SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2022, Vol. 65 192105:1–192105:14
https://doi.org/10.1007/s11432-020-3068-7

Curvature flow learning: algorithm and analysis

Yangyang LI* & Ruqian LU

Key Lab of MADIS Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190*, China*

Received 14 May 2020/Accepted 16 July 2020/Published online 25 August 2022

Abstract In order to describe the nonlinear distribution of the image dataset, researchers propose a manifold assumption, called manifold learning (MAL). The geometry information based on a manifold is measured by the Riemannian metric, such as the geodesic distance. Thus, owing to mining the intrinsic geometric structure of the dataset, we need to learn the real Riemannian metric of the embedded manifold. By the Taylor expansion equation of the Riemannian metric, it clearly indicates that the Riemannian metric is relative to the Riemannian curvature. Based on it, we propose a new algorithm to learn the Riemannian metric by adding the curvature information into metric learning. By optimizing the objective function, we obtain a set of iterative equations. We call this model curvature flow. By employing this curvature flow, we obtain a Mahalanobis metric that approaches the Riemannian metric infinitely and can well uncover the intrinsic structure of the embedded manifold. In theory, we analyze several properties of our proposed method, e.g., the boundedness of the metric and the convergence of curvature flow. To show the effectiveness of our proposed method, we compare our algorithm with several traditional MAL algorithms on three real world datasets. The corresponding results indicate that our proposed method outperforms the other algorithms.

Keywords metric learning, Riemannian metric, curvature, flow learning, manifold learning

Citation Li Y Y, Lu R Q. Curvature flow learning: algorithm and analysis. Sci China Inf Sci, 2022, 65(9): 192105, https://doi.org/10.1007/s11432-020-3068-7

1 Introduction

As a nonlinear distribution of the sample set, the manifold-based assumption has been widely applied in many aspects of machine learning [1]. For example, for a set of data points with high dimensional feature representation, we suppose this dataset is sampled from a nonlinear manifold that is embedded into the high dimensional feature space, and then we can reduce the dimension of these data points by using the mapping from the embedded manifold to the ambient space. We call this machine learning method manifold learning [2, 3] which aims to uncover the low-dimensional embedded intrinsic structure of the dataset. Manifold-based assumption also applies in other machine learning tasks, such as cluster and classification [4]. For example, in computer vision, datasets in different classes can be seen as distributed on multiple manifolds. Since the manifold is a discrete geometric space, the process of the machine learning task based on manifold structure usually involves some geometric structure mining [1]. For example, in cluster task, it uses geodesic distance to measure similarity between two points which are sampled from a manifold. However, due to the nonlinear distribution of the embedded sub-manifold, we need to obtain the global structure of the manifold by aligning the local geometric structure of the manifold.

The common manifold learning algorithms are mainly divided into two classes [5]. One class is called global structure preserved algorithms which aim to preserve the global structure of manifold during dimension reduction, such as Isomap [2]. Another class is called locally preserved methods which preserve the local geometric structure of manifold, such as LEP [6], LLE [3], HLLE [7], and LPP [8]. When mining the local or global geometric structures, these algorithms assume that each local patch of the embedded manifold is close to the Euclidean space and use the Euclidean metric to measure the local distance of the manifold. However, for the Riemannian manifold [9], the intrinsic geometric structure is

* Corresponding author (email: yyli@amss.ac.cn)

determined by the corresponding Riemannian metric. In the process of manifold learning, it is essential to learn an appropriate metric in a low dimensional space, so that the corresponding low dimensional representation can well maintain the geometric structure of the high dimensional data points. In the cluster or classification task, to accurately represent the geodesic distance between two points, we also need to know the Riemannian metric of the manifold. Therefore, it can be seen from the above analysis that the machine learning tasks based on manifold structure cannot avoid Riemannian metric learning.

1.1 Related work

In machine learning, the main purpose of dimensionality reduction for high-dimensional data is to find a suitable low-dimensional space in which learning can perform better than in the original space. In fact, each space corresponds to a distance metric defined on the sample attribute, and finding the suitable space is essentially looking for an appropriate distance metric. Therefore, manifold learning is equivalent to metric learning [10].

Metric learning was first put forward by Xing et al. [11] in 2002, which was proposed to learn a symmetric distance matrix to measure the similarity relationship between any two points. It can be well applied in cluster learning tasks. In recent years, metric learning has made great progress. Almost all metric learning algorithms ultimately are translated into an objective function optimization problem with several constraints [11]. In the optimization process, one method is to add all the constraint conditions to the objective function, which is called offline learning. One problem is the optimization process will become very complex if the number of constraint conditions is very large. To overcome this limitation, some researchers proposed online metric learning which optimizes the objective function step by step by inputting one constraint each time [12]. The distance metric learned in this way is a global metric that measures the similarity between any two points. In general, the distribution of some kinds of datasets cannot be well measured with a global distance metric. For example, if the learned dataset is distributed on a nonlinear Riemannian manifold, there is no global Riemannian metric on it, but only local Riemannian metric in each local neighborhood. Based on this limitation, some researchers proposed local metric learning [13, 14]. The algorithm [14] is proposed to learn a set of local metric basis, and then the corresponding global metric between any two points is represented by the linear combination of this set of metric basis. In this paper, we propose to learn the real Riemannian metric of the manifold by mining the intrinsic geometric structure. As we all know, the global Riemannian metrics on the Riemannian manifold is represented by the set of all local Riemannian metric defined on all local patches of the whole manifold [9]. Therefore, we learn the real Riemannian metric based on the idea of local metric learning.

1.2 Limitations of traditional manifold learning

As we know that the Riemannian metric is not isometric to the Euclidean metric, one main reason is the existence of the Riemannian curvature of the Riemannian manifold. Some researchers [15] proposed to add the curvature information into the semi-supervised model as a regularization item. However, it cannot be applied to Riemannian metric learning. The global structure of the general Riemannian manifold is very complex and its corresponding Riemannian metric structure is also complex, and thus it is impossible to learn an accurate Riemannian metric just by a metric learning model.

Therefore, in order to understand the global structure of the Riemannian manifold, some mathematicians put forward the concept of geometric flow, such as Ricci flow [16], with the aim of establishing the differential equation of time variable between Riemannian metric and Riemannian curvature. Some researchers [17, 18] proposed to use Ricci flow to rectify the similarity among datasets. Our previous work [19] also proposed to add Ricci flow to manifold learning. However, the traditional geometric flow has some limitations, and it cannot be applied to all kinds of manifolds, such as negative curvature manifold. For general manifolds, we proposed to construct a new curvature flow. Based on this curvature flow model, we could learn the Riemannian metric of discrete datasets distributed on any kind of manifold. In our previous work [20], we have put forward this idea. This paper is an extension and in-depth study of our previous work, which is mainly explained in Section 4.

1.3 Main contributions

The main contributions of this paper include the following points:

• A new curvature flow is proposed. Based on this curvature flow, the Riemannian metric can be learned more accurately.

• The chosen parameters are explained in detail during the iterative process of the discrete curvature flow.

• The selection of constraints is proposed, so as to effectively reduce the complexity of the algorithm.

