - P$;
      $M := \{k|\ k$ is a new sample, $k \in N\}$;
      $\delta_s := \delta_s \bigcup (N - M)$; $R := (R - \{a\}) \bigcup M$;
    end while;
  end if;
end for;
3: for all $s, t \in adj(i)$
  for all $a \in \delta_s$
    $bList := \{b|b \in \delta_t, g_{ab} > g_{ai} + g_{ib}\}$;
    for all $b \in bList$
      $g_{ab} := g_{ba} := g_{ai} + g_{ib}$; $\pi_{ab} := \pi_{ib}$;
    end for;
    $cList := \{c|c \in \delta_t, \pi_{ac} = \pi_{ic}, \pi_{ca} = \pi_{ia}\}$
    for all $c \in cList$, $List := List \bigcup \{(a, c)\}$; end for;
  end for;
end for;
4: for all $(a, b) \in List$
  $aChild := \{k|\pi_{ki} = \pi_{ai}\}$; $bChild := \{k|\pi_{ki} = \pi_{bi}\}$;
  for all $\{(s, t)|s \in aChild, t \in bChild, g_{st} \neq \infty, g_{st} > g_{si} + g_{it}\}$
    $g_{st} := g_{ts} := g_{si} + g_{it}$; $\pi_{st} := \pi_{it}$; $\pi_{ts} := \pi_{is}$;
  end for;
end for;

The updateShortCircuits algorithm (Algorithm 4) checks and updates the paths including the short circuits which are checked out in Algorithm 2. Suppose that $e(i, j)$ is a short circuit edge. If it is included in the shortest path $sp(s, t)$, the predecessor of v_j in $sp(s, j)$ is v_i and the predecessor of v_i in $sp(t, i)$ is v_j. This fact can be used in turn to find all vertex pairs R_{ij} whose shortest paths are invalidated due to the removal of edge. Then their new shortest paths are computed and the geodesic distances are updated. The process of computing and updating is similar to the Dijkstra's algorithm, except that only a part of the geodesic distances from source vertex v_i to destination vertices (instead of all the vertices) are unknown.

Algorithm 4 (*updateShortCircuits*). Updating the shortest paths including the short circuits.
Input: $\{e(i, j)\}$, the set of the short circuits; G; Π;
Output: G; Π;
1: for all $e(i, j)$
  $iChild := \{k|\pi_{kj} = i\}$; $jChild := \{k|\pi_{ki} = j\}$;
  $R_{ij} := R_{ij} \bigcup \{(s, t)|s \in iChild, t \in jChild, \pi_{st} = \pi_{jt}, \pi_{ts} = \pi_{is}, s \notin adj(t), t \notin adj(s)\}$;
  $g_{ij} := g_{ji} := \infty$; $\pi_{ij} := 0$; $\pi_{ji} := 0$;
 end for;
2: for all x_i, $i = 1, \ldots, n$
  $jSet := \{j|g_{ji} = \infty, j = 1, \ldots, n\}$; $H := \varphi$;
2.1 for all $j \in jSet$
    $tN := adj(i) - jSet$; $\delta_j := \min_{k \in tN}(g_{ik} + g_{kj})$;
    $H := H \bigcup \{\delta_j\}$;
    if $\delta_j \neq \infty$ then
      $k := \arg(\delta_j = g_{ik} + g_{kj})$; $\pi_{ji} := \pi_{ki}$; $\pi_{ji} := \pi_{ki}$;
    end if;
  end for;
2.2 while $H \neq \Phi$ do
    $k := \arg\min_j \{\delta_j, \delta_j \in H\}$;
    $jSet := jSet - \{k\}$; $H := H - \{\delta_k\}$; $g_{ik} := g_{ki} = \delta_k$;
    for all $j \in adj(k) \bigcap jSet$
      $dist := g_{ik} + g_{kj}$;
      if $dist < \delta_j$ then
        $\delta_j := dist$;
        update the value of δ_j in H;
        $\pi_{ij} := k$; $\pi_{ji} := \pi_{ki}$;
      end if;
    end for;
  end while;
end for;

Step 3: constructing the embedding results.

