Geodesic Distance Function Learning via Heat Flow on Vector Fields

Binbin Lin[†] BINBIN.LIN@ASU.EDU
Ji Yang[‡] YANGJI9181@GMAIL.COM
Xiaofei He[‡] XIAOFEIHE@CAD.ZJU.EDU.CN
Jieping Ye[†] JIEPING.YE@ASU.EDU
[†]Center for Evolutionary Medicine and Informatics, Arizona State University, Tempe, AZ 85287, USA
[‡]State Key Lab of CAD&CG, College of Computer Science, Zhejiang University Hangzhou 310058, China

Abstract

Learning a distance function or metric on a given data manifold is of great importance in machine learning and pattern recognition. Many of the previous works first embed the manifold to Euclidean space and then learn the distance function. However, such a scheme might not faithfully preserve the distance function if the original manifold is not Euclidean. In this paper, we propose to learn the distance function directly on the manifold without embedding. We first provide a theoretical characterization of the distance function by its gradient field. Based on our theoretical analysis, we propose to first learn the gradient field of the distance function and then learn the distance function itself. Specifically, we set the gradient field of a local distance function as an initial vector field. Then we transport it to the whole manifold via heat flow on vector fields. Finally, the geodesic distance function can be obtained by requiring its gradient field to be close to the normalized vector field. Experimental results on both synthetic and real data demonstrate the effectiveness of our proposed algorithm.

1. Introduction

Learning a distance function or metric on a given data manifold is of great importance in machine learning and pattern recognition. The goal of *distance metric learning* on the manifold is to find a desired distance function $d(x, y)$ such that it provides a natural measure of the *similarity* between two data points x and y on the manifold. It has been applied widely in many problems, such as information retrieval (McFee & Lanckriet, 2010), classification and clustering (Xing et al., 2002). Depending on whether there is label information available, metric learning methods can be classified into two categories: supervised and unsupervised. In supervised learning, one often assumes that data points with the same label should have small distance, while data points with different labels should have large distance (Xing et al., 2002; Weinberger et al., 2006; Jin et al., 2009). In this paper, we consider the problem of unsupervised distance metric learning.

Unsupervised manifold learning can be viewed as an alternative way of distance metric learning. It aims to find a map F from the original high dimensional space to a lower dimensional Euclidean space such that the mapped Euclidean distance $d(F(x), F(y))$ preserves the original distance $d(x, y)$. The classical Principal Component Analysis (PCA, Jolliffe 1989) can be considered as linear manifold learning method in which the map F is linear. The learned Euclidean distance after linear mapping is also referred to as Mahalanobis distance. Note that when the manifold is nonlinear, the Mahalanobis distance may fail to faithfully preserve the original distance.

The typical nonlinear manifold learning approaches include Isomap (Tenenbaum et al., 2000), Locally Linear Embedding (LLE, Roweis & Saul 2000), Laplacian Eigenmaps (LE, Belkin & Niyogi 2001), Hessian Eigenmaps (HLLE, Donoho & Grimes 2003), Maximum Variance Unfolding (MVU, Weinberger et al. 2004) and Diffusion Maps (Coifman & Lafon, 2006). Both Isomap and HLLE try to preserve the original geodesic distance on the data manifold. Diffusion maps try to preserve diffusion distance on the mani-

Proceedings of the 31[st] *International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32.

fold which reflects the connectivity of data. Coifman and Lafon (Coifman & Lafon, 2006) also showed that both LLE and LE belong to the diffusion map framework which preserves the local structure of the manifold. MVU is proposed to learn a kernel eigenmap that preserves pairwise distances on the manifold. One problem of the existing manifold learning approaches is that there may not exist a distance preserving map F such that $d(F(x), F(y)) = d(x, y)$ holds since the geometry and topology of the original manifold may be quite different from the Euclidean space. For example, there does not exist a distance preserving map between a sphere S^2 and a 2-dimensional plane.