• Several properties of the curvature flow are theoretically analyzed and proved, such as the convergence of curvature flow, the existence of the upper and lower boundary of the Riemannian metric matrix, and the selection of parameters.

2 Basic knowledge

In this section, we focus on introducing some basic knowledge needed for our work. Firstly, we give the relationship between manifold learning and metric learning. Secondly, we introduce the relationship between the Riemannian metric and Riemannian curvature. Finally, we give the problem statement of this paper. Suppose the input dataset is represented as $\{x_1, x_2, \ldots, x_N\}, x_i \in \mathbb{R}^D$, where N is the number of data points and D is the dimension of each data point.

2.1 Metric learning

Metric is a criterion to measure the relative distance between data points. In mathematics, the concept of metric usually satisfies four conditions: symmetry, nonnegative, triangle inequality, and identity. One special metric is represented by a symmetric positive definite (SPD) matrix named Mahalanobis metric, denoted by M. Based on this metric, the Mahalanobis distance between two points is shown as

$$d_M(x_i, x_j) = (x_i - x_j)^{\mathrm{T}} M (x_i - x_j). \tag{1}$$

If the matrix M is set to I (unit matrix), then the metric is called Euclidean metric.

Since metric M is an SPD matrix, it can be decomposed as the following form:

$$M = W^{\mathrm{T}} W. \tag{2}$$

The corresponding distance can be transformed into the following form:

$$d_M(x_i, x_j) = (Wx_i - Wx_j)^{\mathrm{T}} (Wx_i - Wx_j). \tag{3}$$

If the order of matrix W is lower than the dimension of the input data point x_i, the dimension of the input dataset can be reduced by linear transformation. So here it can be shown that metric learning is equivalent to manifold learning.

2.2 Riemannian metric

For the Riemannian manifold, the set of inner products defined on the tangent space of each point is called the Riemannian metric [9]. The ability to accurately uncover the corresponding Riemannian metric directly affects the efficiency of manifold learning. As we have shown, there only exists a local Riemannian metric in each local patch. By the same approach as in the previous paper [20], we use the unit decomposition theorem to represent the global structure of the Riemannian metric. The selection of local neighborhood set also adopts the method mentioned in our previous paper [20]. In Riemannian geometry, Riemannian metric is seriously relative to Riemannian curvature, which is shown by the Taylor expansion of Riemannian metric [21]:

$$g_{ij} = g_{ij}|_p + \partial_k g_{ij}|_p x^k + \frac{1}{2}\partial_l \partial_k g_{ij}|_p x^k x^l + \frac{1}{6}\partial_m \partial_l \partial_k g_{ij}|_p x^k x^l x^m + \cdots, \tag{4}$$

where

$$\begin{aligned} g_{ij}|_p &= \delta_{ij}, \\ \partial_k g_{ij}|_p x^k &= 0, \\ \frac{1}{2}\partial_l \partial_k g_{ij}|_p x^k x^l &= -\frac{1}{3} R_{ikjl}|_p x^k x^l, \\ \frac{1}{6}\partial_m \partial_l \partial_k g_{ij}|_p x^k x^l x^m &= -\frac{1}{6} \nabla_m R_{ikjl}|_p x^k x^l x^m. \end{aligned}$$

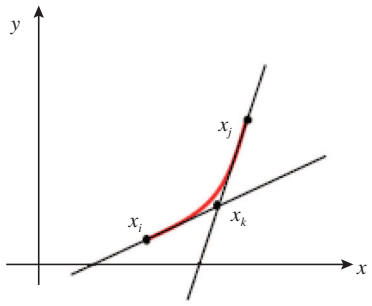

Figure 1 (Color online) Curvature estimation.

g_{ij} is the Riemannian metric and R_{ikjl} is the Riemannian curvature.

By analyzing (4), it can be seen that the Euclidean metric is not isometric to the Riemannian metric which is completely influenced by Riemannian curvature. Therefore, in order to learn the real Riemannian metric of the dataset, it is crucial to excavate the Riemannian curvature first.

Riemannian curvature is a special high-order geometric quantity, which proposes to measure the nonlinearity of the Riemannian manifold [9]. In Riemannian geometry, Riemannian curvature is represented by a fourth-order tensor. However, in machine learning, we only train a set of discrete data points. To calculate the expression of Riemannian curvature at any point, some researchers [15] proposed to calculate the analytic structure of the corresponding local neighborhood. In general, the analytic structure of the space is estimated by a suitable multivariate polynomial function. However, this operation usually has several problems, such as the local coordinate system is bad to determine, the parameter of the polynomial function is bad to determine, and for the dataset with high dimension the time cost of estimating polynomial is especially high. Therefore, it is impractical to represent the Riemannian curvature tensor in this way, if the size of the dataset is especially large.

Back to our paper, the final purpose of this paper is to learn a proper Riemannian metric to uncover the intrinsic geodesic distance between any two points more accurately. In the discrete dataset, the geodesics among data points are independent of each other. Compared with the differential Riemannian manifold, the manifold formed by the discrete dataset does not have many complicated geometric structures, because the differential structure on the discrete dataset does not exist. Therefore, in this paper, we just consider the geodesic curvature between any two points. In our previous work, we have pointed out that the discrete geodesic curvature is shown as

$$|R_{ij}| = \frac{d_M(x_i, x_k) + d_M(x_k, x_j) - d_M(x_i, x_j)}{d_M(x_i, x_k) + d_M(x_k, x_j)}, \tag{5}$$

where R_{ij} represents the geodesic curvature between x_i and x_j, and x_k is the intersection between the tangent vectors of x_i and x_j. An intuitive illustration is shown in Figure 1.

2.3 Problem statement

As we have analyzed, our proposed method aims to use a local metric learning model combined with geodesic curvature information to learn the Riemannian metric. The traditional metric learning model [12] is represented as follows:

$$\min_{M} D_\phi(M, M_0), \tag{6}$$

where $D_\phi(\cdot)$ represents the similarity relationship between two matrices, M_0 is the initial metric.

Since there exists some prior information about the dataset, some constraints are added in the optimization process. For example, in supervised learning, constraints are added to make the Mahalanobis distance between data points in the same class as small as possible and the distance in different classes as large as possible. At the same time, in order to make the metric meet some additional properties, a regularization item is added to modify and punish the metric learning.

In this paper, we propose to learn a real Riemannian metric so that the Mahalanobis distance between two points is as close as possible to the real geodesic distance. Therefore, for this purpose, the objective function can be optimized based on the metric learning model as follows:

$$\min_{M_R} D_\phi(M_R, M_0) + \lambda\left(d_{\mathrm{ge}}(x_i, x_j) - d_{M_R}(x_i, x_j)\right)^2, \tag{7}$$

where M_R represents the Mahalanobis distance that should be learnt, M_0 is the initial Euclidean metric, $d_{\mathrm{ge}}(x_i, x_j)$ represents the geodesic distance between x_i and x_j, and $d_{M_R}(x_i, x_j)$ is the Mahalanobis distance between x_i and x_j.