The new embedding results $\{y_1, \ldots, y_{n+m}\}$ should be updated in view of B_{new}. This also can be viewed as an incremental eigen-value problem, as y_i can be obtained by the eigen-decomposition of B_{new}. In order to use the previous computation results as much as possible and effectively get the eigen-values and eigenvectors of B_{new}, we give a good initial guess $\widetilde{U}_{new}$ for dominant vector sub-

space of and use subspace iteration with Rayleigh–Ritz acceleration [4] to find the better eigen-space for it:

- Estimating $\widetilde{U}_{new}$.

According to MDS, the distance matrix G is converted to a symmetric and positive semi-definite matrix $B = -\frac{1}{2}(I_n - \frac{J_n}{n})A(I_n - \frac{J_n}{n}) = Y^T Y$, where $a_{ij} = g_{ij}^2$. Then B can be decomposed into $B = U \sum U^T$, where U is a matrix whose columns are orthonormal eigenvectors and $\sum$ is a diagonal matrix of eigen-values. The d-dimensional coordinate vectors that would give rise to the kernel matrix B are the scaled rows of U. In order to minimize the difference between the embedded and original distances with a fixed number of dimensions d, the eigenvectors with the top d eigen-values are retained. Assuming that the eigenvalues are ordered by decreasing eigen-value, the coordinate vector is $Y = \sum_d^{1/2} U_d^T = diag(\lambda_1^{1/2}, ..., \lambda_d^{1/2})[u_1 ... u_d]^T$. So $U_d = Y^T \sum_d^{-1/2}$.

When new data arrive, the coordinates of the new samples can be estimated by Nyström method [7,1,3]. Suppose that $Y_{new} = [\widetilde{Y}\ Y_1]$ and $\widetilde{Y} \approx Y$, then $B_{new} = \begin{bmatrix} \widetilde{B} & B_1 \\ B_1^T & B_2 \end{bmatrix} = \begin{bmatrix} \widetilde{Y}^T\widetilde{Y} & \widetilde{Y}^T Y_1 \\ Y_1^T \widetilde{Y} & Y_1^T Y_1 \end{bmatrix}$. $Y_1 = (\widetilde{Y}^T)^{-1} B_1 \approx (Y^T)^{-1} B_1 = \sum_d^{-1/2} U_d^T B_1$. Because the topology of the manifold do not changed, the eigen-values of B_{new} change slightly. Suppose that the eigen-values of B_{new} are not changed, the eigenvectors $U_{1d} = {Y_1}^T \sum_d^{-1/2} = B_1^T U_d \sum_d^{-1}$ are decided. In order to restrict $\widetilde{U}_{new}$ to be orthogonal matrix and $\sum Y_{new} = 0$, U_{1d} is centered and Gram–Schmidt orthogonalized, and then the column space of $\widetilde{U}_{new} = \frac{1}{\sqrt{2}}[U_d\ U_{1d}]^T$ is the good initial guess for the subspace of dominant eigenvectors of B_{new}.

- Finding the better eigen-space for B_{new}.

After giving the estimation of $\widetilde{U}_{new}$, we can use subspace iteration with Rayleigh–Ritz acceleration to find the better eigen-space for B_{new}. 1) Compute $Z = B_{new}\widetilde{U}_{new}$ and perform QR decomposition on Z, that is to say, $Z = QR$ and let $V = Q_d$. 2) Form $Z^* = V^T B_{new} V$ and perform eigen-decomposition of the $d \times d$ matrix Z^*. Let λ_i and u_i be the i-th eigen-value and the corresponding eigenvector. Since d is small, the time for eigen-decomposition of Z^* is negligible. 3) $U_{new} = V[u_1 ... u_d]$ is the improved set of eigenvectors of B_{new}, and $[\lambda_1 ... \lambda_d]$ is the corresponding eigen-values.

4. Computational complexity

The computational complexity of the proposed incremental algorithm (it is called our-IISOMAP as follows) is undoubtedly one of the most crucial issues. To clarify this point, its computational efficiency is theoretically analyzed. For comparison, we also take ISOMAP [19] and Law-IISOMAP [12].

First, we present the computational complexity of ISOMAP. It should be recalled that ISOMAP contains three steps: neighborhood graph construction, geodesic distances computation, and embedding results calculation. For a data set with size n, the complexity for constructing a k-NN neighborhood graph is $O(n \log n)$; for computing the geodesic distances, it is $O(kn^2 \log n)$; and for calculating the eigenvalues and eigenvectors of a full $n \times n$ matrix, it is $O(n^3)$. Therefore, the total computational complexity of ISOMAP is $O(n \log n + kn^2 \log n + n^3)$. For comparison, When m new points are accumulated to n original points, the computational complexity of ISOMAP is $O((n+m)\log(n+m) + k(n+m)^2 \log(n+m) + (n+m)^3)$.