In this paper, we assume the data lies approximately on a low-dimensional manifold embedded in Euclidean space. Our aim is to approximate the geodesic distance function on this manifold. The geodesic distance is a fundamental intrinsic distance on the manifold and many useful distances (e.g., the diffusion distance) are variations of the geodesic distance. There are several ways to characterize the geodesic distance due to its various definitions and properties. The most intuitive and direct characterization of the geodesic distance is by definition that it is the shortest path distance between two points (e.g., Tenenbaum et al. 2000). However, it is well known that computing pairwise shortest path distance is time consuming and it cannot handle the case when the manifold is not geodesically convex (Donoho & Grimes, 2003). A more convincing and efficient way to characterize the geodesic distance function is using partial differential equations (PDE). Mémoli *et al.* (Mémoli & Sapiro, 2001) proposes an iterated algorithm for solving the Hamilton-Jacobi equation $\|\nabla r\| = 1$ (Mantegazza & Mennucci, 2003), where ∇r represents the gradient field of the distance function. However, the fast marching part requires a grid of the same dimension as the ambient space which is impractical when the ambient dimension is very high.

Note that the tangent space dimension is equal to the manifold dimension (Lee, 2003) which is usually much smaller than the ambient dimension. One possible way to reduce the complexity of representing the gradient field ∇r is to use the local tangent space coordinates rather than the ambient space coordinates. Inspired by recent work on vector fields (Singer & Wu 2012, Lin et al. 2011, Lin et al. 2013) and heat flow on scalar fields (Crane et al. 2013), we propose to learn the geodesic distance function via the characterization of its gradient field and heat flow on vector fields. Specifically, we study the geodesic distance function $d(p, x)$ for a given base point p. Our theoretical analysis shows that if a function $r_p(x)$ is a local distance function around p, and its gradient field ∇r_p has unit norm or ∇r_p is parallel along geodesics passing through p, then $r_p(x)$ is the unique geodesic distance function $d(p, x)$. Based on our theoretical analysis, we set the gradient field of a local distance function around a given point as an initial vector field. Then we transport the initial local vector field to the whole manifold via heat flow on vector fields. By asymptotic analysis of the heat kernel, we show that the learned vector field is approximately parallel to the gradient field of the distance function at each point. Thus, the geodesic distance function can be obtained by requiring its gradient field to be close to the normalized vector field. The corresponding optimization problem involves sparse linear systems which can be solved efficiently. Moreover, the sparse linear systems can be easily extended to matrix form to learn the complete distance function $d(\cdot, \cdot)$. Both synthetic and real data experiments demonstrate the effectiveness of our proposed algorithm.

2. Characterization of Distance Functions using Gradient Fields

Let $(\mathcal{M}, g)$ be a d-dimensional Riemannian manifold, where g is a Riemannian *metric tensor* on $\mathcal{M}$. The goal of *distance metric learning* on the manifold is to find a desired distance function $d(x, y)$ such that it provides a natural measure for the *similarity* between two data points x and y on the manifold. In this paper, we study a fundamental intrinsic distance function[1] - the geodesic distance function. Similar to many geometry textbooks (e.g., Jost 2008; Petersen 1998), we call it the distance function. In the following, we will briefly introduce the most relevant concepts.

We first show how to assign a metric structure on the manifold. For each point p on the manifold, a Riemannian metric tensor g at p is an inner product g_p on each of the tangent space $T_p\mathcal{M}$ of $\mathcal{M}$. We define the norm of a tangent vector $v \in T_p\mathcal{M}$ as $\|v\| = \sqrt{g_p(v, v)}$. Let $[a, b]$ be a closed interval in $\mathbb{R}$, and $\gamma : [a, b] \to \mathcal{M}$ be a smooth curve. The *length* of γ can then be defined as $l(\gamma) := \int_a^b \|\frac{d\gamma}{dt}(t)\| dt$. The *distance* between two points p, q on the manifold $\mathcal{M}$ can be defined as:

$$d(p, q) := \inf\{l(\gamma) : \gamma : [a, b] \to \mathcal{M} \text{ piecewise smooth}, \\ \gamma(a) = p \text{ and } \gamma(b) = q\}.$$

We call $d(\cdot, \cdot)$ the *distance function* and it satisfies the usual axioms of a metric, i.e., positivity, symmetry and triangle inequality (Jost, 2008). We study the distance function $d(p, \cdot)$ when p is given.

Definition 2.1 (Distance function based at a point)**.**

[1] A distance function $d(\cdot, \cdot)$ defined by its Riemannian metric g is often called an intrinsic distance function.