In general, the learned manifold is embedded into a high dimensional feature space, therefore, the curvature of the embedded manifold is relative to the ambient space. Measuring the curvature of geodesic can be viewed as measuring the difference between geodesic distance and the corresponding Euclidean distance. Therefore, the geodesic curvature can be approximately estimated as shown in (8), and the corresponding discrete representation is shown in (5).

$$R_{ij} = \frac{d_{\mathrm{ge}}(x_i, x_j) - d_{\mathrm{Eu}}(x_i, x_j)}{d_{\mathrm{ge}}(x_i, x_j)}, \tag{8}$$

where $d_{\mathrm{Eu}}(x_i, x_j)$ represents the Euclidean distance between x_i and x_j and R_{ij} represents the geodesic curvature between x_i and x_j.

Therefore, for the regularization item in the objective function, we can replace it with geodesic curvature, and combining with (8) we can convert (7) into the following form:

$$\min_{M_R} D_\phi(M_R, M_0) + \lambda_1 |R_{ij}|^2. \tag{9}$$

In other words, our algorithm adds a regularization item about geodesic curvature in the optimization process, so that in the process of learning, the curvature of geodesic is decreasing, and the Mahalanobis distance is getting closer and closer to the geodesic distance.

The main task of our method is to solve this objective function. The optimization function establishes the relationship between the Riemannian metric and curvature. Because of the nonlinearity of the Riemannian manifold, it is impossible for the objective function to obtain the optimal result at one time. In Section 3, we give the optimization process using online learning model and analyze the relationship between the optimization result and the curvature flow.

3 Algorithm

The main purpose of this section is to describe the overall process of our proposed algorithm in detail. We need to solve three problems: the first one is to describe the optimization process of the objective function in detail, the second one is to explain the selection of constraint point pairs, and finally, the selection of parameters in the objective function should be described in detail.

3.1 Curvature flow

We have pointed out that the final purpose of our algorithm is to learn the Riemannian metric, so that the geometric structure of the embedded manifold can be well mining. In the learning process, we still propose to preserve the local structure of the manifold, so we add a set of constraints on the local neighborhood. In order to avoid the limitations of offline learning, we use the online learning model [22, 23] to optimize the objective function, which means that only one constraint condition is added to the objective function at each iteration. The corresponding objective function can be transformed into the following form:

$$\begin{aligned} \arg\min \quad & D_\phi(M_{t+1}, M_t) + \lambda_{1,t} |R_t|^2, \\ \text{s.t.} \quad & f_i(x_j) \cdot d_{M_t}(x_i, x_j) < \rho, \end{aligned} \tag{10}$$

where $f_i(x_j)$ is defined as follows:

$$f_i(x_j) = \begin{cases} 1, & \text{if } x_j \in U_i, \\ 0, & \text{elsewise.} \end{cases}$$

As the same method with our previous paper, we use Log-det divergence $D_{\mathrm{ld}}(M_R, M_0)$ [24] to measure the similarity $D_\phi(M_R, M_0)$ between two SPD matrices. So,

$$D_{\mathrm{ld}}(M_R, M_0) = \mathrm{tr}\left(M_R M_0^{-1}\right) - \log\det\left(M_R M_0^{-1}\right) - d. \tag{11}$$

In the optimization process, we use the Lagrange multiplier method to solve the objective function. The corresponding Lagrange multiplier formula is shown as follows:

$$L\left(M_t, \lambda_{1,t}, \lambda_{2,t}\right) = D_{\mathrm{ld}}\left(M_{t+1}, M_t\right) + \lambda_{1,t}|R_t|^2 + \lambda_{2,t}\left(f_i\left(x_j\right) \cdot d_{M_t}\left(x_i, x_j\right) - \rho\right). \tag{12}$$

The result is shown as

$$M_{t+1} = [M_t^{-1} + 2\lambda_{1,t}R_tA_{ik} + 2\lambda_{1,t}R_tA_{kj} - (2\lambda_{1,t}R_t - \lambda_{2,t})\,A_{ij}]^{-1}. \tag{13}$$

We next use the Shermann-Morrison inverse formula [24] and obtain the following iterative formulas:

$$\Gamma_t = M_t - \frac{2\lambda_{1,t}R_tM_tA_{ik}M_t}{1 + 2\lambda_{1,t}R_tv_{ik}^{\mathrm{T}}M_tv_{ik}}, \tag{14}$$

$$H_t = \Gamma_t - \frac{2\lambda_{1,t}R_t\Gamma_tA_{kj}\Gamma_t}{1 + 2\lambda_{1,t}R_tv_{kj}^{\mathrm{T}}\Gamma_tv_{kj}}, \tag{15}$$

$$M_{t+1} = H_t + \frac{(2\lambda_{1,t}R_t - \lambda_{2,t})\,H_tA_{ij}H_t}{1 - (2\lambda_{1,t}R_t - \lambda_{2,t})\,v_{ij}^{\mathrm{T}}H_tv_{ij}}, \tag{16}$$

where $v_{ij} = x_i - x_j$, and the same as v_{ik} and v_{jk}.

We call this iterative process curvature flow. One reason is that the iterative process is a set of equations about the metric and curvature. Unlike traditional geometric flows, such as Ricci flow, defined on Riemannian manifolds, our curvature flow is learned from a set of discrete data points. Under this curvature flow, the manifold structure continues to deform along the time axis and eventually converges to a flat Euclidean space. The learned Mahalanobis metric can well uncover the geodesic distance between data points.

3.2 Selection of constraints

In this subsection, we set the selection rules for constraints. In our previous work [20], we iterated through all the point pairs in the neighborhood and optimized these constraints one by one during the iteration of the objective function. This greedy method can result in a very large time consumption of the optimization process. In this paper, we re-analyze the selection of constraints to minimize the number of constraints without affecting the optimization results. Therefore, for this purpose, we set some selection rules for constraint conditions, which are shown as follows:

- Since the Riemannian metric only has local representation, the constraint data point pair is limited to the same neighborhood and the non-neighborhood is not selected.
- In the same neighborhood, we only choose the constraint between the center point and its non-center point.
- Set a threshold δ, and select the point pairs where the Mahalanobis distance between them is greater than δ.

Supposing the number of the minimum overlapping neighborhood sets is m, the number of sample points in each neighborhood is K, then the number of constraints constructed based on our proposed rules is approximate $O\left(mK\right)$. Compared with the number of the input data points N, the number of constraints constructed by this method is much smaller. Therefore, selecting constraints based on this rule would greatly reduce the number of constraints.