In law-IISOMAP algorithm [13], the complexity of updating the neighborhood graph is $O(nq)$. The complexity of updating geodesic distance is $O(q(|F| + |H|))$, where F and H contain vertex pairs whose geodesic distances are lengthened and shortened because of the new point v_{n+1}, respectively. The complexity of computing the new embedding results is n^2 because of the matrix multiplication in the subspace iteration. Since d is small, the time for eigen-decomposition is negligible. Hence, the computational complexity of law-IISOMAP is $O(nq + q(|F| + |H|) + n^2)$. For comparison, when m new points are accumulated to n original points, the computational complexity of law-IISOMAP is $O(\sum_{i=n+1}^{n+m}(iq + q(|F| + |H|) + i^2))$.

For the proposed incremental algorithm (our-IISOMAP), the dynamicalKNN algorithm spends $O(m^2 \log m + nq^3)$ to re-construct the neighborhood graph, where q is the maximum degree of all vertices in neighborhood graph after inserting the new points; In the worst case, the updateConflictPaths and updateShortCircuits algorithm take $O(m(nq + nq^2) + |C|)$ and $O(|S| \log |S| + q|S|)$ time for updating the geodesic distances, where $|C|$ is the number of conflict paths and $|S|$ is the number of short circuits edges. In practice, $|S|$ is generally small, and the overall complexity of updating of geodesic distance can be loosely bounded by $O(mnq^2 + |C|)$. Computing the new embedding results also spends n^2. The total computational complexity of our-IISOMAP is $O(m^2 \log m + nq^3 + mnq^2 + |C| + n^2)$.

Accordingly, ISOMAP algorithm runs in a batch mode and its computational complexity is related to $n + m$. So it cannot be effectively applied when data are collected sequentially. The law-IISOMAP algorithm must run m times iteratively when m new points are accumulated. Because the change of a single edge can modify the shortest paths among multiple vertices, the larger the m is, the larger the $|F| + |H|$ may be. So the law-IISOMAP algorithm may be slowest in three algorithms. The proposed algorithm not only can be applied to the accumulated points, but also its computational complexity is better than that of law-IISOMAP. Therefore, the efficiency of the our-IISOMAP algorithm is evident in the theoretical analyses.

5. Experiments

In order to evaluate the accuracy and efficiency of the proposed approach, ISOMAP [19], law-IISOMAP [12] and our-IISOMAP (the proposed incremental ISOMAP algorithm) are respectively run on several datasets in five sets of experiments. To quantify the accuracy of embedding results update of the incremental algorithms, the residual variance $\delta = 1 - \rho^2_{D_X D_Y}$ is used as an error measure. Here, D_X is the matrices of geodesic distances between pairs of points in input space, D_Y is the matrices of Euclidean

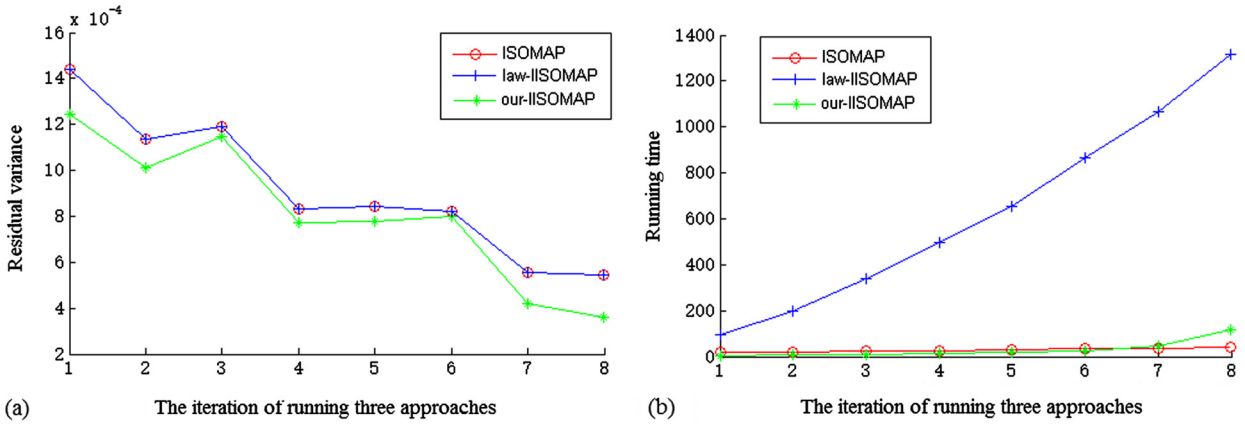

Fig. 4. Comparing the residual variances and running times of ISOMAP, law-IISOMAP and our-IISOMAP running on "Swiss roll" dataset. (a) The residual variances varying with the iteration of running three approaches; (b) the running time varing with the iteration of running three approaches.