3.3 Parameter selection

From the algorithm iteration process above, we can see that the curvature flow is affected by two parameters $\lambda_{1,t}, \lambda_{2,t}$. In this subsection, we explain the selection of these two parameters. These two parameters respectively control the effect of curvature item and constraint conditions in the optimization process. If $\lambda_{1,t}$ is large, the curvature of geodesic between two points would be reduced, while the corresponding Mahalanobis distance would be very large under this learned metric. If $\lambda_{2,t}$ is large, the learned metric can keep the local structure of each neighborhood well, but cannot reduce the curvature value between data points too much. Therefore, we should choose suitable parameters, so that the learned metric achieves the optimal result. In order to select the suitable parameters, the learned metric under these parameters should satisfy the following three conditions:

- The metric M_{t+1} satisfies the semi-positive property, that is $M_{t+1} \succeq 0$, equivalently $M_{t+1}^{-1} \succeq 0$.
- The metric M_{t+1} still retains the local geometry between the neighborhood points, that is

$$(x_i - x_j)^{\mathrm{T}} M_{t+1} (x_i - x_j) < \rho.$$

- Both parameters should be non-negative:

$$\lambda_{1,t} \geqslant 0, \ \lambda_{2,t} \geqslant 0.$$

For the first condition, we can get the corresponding representation as follows:

$$M_{t+1}^{-1} = M_t^{-1} + 2\lambda_{1,t} R_t A_{ik} + 2\lambda_{1,t} R_t A_{kj} - (2\lambda_{1,t} R_t + \lambda_{2,t}) A_{ij} \succeq 0. \tag{17}$$

For the second condition, we can obtain the following representation:

$$(x_i - x_j)^{\mathrm{T}} M_{t+1} (x_i - x_j) < \rho. \tag{18}$$

By computing these three inequality conditions, the parameters $\lambda_{1,t}$ and $\lambda_{2,t}$ are estimated as

$$\begin{aligned} \lambda_{1,t} &= \frac{\eta_t}{2R_t \mathrm{tr}\left(M_t A_{kj}\right)}, \\ \lambda_{2,t} &= \frac{\eta_t}{d_{ij}} \mathrm{tr}[((A_{ik} + A_{kj} + A_{ij}) M_t)^{-1}], \end{aligned} \tag{19}$$

where η_t is a learning rate parameter, which is set to $(0, 1)$, $A_{ij} = (x_i - x_j)(x_i - x_j)^{\mathrm{T}}$, the same as A_{ik} and A_{kj}.

Based on the above three problems solved, the detailed algorithm process is shown in Algorithm 1. we would give a detailed theoretical analysis of our proposed curvature flow in Section 4.

Algorithm 1 Riemannian metric learning based on curvature flow (CFML)

Input: Training data points $\{x_1, x_2, \ldots, x_N\}, x_i \in \mathbb{R}^{\mathcal{D}}$, neighbor-size parameter K, distance threshold ρ, Euclidean metric M_0, slack parameter η;
Output: Riemannian metric M_R, low-dimensional representations $\{y_1, y_2, \ldots, y_N\}, y_i \in \mathbb{R}^d$;
1. **for** $i = 1$ **to** N **do**
2. Find K-nearest neighbor set of x_i, denoted as U_i;
3. **end for**
4. Find a minimum set of overlapping local neighborhoods $\{U_i\}_{i=1}^m$ of $\mathcal{M}$;
5. **Initialization:** $t \leftarrow 1$, $M_0 = I$, $M_1 \leftarrow M_0$, $\lambda_{1,1}$, $\lambda_{2,1}$;
6. **Repeat**
7. Update $\lambda_{1,t}$, $\lambda_{2,t}$ using (19);
8. Update the curvature flow equations using (14)–(16).
9. **Until** Convergence;
10. **Do:** Singular value decomposition to M_R using (2) and obtain $W_R \in \mathbb{R}^{d \times \mathcal{D}}$;
11. **Return** $y_i = W_R \cdot x_i$.

4 Theoretical analysis

In this section, we first attempt to analyze several properties of curvature flow, such as the convergence of curvature flow and the boundedness of metric. Secondly, we give a detailed computational cost analysis of this method.

4.1 Three properties

Theorem 1. If the metric M_t satisfies $0 \preceq M_t \preceq I$ during the iteration process, we can obtain $0 \preceq M_{t+1} \preceq I$.

Proof. Since $\lambda_{1,t}$ is negative, we can obtain directly

$$\begin{aligned} &\Gamma_t \preceq I, \text{ if } M_t \preceq I, \\ &H_t \preceq I, \text{ if } M_t \preceq I. \end{aligned}$$

First we prove that $M_{t+1} \succeq 0$. From the representation of M_{t+1}, we can see that to prove the semi-definiteness of M_{t+1}, we only need to prove the semi-definiteness of H_t.

$$\begin{aligned} H_t &= \Gamma_t - \frac{2\lambda_{1,t} R_t \Gamma_t A_{kj} \Gamma_t}{1 + 2\lambda_{1,t} R_t v_{kj}^{\mathrm{T}} \Gamma_t v_{kj}} \\ &= M_t - \frac{2\lambda_{1,t} R_t M_t A_{ik} M_t}{1 + 2\lambda_{1,t} R_t v_{ik}^{\mathrm{T}} M_t v_{ik}} - \frac{2\lambda_{1,t} R_t \Gamma_t A_{kj} \Gamma_t}{1 + 2\lambda_{1,t} R_t v_{kj}^{\mathrm{T}} \Gamma_t v_{kj}} \\ &\succeq M_t - \frac{2\lambda_{1,t} R_t M_t A_{ik} M_t}{1 + 2\lambda_{1,t} R_t v_{ik}^{\mathrm{T}} M_t v_{ik}} - \frac{2\lambda_{1,t} R_t M_t A_{kj} M_t}{1 + 2\lambda_{1,t} R_t v_{kj}^{\mathrm{T}} \Gamma_t v_{kj}} \\ &\succeq M_t - 2\lambda_{1,t} R_t M_t A_{ik} M_t - 2\lambda_{1,t} R_t M_t A_{kj} M_t. \end{aligned} \tag{20}$$

According to the choice of parameters, we know that $\lambda_{1,t} = \frac{\eta_t}{2R_t \mathrm{tr}(M_t A_{kj})}$. Under this parameter, the above inequality is transformed into the following form:

$$H_t \succeq M_t - 2\eta_t M_t. \tag{21}$$

Therefore, from the above inequality, we can see that if $\eta_t < 0.5$, $H_t \succeq 0$. Then, we can prove that $M_{t+1} \succeq 0$.

Next, let us prove that $M_{t+1} \preceq I$. First we give the iteration form of M_{t+1}:

$$I - M_{t+1} = I - H_t - \frac{2\lambda_{1,t} R_t M_t A_{ik} M_t}{1 - (2\lambda_{1,t} R_t - \lambda_{2,t}) v_{ij}^{\mathrm{T}} H_t v_{ij}}. \tag{22}$$

With proper scaling, the above inequality can be translated into the following form:

$$I - M_{t+1} \succeq I - M_t + \lambda_{1,t} R_t M_t A_{ik} M_t + \lambda_{1,t} R_t \Gamma_t A_{kj} \Gamma_t - \frac{(2\lambda_{1,t} R_t - \lambda_{2,t}) M_t A_{ij} M_t}{1 - (2\lambda_{1,t} R_t - \lambda_{2,t}) v_{ij}^{\mathrm{T}} M_t v_{ij}}. \tag{23}$$

When the learning rate parameter $\eta_t \to 1$, we can prove that $I - M_{t+1} \succeq 0$.

The purpose of this paper is to study a new Mahalanobis metric, so that the Mahalanobis distance between any two neighborhood points is constantly approaching the corresponding geodesic distance. In other words, under the new Mahalanobis metric, the geodesic curvature decreases continuously. We present the detailed proof in Theorem 2.

Theorem 2. As the iteration process, the geodesic curvature between two points is decreasing under the new Mahalanobis metric.

Proof. This theorem is proved mainly by the properties of similar triangles. By the definition of discrete geodesic curvature in (5), if the change ratio of $d_{M_{t+1}}(x_i, x_j)$ is greater than the other two sides, the geodesic curvature of x_i and x_j decreases in the iterative process.

In the following, we respectively give the Mahalanobis distance variation of the three edges in each iteration:

$$\begin{aligned} \Delta d_{M_{t+1}}(x_i, x_k) = &- \frac{2\lambda_{1,t} R_t d_{M_t}(x_i, x_k)}{1 + 2\lambda_{1,t} R_t d_{M_t}(x_i, x_k)} \\ &- \frac{2\lambda_{1,t} R_t d_{\Gamma_t}(x_k, x_j) \cos^2\theta_3}{1 + 2\lambda_{1,t} R_t d_{\Gamma_t}(x_k, x_j)} + \frac{(2\lambda_{1,t} R_t + \lambda_{2,t}) d_{H_t}(x_i, x_j) \cos^2\theta_1}{1 - (2\lambda_{1,t} R_t + \lambda_{2,t}) d_{H_t}(x_i, x_j)}, \end{aligned} \tag{24}$$

$$\begin{aligned} \Delta d_{M_{t+1}}(x_i, x_j) = &- \frac{2\lambda_{1,t} R_t d_{M_t}(x_i, x_k) \cos^2\theta_1}{1 + 2\lambda_{1,t} R_t d_{M_t}(x_i, x_k)} \\ &- \frac{2\lambda_{1,t} R_t d_{\Gamma_t}(x_k, x_j) \cos^2\theta_2}{1 + 2\lambda_{1,t} R_t d_{\Gamma_t}(x_k, x_j)} + \frac{(2\lambda_{1,t} R_t + \lambda_{2,t}) d_{H_t}(x_i, x_j)}{1 - (2\lambda_{1,t} R_t + \lambda_{2,t}) d_{H_t}(x_i, x_j)}, \end{aligned} \tag{25}$$

$$\begin{aligned} \Delta d_{M_{t+1}}(x_j, x_k) = &- \frac{2\lambda_{1,t} R_t d_{M_t}(x_i, x_k) \cos^2\theta_3}{1 + 2\lambda_{1,t} R_t d_{M_t}(x_i, x_k)} \\ &- \frac{2\lambda_{1,t} R_t d_{\Gamma_t}(x_k, x_j)}{1 + 2\lambda_{1,t} R_t d_{\Gamma_t}(x_k, x_j)} + \frac{(2\lambda_{1,t} R_t + \lambda_{2,t}) d_{H_t}(x_i, x_j) \cos^2\theta_2}{1 - (2\lambda_{1,t} R_t + \lambda_{2,t}) d_{H_t}(x_i, x_j)}. \end{aligned} \tag{26}$$

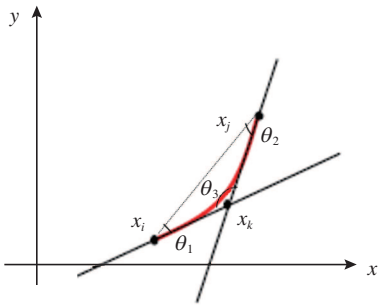

Figure 2 (Color online) Intersection angles.

For the convenience of analysis, we give the following hypothesis:

$$A_1 = \frac{2\lambda_{1,t}R_t d_{M_t}(x_i, x_k)}{1 + 2\lambda_{1,t}R_t d_{M_t}(x_i, x_k)}, \quad A_2 = \frac{2\lambda_{1,t}R_t d_{\Gamma_t}(x_j, x_k)}{1 + 2\lambda_{1,t}R_t d_{\Gamma_t}(x_j, x_k)},$$

$$A_3 = \frac{(2\lambda_{1,t}R_t + \lambda_{2,t})\, d_{H_t}(x_i, x_j)}{1 - (2\lambda_{1,t}R_t + \lambda_{2,t})\, d_{H_t}(x_i, x_j)}.$$

The above incremental representation is then converted to the following representation:

$$\begin{aligned}
\Delta d_{M_{t+1}}(x_i, x_k) &= -A_1 - A_2\cos^2\theta_3 + A_3\cos^2\theta_1, \\
\Delta d_{M_{t+1}}(x_i, x_j) &= -A_1\cos^2\theta_1 - A_2\cos^2\theta_2 + A_3, \\
\Delta d_{M_{t+1}}(x_j, x_k) &= -A_1\cos^2\theta_3 - A_2 + A_3\cos^2\theta_2,
\end{aligned} \tag{27}$$

where the intersection angles θ_1, θ_2, and θ_3 are shown in Figure 2.

It can be seen from the analysis of these three increments that the incremental ratio of $\Delta d_{M_{t+1}}(x_i, x_j)$ is larger than the other two increments. Therefore, under this metric, the geodesic curvature between x_i and x_j is decreasing.

In Theorem 2, we have proved that the curvature decreases during the iterative process, while we do not prove that the geodesic curvature converges to zero. In the next theorem, we prove the convergence of curvature flow. The final conclusion is shown in Theorem 3.

Theorem 3. The curvature flow convergences if and only if there exists a metric with zero geodesic curvature everywhere.

Proof. First suppose there exists a metric M_T, under this metric, the geodesic curvature at any point of the manifold is zero, that is $R_T = 0$. Then we consider the next iteration metric M_{T+1} according to (14)–(16) shown as follows:

$$\begin{aligned}
\Gamma_T &= M_T, \\
H_T &= M_T, \\
M_{T+1} &= M_T - \frac{\lambda_{2,T} M_T A_{ij} M_T}{1 - \lambda_T v_{ij}^{\mathrm{T}} M_T v_{ij}}.
\end{aligned}$$

If the geodesic curvature is zero, the Mahalanobis distance is equivalent to the geodesic distance and the parameter $\lambda_{2,T}$ is set to zero. That is

$$M_{T+1} = M_T.$$

This is proved that the curvature flow convergences.

Next, suppose the curvature flow convergences, and then we have $M_t \to M^*$, as $t \to \infty$, which implies that there exists a sequence $\xi_n \to \infty$ such that

$$M_{\xi_{n+1}} - M_{\xi_n} \to 0.$$

Since the Mahalanobis distance metric M_t convergences, the distance between two points is constant, thus $\lambda_{2,\xi_n} = 0$. That is

$$M_{\xi_{n+1}} - M_{\xi_n} = R_{\xi_n}\left(-\frac{2\lambda_{1,\xi_n} M_n A_{ik} M_{\xi_n}}{1 + 2\lambda_{1,\xi_n} R_{\xi_n} v_{ik}^{\mathrm{T}} M_{\xi_n} v_{ik}} - \frac{2\lambda_{1,\xi_n}\Gamma_{\xi_n} A_{jk}\Gamma_{\xi_n}}{1 + 2\lambda_{1,\xi_n} R_{\xi_n} v_{kj}^{\mathrm{T}}\Gamma_{\xi_n} v_{jk}} + \frac{2\lambda_{1,\xi_n} H_{\xi_n} A_{ij} H_{\xi_n}}{1 - 2\lambda_{1,\xi_n} R_{\xi_n} v_{ij}^{\mathrm{T}} H_{\xi_n} v_{ik}}\right).$$

As $\xi_n \to \infty$, $M_{\xi_n+1} - M_{\xi_n} \to 0$, we have $R_{\xi_n} \to R^*$, $R^* = 0$.