distances between pairs of points in output spaces, respectively, and ρ is the standard linear correlation coefficient taken over all entries of D_X and D_Y.

A. The effect of the parameter *m*

According to the analysis of complexity, the running time of the proposed algorithm is related to some factors. The number of original points (n) and the size of neighborhood graph (k) cannot be changed in incremental algorithms. But the number of new points (m) can be controlled. So the experiment is designed to show the effect of the parameter m.

1200 points are sampled randomly from the "Swiss roll". 800 points are fixed to be original points, three algorithms are run 8 times respectively, with m increasing from 50 to 400, the residual variances and the running times are shown in Fig. 4.

In Fig. 4(a), following the increase of m, the residual variances of three algorithms are all decreased gradually. Although the residual variance of our-IISOMAP is better than other two, the difference is not significant. So the accuracy of the incremental algorithm is not sensitive to the parameter m. In Fig. 4(b), the running time of two incremental algorithms are sensitive to m, especially the law-IISOMAP, which validates the theoretical analysis in computational complexity. But when m is increased from 50 to 300, the running time of our-IISOMAP is shorter than those of other two algorithms. That is to say, the proposed algorithm would be effective only if m is limited to a small value (such as $m \ll n$).

B. Experiments on synthetic "Swiss roll" dataset

1000 points are sampled randomly from the "Swiss roll". 600 points are used to be original points and other 400 points are divided into 8 sets to be new data sets. That is to say, n is accumulated from 600 to 1000 by adding $m = 50$ points each time. After respectively running three algorithms 8 times, with a k-nn neighborhood of size 8, five sets of embedding results are shown in Fig. 5, and the residual variances and the running times are shown in Fig. 6.

It is apparent that (1) as more and more data points are accumulated, the decrease of their residual variances show that the accuracy of three algorithms are all gradually improved; (2) the law-IISOMAP is almost as accurate as rerunning ISOMAP, because it aims to construct same geodesic distances and embedding results with ISOMAP; (3) when more data are accumulated, the accuracy of our-IISOMAP is slightly better than those of other two algorithms because its neighborhood relationship is more enough; (4) the running time of our-IISOMAP is shortest in three algorithms, and its speed of growth is obviously lowest because m is fixed to a small value.

C. Experiments of data visualization

The CMU hands dataset includes a motion sequence with 481 images of a hand holding a rice bowl under overcast sky. Each image is $480 * 512$ grayscale, but for computing flexibility, we reduce images in $30 * 32$ grayscale. So the dataset can be seen as 481 points in a 960 high dimensional space. Some typical images are shown in Fig. 7(a). 300 points are used to be original points and the initial coordinates are computed by running ISOMAP, with a k-nn neighborhood of size 8. Then other 180 points are divided into 9 sets to be new data sets and three algorithms are used to update the coordinates. That is to say, n is accumulated from 300 to 480 by adding $m = 20$ new samples each time. The 2-dimension embedding results are shown in Fig. 7(b)–(d), and the residual variances and running time are shown in Fig. 8. We can see that our-IISOMAP still can construct an accurate low-dimensional representation of the real images datasets in an efficient manner.

D. Experiments of classification on MNIST dataset

The MNIST database of handwritten digits has a training set of 60 000 examples and a test set of 10 000 examples. The digits have been size-normalized and centered in 28×28 pixels grayscale images. The dataset is comprised of handwriting digit $0 \sim 9$ with its category information. 500 different 2, 3, 5 and 6 images are randomly selected from the training set while 1000 corresponding images are selected from the testing set. 500 points in the testing set

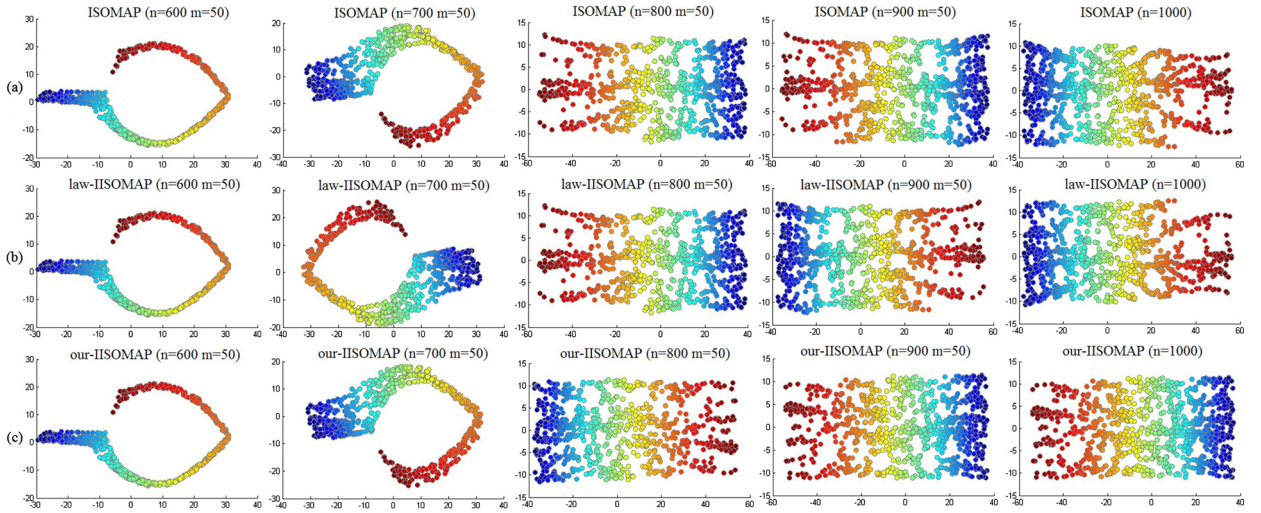

Fig. 5. Comparing the embedding results of ISOMAP, law-IISOMAP and our IISOMAP running on "Swiss roll" dataset. (a) The embedding results of "Swiss roll" after re-running ISOMAP; (b) the embedding results of "Swiss roll" after running law-IISOMAP; (c) the embedding results of "Swiss roll" after running our IISOMAP.

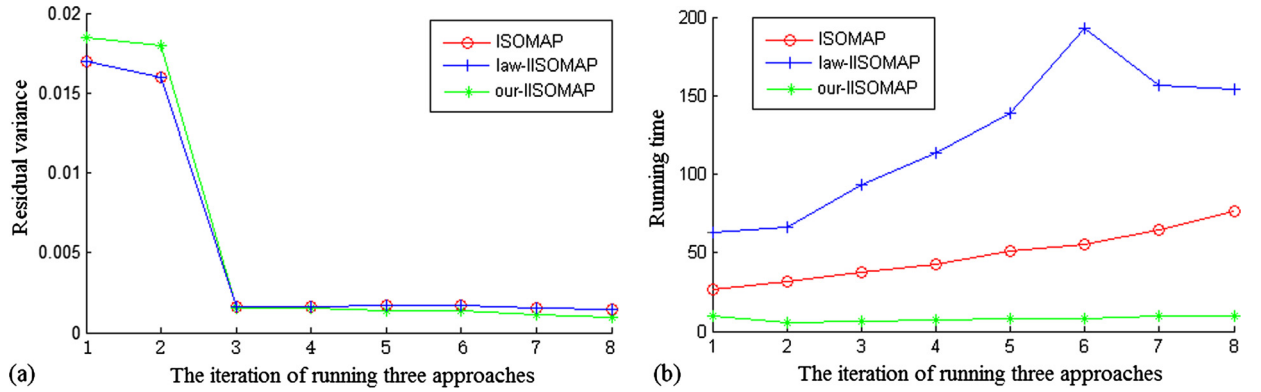

Fig. 6. Comparing the residual variances and running times of ISOMAP, law-IISOMAP and our-IISOMAP running on "Swiss roll" dataset. (a) The residual variances varying with the iteration of running three approaches; (b) the running time varing with the iteration of running three approaches.

are used to be original points and the other 500 points are divided into 10 sets to be new data sets. After the three algorithms updating the coordinates, with a k-nn neighborhood of size 8, the k-NN ($k = 5$) classifier is used for classification because of its simplicity, and the mean classification accuracies of k-NN are shown in Table 1.