4.2 Computation cost

In this subsection, we give the time complexity analysis of our method. For the sake of analysis, we define a set of symbols, the number of data points N, the input dimension D, the intrinsic dimension d, the neighbor-size k, and the number of minimal neighborhood subsets. The computation cost of this method is mainly contained in the following steps. (1) The time cost for obtaining the minimal neighborhood subset is $O(m^2\,(D+k))$. (2) For the selection of neighborhood pairs, the time complexity of this step is $O\,(mk)$. (3) For the online learning iterative process, the time complexity of each iteration is polynomial. (4) In the process of dimensionality reduction, it is actually to decompose the metric matrix, so the time cost is $O(dN^2)$. Overall, the total time cost of our proposed method is polynomial.

5 Experiments

In this section, we mainly run a set of experiments to test the performance of our proposed algorithm. First, we compare with several traditional dimension reduction algorithms and metric learning algorithms on three real-world datasets, e.g., YTC dataset, USPS dataset, and Extended Yale Face B dataset (YFB dataset). Secondly, we run experiment on two synthetic datasets, e.g., Puncture Sphere and Twin Peaks[1)].

5.1 Dataset

In this subsection, we first introduce the three real-world datasets in detail.

5.1.1 *YTC dataset*

The YouTube celebrity dataset short for YTC dataset collects 1910 video sequences of 47 subjects from YouTube. The face regions of the original images from the video sequences are not all the same and especially high. We resize the face region into 20×20 intensity image. Moreover, we transform each 20×20 pixel matrix into a 400 dimension vector. In this experiment, we respectively choose the video sequences from 30 different individuals, and for each individual we choose 20 video sequences respectively.

5.1.2 *USPS dataset*

The USPS database consists of 9298 images. It refers to numeric data obtained from the scanning of handwritten digits from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations. The images here have been size normalized, resulting in 16×16 gray-scale images, so the original dimension of this database is 256.

5.1.3 *YFB dataset*

The YFB dataset contains 2414 single light source images of 38 individuals each seen under about 64 near frontal images. For every subject in a particular pose, an image with ambient illumination is also captured. The face region in each image is resized into 32×32, so the original dimension of this database is 1024.

5.2 Evaluation

In this experiment, the test procedures are mainly divided into two steps. For these three used datasets, we first use our proposed method and several traditional manifold learning algorithms to reduce their dimensions. Then, we use the K-nearest neighbor classifier to test the classification accuracy of different datasets in the low dimensional space. Here, the classification accuracy is set to measure the dimension reduction accuracy of different algorithms.

1) Wittman T. Manifold learning Matlab demo. 2015. http://www.math.umn.edu/wittman/mani.

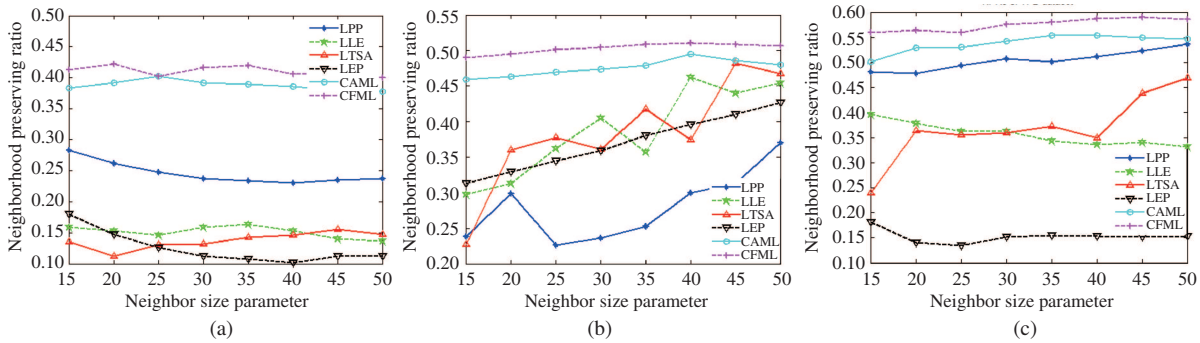

Figure 3 (Color online) The neighborhood preserving ratios of different dimension reduction algorithms about the YTC dataset (a), USPS dataset (b), and YFB dataset (c).

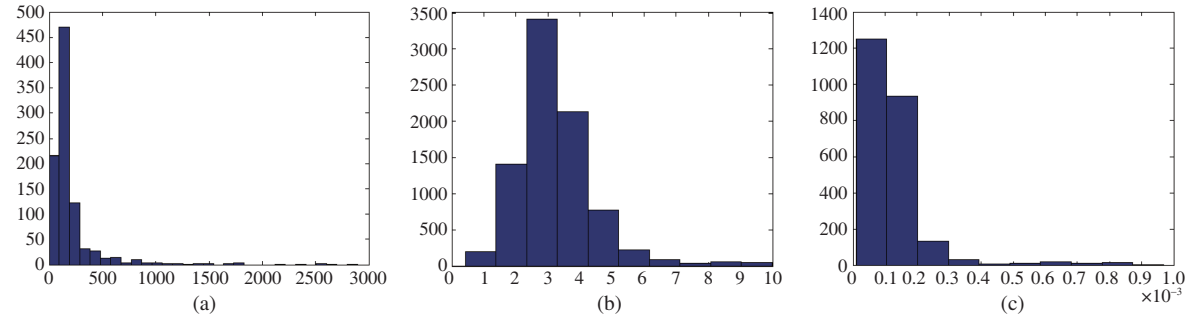

Figure 4 (Color online) The curvature distributions of the YTC dataset (a), USPS dataset (b), and YFB dataset (c).

5.2.1 *Dimension reduction*

In this subsection, to fully evaluate the performance of our proposed algorithm, we compare our method with five traditional manifold learning algorithms (e.g., Isomap, LLE, LEP, HLLE, LTSA). We use these algorithms to reduce the dimension of the three datasets. For YTC dataset, the corresponding lower dimension is set to 20, for USPS dataset, the lower dimension is set to 10, and for YFB dataset, we choose the lower dimension as 10.

We first test the sensitivity of these algorithms about neighbor size parameter K by computing the neighborhood preserving ratio (NPR) during dimension reduction. The NPR is defined as

$$\mathrm{NPR} = \frac{1}{KN}\sum_{i=1}^{N} |\mathcal{N}(x_i) \cap \mathcal{N}(y_i)|, \tag{28}$$

where x_i represents the input data point and y_i is the corresponding low dimensional representation; $\mathcal{N}(x_i)$ is the set of subscripts $\{j\}$, where x_j is the K-nearest neighbor of y_i; $|\cdot|$ represents the number of intersection points.