Obviously, following the accumulation of the samples, the classification accuracies on the MNIST dataset are improved. The classification accuracies of law-IISOMAP are always equal to those of re-running ISOMAP, because their embedding results are always same. The classification accuracies of our-IISOMAP are better than those of other approaches, which verify the proposed approach further.

E. Experiments on large-scale dataset

In order to validate the efficiency of the proposed algorithm running on the large-scale datasets, three algorithms are running on 4 sets of data. The "Swiss roll" and "S-curve" are standard benchmark for manifold learning. The "Frey face" and "MNIST" datasets are typical real-world image datasets. When the same m new points are accumulated to the same n original points, three algorithms are run respectively, and their residual variances and running time are compared in Table 2.

According to the comparison, the efficiency of our-IISOMAP is the best in three algorithm, and its running time is less than those of other two algorithms obviously, especially when m is small. The running time of law-IISOMAP is the longest in three algorithms, and it may be not fit for large-scale datasets.

6. Conclusion

We have presented a novel method to incrementally update the coordinates produced by ISOMAP. It can make full use of the previous computation results to construct the accurate low-dimensional representation of the new data sets in an efficient manner, when many new data are accumulated into the original date sets. Its performances, especially on the running time, are improved gradually. The main contributions in this paper include: (1) the dynamicalKNN algorithm which updating the neighborhood

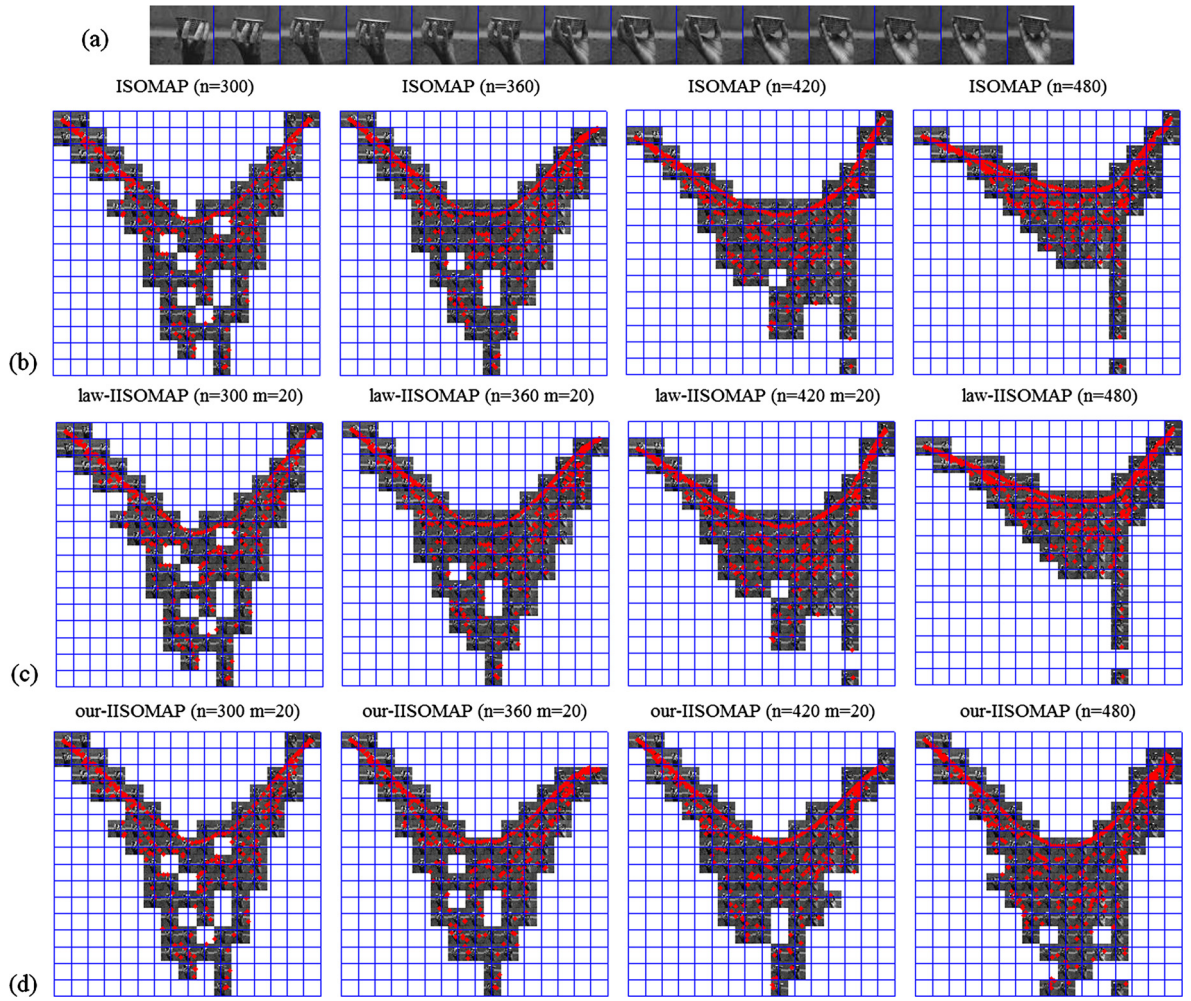

Fig. 7. Comparing the embedding results of ISOMAP, law-IISOMAP and our-IISOMAP running on CMU hands dataset. (a) Typical CMU hand images; (b) 2D embedding results of ISOMAP; (c) 2D embedding results of law-IISOMAP; (d) 2D embedding results of our-IISOMAP.

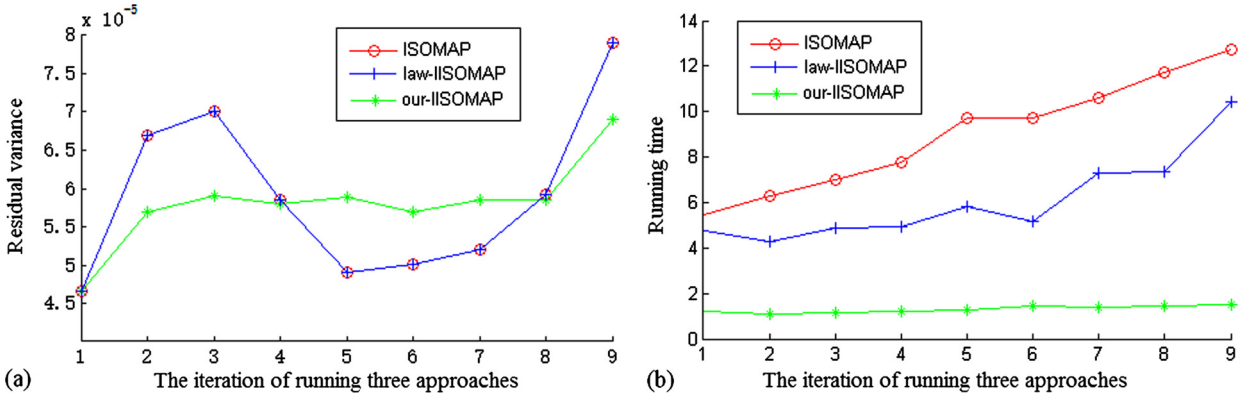

Fig. 8. Comparing the residual variances and running times of ISOMAP, law-IISOMAP and our-IISOMAP running on CMU hands dataset. (a) The residual variances varying with the iteration of running three approaches; (b) the running time varing with the iteration of running three approaches.

graph simply and efficiently; (2) the algorithm to re-estimate the geodesic distances matrix; (3) the algorithm to check out the short circuits and (4) the method to solve the eigen-decomposition problem. And the experiments on synthetic data as well as real world images datasets demonstrate the accuracy and efficiency of the algorithms.

There is still room to improve the accuracy or efficiency of the proposed algorithm. The dynamical neighbor-

Table 1
Comparison of the mean classification accuracies of k-nearest neighbor ($k = 5$). Following the accumulation of the samples, the classification accuracies are improved.