We compare the NPRs of different algorithms with respect to (w.r.t) different neighbor-size parameter values K. For these three datasets, K is set to $K = \{15, 20, 25, 30, 35, 40, 45, 50\}$, and the corresponding comparison result is shown in Figure 3. From Figure 3, we can see that, for YTC dataset, the NPRs of traditional manifold learning algorithms are especially low, mostly focusing on 0.1 to 0.3. For USPS and YFB datasets, the NPRs of traditional algorithms are ranged 0.3 to 0.5. In general, the dimension reduction results of traditional MAL algorithms are not very well. One obvious reason is that the traditional algorithms do not consider the curvature information of the datasets, which has a great influence on the results of dimension reduction.

The corresponding curvature distributions of these three datasets are shown in Figure 4. From Figure 4, we can see that the average curvature value of YTC dataset is higher than the other two datasets. In order to analyze the influence of curvature information on dimension reduction, we use CAML [25] and our proposed algorithm CFML to compare with traditional MAL algorithms, and the results are shown in Figure 3. For these two curvature-aware algorithms, they obviously outperform the traditional MAL algorithms. Moreover, the NPRs of our proposed algorithm are higher than the CAML algorithm in almost any case.

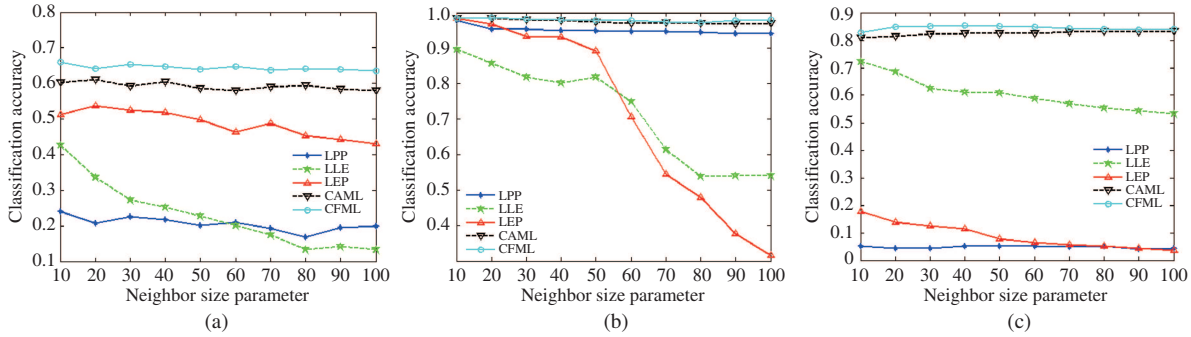

Figure 5 (Color online) The classification accuracy comparisons of the YTC dataset (a), USPS dataset (b), and YFB dataset (c).

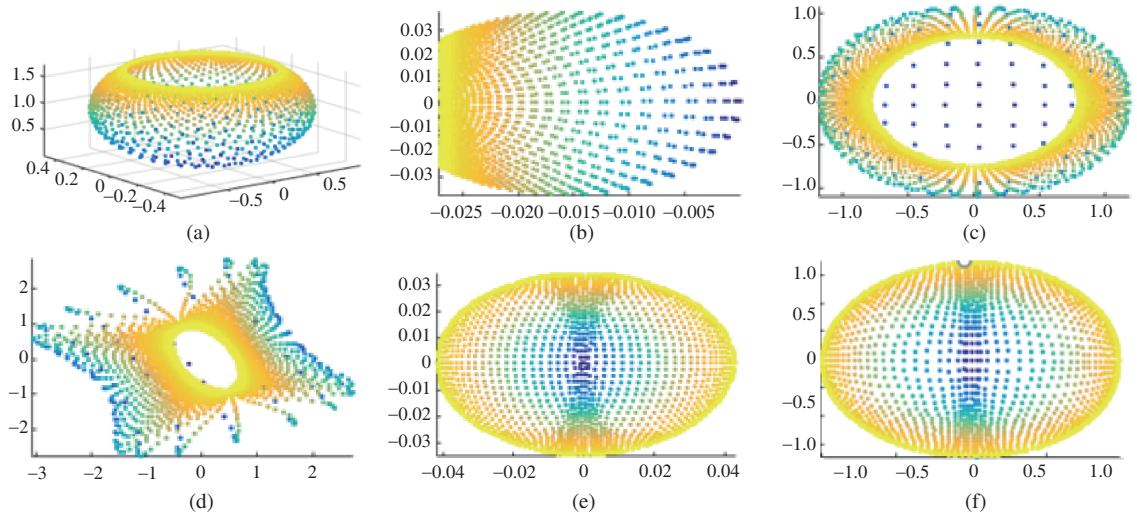

Figure 6 (Color online) The visual embedding comparison of puncture sphere dataset: (a) original dataset, (b) PCA, (c) Isomap, (d) LLE, (e) LEP, and (f) CFML.

5.2.2 *Classification accuracy*

In this experiment, we mainly compare our proposed algorithm with several existing dimension reduction algorithms about the classification accuracies of these three datasets. The experiment is mainly divided into two steps. The first step is to use dimension reduction algorithms to reduce the dimension of these three datasets, and the dimension reduction results have been obtained in Subsection 5.2.1. In this subsection, we do the second step to use the KNN classifier to compute the classification accuracy of different datasets.

For YTC dataset, we respectively choose 15 video sequences per individual for training, and the rest video sequences as a test set. For USPS dataset, we choose 400 images per subject for training, and the rest for a test. And for YFB dataset, the training set is chosen 30 images per individual. Both the KNN classifier and dimension reduction algorithms are influenced by the value of neighbor-size parameter K. Thus we run a series of experiments to analyze the sensitivity of parameter K. The corresponding result is shown in Figure 5. From experimental results on YTC dataset, we can see that our proposed algorithm achieves the highest accuracy rate in all cases. For the curvature-aware algorithms, focusing on the comparison between CAML and our proposed algorithm, our method outperforms CAML everywhere.

5.3 Visual embedding

Owing to visualizing the embedding results of datasets, we compare our proposed method with several traditional manifold learning algorithms on two synthetic databases, e.g., Punctured Sphere and Twin Peaks.

In this experiment, we use different algorithms to map the 3D synthetic databases into two dimensional space. The intuitive embedding results are shown in Figures 6 and 7. From these two figures, we can

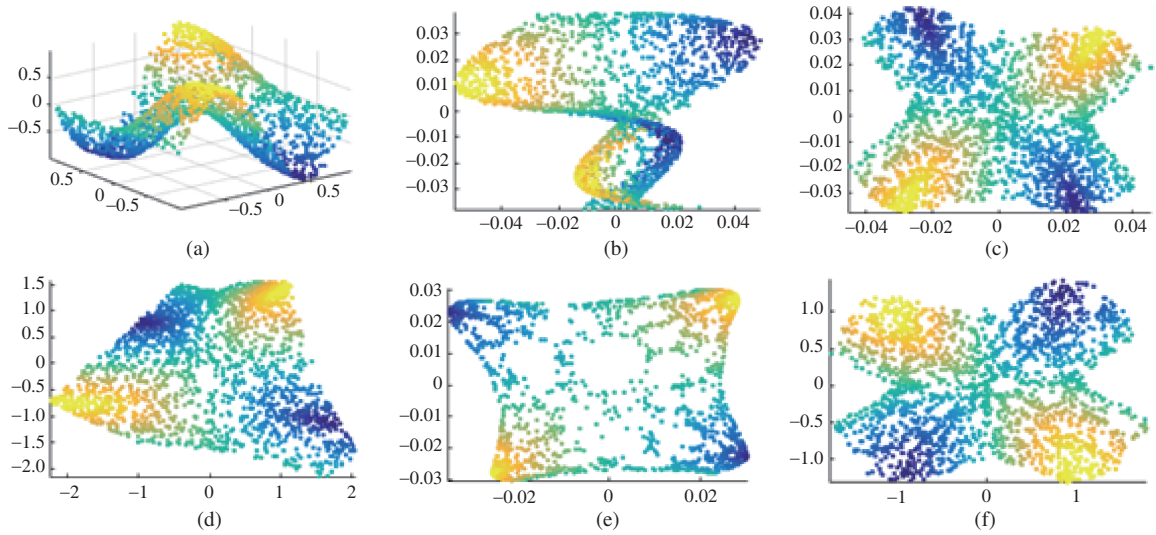

Figure 7 (Color online) The visual embedding comparison of Twin Peaks dataset: (a) original dataset, (b) PCA, (c) Isomap, (d) LLE, (e) LEP, and (f) CFML.

see that the embedding results of Puncture Sphere dataset and Twin Peaks dataset are highly sensitive to the curvature information. The results of our proposed method outperform the other traditional MAL algorithms. By the visually embedding comparison, we can see that for almost all general nonlinear manifolds, the curvature information is necessary to be considered for the dimension reduction algorithms.

6 Conclusion

At present, almost all researchers use graph embedding ideas to reduce the dimensionality of dataset, which does not consider the higher order geometric structure of the dataset. To precisely uncover the intrinsic structure of the embedded sub-manifold, we attempt to consider the curvature information of data points. Based on the idea of traditional geometric flow learning, we propose a new curvature flow learning to learn a new Mahalanobis metric, which approaches the Riemannian metric infinitely. In the experiment, we choose three real world datasets and run a series of experiments to compare with traditional manifold learning algorithms. The comparison result shows that our proposed method outperforms the other algorithms. The main contribution of our work presented in this paper is that we give some properties of the proposed curvature flow model, such as the convergence of curvature and the boundedness of Mahalanobis matrix, and get the full proof.

However, currently our method can only train on the small size dataset. For big data, owing to insufficient CPU memory, it does not work well. In the future, we intend to combine deep learning model to train our algorithm on big data.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2016YFB1000902) and National Natural Science Foundation of China (Grant Nos. 61472412, 61621003).

References

1 Lin B, He X, Ye J. A geometric viewpoint of manifold learning. Appl Inform, 2015, 2: 3
2 Tenenbaum J B, Silva V D, Langford J C. A global geometric framework for nonlinear dimensionality reduction. Science, 2000, 290: 2319–2323
3 Roweis S T. Nonlinear dimensionality reduction by locally linear embedding. Science, 2000, 290: 2323–2326
4 Harandi M T, Sanderson C, Wiliem A, et al. Kernel analysis over Riemannian manifolds for visual recognition of actions, pedestrains and textures. In: Proceedings of IEEE Workshop on the Applications of Computer Vision, 2012
5 Silva V D, Tenenbaum J B. Global versus local methods in nonlinear dimensionality reduction. In: Proceedings of Conference and Workshop on Neural Information Processing Systems, 2003. 16: 705–712
6 Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Proceedings of Conference and Workshop on Neural Information Processing Systems, 2001. 14: 585–591
7 Donoho D L, Grimes C E. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. Proc Natl Acad Sci USA, 2003, 100: 5591–5596
8 He X, Niyogi P. Locality preserving projections. In: Proceedings of Conference and Workshop on Neural Information Processing Systems, 2003. 16: 153–160
9 Lee J M. Riemannian Manifolds: An Introduction to Curvature. New York: Springer, 1997

10 Hauberg S, Freifeld O, Black M J. A geometric take on metric learning. In: Proceedings of Conference and Workshop on Neural Information Processing Systems, 2012. 2024–2032
11 Xing E P, Ng A Y, Jordan M I, et al. Distance metric learning with application to clustering with side-information. In: Proceedings of Conference and Workshop on Neural Information Processing Systems, 2002. 15: 505–512
12 Davis J V, Kulis B, Jain P, et al. Information theoretic metric learning. In: Proceedings of the 24th International Conference on Machine Learning, 2007. 209–216
13 Wang J, Kalousisand A, Woznica A. Parameter local metric learning for nearest neighbor classification. In: Proceedings of Conference and Workshop on Neural Information Processing Systems, 2012
14 Saxena S, Verbeek J. Coordinated local metric learning. In: Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 2015. 369–377
15 Kim K I, Tompkin J, Theobalt C. Curvature-aware regularization on riemannian sub-manifolds. In: Proceedings of IEEE International Conference on Computer Vision, 2013. 881–888
16 Hamilton R S. Three-manifolds with positive Ricci curvature. J Differ Geom, 1982, 17: 255–306
17 Xu W, Hancock E R, Wilson R C. Rectifying non-Euclidean similarity data using Ricci flow embedding. In: Proceedings of International Conference on Pattern Recognition, 2010. 3324–3327
18 Xu W, Hancock E R, Wilson R C. Ricci flow embedding for rectifying non-Euclidean dissimilarity data. Pattern Recognition, 2014, 47: 3709–3725
19 Li Y Y, Lu R Q. Applying Ricci flow to high dimensional manifold learning. Sci China Inf Sci, 2019, 62: 192101
20 Li Y Y, Lu R Q. Riemannian metric learning based on curvature flow. In: Proceedings of the 24th International Conference on Pattern Recognition, 2018
21 Guarrera D T, Johnson N G, Wolfe H F. The taylor expansion of a Riemannian metric. 2002. http://studylib.net/doc/13872779/the-taylor-expansion-of-ariemannian-metric
22 Jain P, Kulis B, Dhillon I S, et al. Online metric learning and fast similarity search. In: Proceedings of the 22nd Annual Conference on Neural Information Processing Systems, Vancouver, 2008. 761–768
23 Shalev-Shwartz S. Online learning and online convex optimization. Machine Learn, 2004, 56: 209–239
24 Mei J, Liu M, Karimi H R, et al. LogDet divergence based metric learning using triplet labels. In: Proceedings of ICML Workshop Divergence Learning, 2013. 1–9
25 Li Y Y. Curvature-aware manifold learning. Pattern Recogn, 2018, 83: 273–286