	The number of running three approaches									
	1	2	3	4	5	6	7	8	9	10
ISOMAP	82.74	82.90	83.31	82.75	84.6	84.62	85.15	85.37	85.66	85.91
Law-IISOMAP	82.74	82.90	83.31	82.75	84.6	84.62	85.15	85.37	85.66	85.91
Our-IISOMAP	82.71	83.55	85.43	84.87	85.31	85.84	85.8	85.97	85.92	86.16

Table 2
Comparison of the residual variances and running time of ISOMAP, law-IISOMAP and our-IISOMAP running on large-scale datasets.

	n	m	ISOMAP		Law-IISOMAP		Our-IISOMAP	
			Residual variance	Running time	Residual variance	Running time	Residual variance	Running time
Swiss roll	2000	500	4.2639e−04	252.2107	4.2639e−04	3.7568e+03	3.9637e−04	141.8703
S-curve	2000	500	7.6998e−04	227.5863	7.6998e−04	7.7918e+03	6.3271e−04	211.5516
Frey face	1500	300	0.2476	119.1626	0.2476	2.9894e+03	0.2396	88.8094
MNIST	4500	500	0.3265	1.7650e+03	0.3265	9.9391e+04	0.3258	1.2399e+03

hood can be used instead of k-NN. The update strategy for geodesic distances and coordinates can be more aggressive. In the future, we plan to continually improve its performances and extent the approach to large data.

Acknowledgements

The authors would like to thank all the anonymous reviewers and editors for their helpful comments and suggestions about the improvements of this paper.

References

[1] S. Belongie, et al., Spectral partitioning with indefinite kernels using the Nyström extension, in: Proc. ECCV, 2002.
[2] Y. Bengio, et al., Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and Spectral Clustering [C], in: Advances in Neural Information Processing Systems, Cambridge, 2003.
[3] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, Adv. Neural Inf. Process. Syst. 13 (2001) 682–688.
[4] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, 1996, pp. 405–425.
[5] Li Housen, et al., Incremental manifold learning by spectral embedding methods, Pattern Recognit. Lett. 32 (2011) (2011) 1447–1455.
[6] P. Jia, et al., Incremental Laplacian eigenmaps by preserving adjacent information between data points, Pattern Recognit. Lett. 30 (2009) 1457–1463.
[7] John C. Platt, FastMap, MetricMap, and Landmark MDS are all Nyström algorithms, in: Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics, 2005, p. 261.
[8] I.T. Jolliffe, Principal Component Analysis, Springer-Verlag, New York, 1986.
[9] Z. Karina, et al., Estimation of tangent planes for neighborhood graph correction, in: ESANN'2007 Proceedings—European Symposium on Artificial Neural Networks Bruges, Belgium, 2007, pp. 25–27.
[10] O. Kouropteva, O. Okun, M. Pietikainen, Selection of the optimal parameter value for the locally linear embedding algorithm, in: Proc. of the 1st Int'l Conf. on Fuzzy Systems and Knowledge Discovery, Singapore, 2002, pp. 359–363.
[11] X. Liu, et al., Incremental manifold learning via tangent space alignment, Artif. Neural Netw. Pattern Recognit. 4087 (2006) 107–121.
[12] Martin H.C. Law, Anil K. Jain, Incremental nonlinear dimensionality reduction by manifold learning, IEEE Trans. Pattern Anal. Mach. Intell. 28 (3) (2006) 377–391.
[13] Martin H.C. Law, N. Zhang, A.K. Jain, Non-linear manifold learning for data stream, in: Proc. SIAM International Conference for Data Mining, 2004, 2004, pp. 33–44.
[14] T. Martinetz, K.J. Schulten, Topology representing networks, Neural Netw. 7 (1994) 507–522.
[15] Nathan Mekuz, K. John, Tsotsos. Parameterless isomap with adaptive neighborhood selection, in: K. Franke, et al. (Eds.), DAGM, 2006, Springer-Verlag, Berlin/Heidelberg, 2006.
[16] Olga Kouropteva, et al., Incremental locally linear embedding, Pattern Recognit. 38 (2005) 1764–1767.
[17] S. Roweis, L. Saul, Nonlinear dimentionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[18] M. Steyvers, Multidimensional scaling, in: Encyclopedia of Cognitive Science, 2002.
[19] J. Tenenbaum, et al., A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[20] J. Wang, et al., Adaptive manifold learning, in: NIPS 17, 2004, http://books.nips.cc/papers/files/nips17/NIPS2004_0612.pdf.
[21] K. Weinberger, L.K. Saul, Unsupervised learning of image manifolds by semidefinite programming, in: IEEE Internat. Conf. Computer Vision and Pattern Recognition, vol. 2, 2004, pp. 988–995.
[22] X. Gao, J. Liang, The dynamical neighborhood selection based on the sampling density and manifold curvature for isometric data embedding, Pattern Recognit. Lett. 32 (2011) (2011) 202–209.