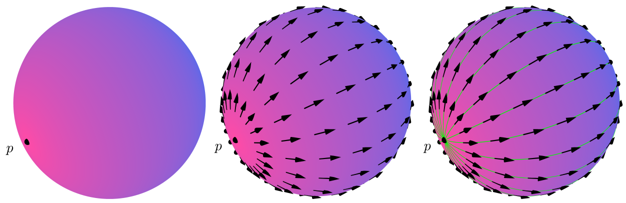


Figure 1. (a) shows the distance function $r_p(x)$ on the sphere, where the base point p is marked in black. Different colors indicate different distance values. (b) shows the gradient field ∇r_p. (c) shows the geodesics passing through p which are denoted by the green lines.

Let $\mathcal{M}$ be a Riemannian manifold, and let p be a point on the manifold $\mathcal{M}$. A distance function *on $\mathcal{M}$ based at p is defined as $r_p(x) = d(p,x)$. For simplicity, we might write $r(\cdot)$ instead of $r_p(\cdot)$.*

Definition 2.2 (Geodesic, Petersen 1998)**.** *Let $\gamma : [a,b] \to \mathcal{M}, t \mapsto \gamma(t)$ be a smooth curve. γ is called a* geodesic *if $\gamma'(t)$ is parallel along γ, i.e., $\nabla_{\gamma'(t)}\gamma'(t) = 0$ for all $t \in [a,b]$.*

Here ∇ is the covariant derivative on the manifold which measures the change of vector fields. A geodesic can be viewed as a *curved straight line* on the curved manifold. The geodesics and the distance function is related as follows:

Theorem 2.1 (Petersen 1998)**.** *If γ is a local minimum for $\inf l(\gamma)$ with fixed end points, then γ is a geodesic.*

In the following, we characterize the distance function $r_p(x)$ by using its gradient field ∇r. A vector field X on the manifold is called a *gradient field* if there exists a function f on the manifold such that $X = \nabla f$ holds. Therefore, gradient fields are one kind of vector fields. Interestingly, we can precisely characterize the distance function based at a point by its gradient field. For simplicity, let ∂_r denote the gradient field ∇r. Then we have the following result.

Theorem 2.2. *Let $\mathcal{M}$ be a complete manifold. A continuous function $r : \mathcal{M} \to \mathbb{R}$ is a distance function on $\mathcal{M}$ based at p if and only if (a) $r(x) = \|\exp_p^{-1}(x)\|$ holds for a neighborhood of p; (b) $\nabla_{\partial r}\partial r = 0$ holds on the manifold $\mathcal{M}$ except for $p \cup \mathrm{Cut}(p)$.*

Here $\exp_p$ is the exponential map at p and $\mathrm{Cut}(p)$ is the cut locus of p. Condition (a) states that locally $r(x)$ is a Euclidean distance function in the exponential coordinates. Combining condition (b) which states that the integral curves of ∂_r are all geodesics, we assert that r is a global distance function. As can be seen from Fig. 1(c), the gradient field of the distance function is parallel along the geodesics passing through p. It might be worth noting that condition (a) cannot be replaced by a weaker condition $r(p) = 0$ which is often used in PDE. A simple counter-example would be the function $r_p(x) = x$ defined on $\mathcal{M} = \mathbb{R}$ with $p = 0$. $r_p(x)$ satisfies $r_p(0) = 0$ and $\nabla_{\partial_r}\partial_r = 0$ holds for all x. However, it is not a distance function since it does not satisfy the positivity condition.

The second order condition $\nabla_{\partial_r}\partial_r = 0$ can be replaced by a first order condition $\|\partial_r\| = 1$.

Theorem 2.3. *Let $\mathcal{M}$ be a complete manifold. A continuous function $r : \mathcal{M} \to \mathbb{R}$ is a distance function on $\mathcal{M}$ based at p if and only if (a) $r(x) = \|\exp_p^{-1}(x)\|$ holds for a neighborhood of p; (b) $\|\partial_r\| = 1$ holds on the manifold $\mathcal{M}$ except for $p \cup \mathrm{Cut}(p)$.*

A detailed proof of Theorems 2.2 and 2.3 can be found in the long version of this paper (Lin et al., 2014). We visualize the relationship among the distance function, the gradient field of the distance function and geodesics in Fig. 1. It can be seen from the figure that: (1) the gradient field of the distance function is parallel along geodesics passing through p; (2) the gradient field of the distance function has unit norm almost everywhere except for p and its cut locus which is the antipodal point of p.

3. Geodesic Distance Function Learning

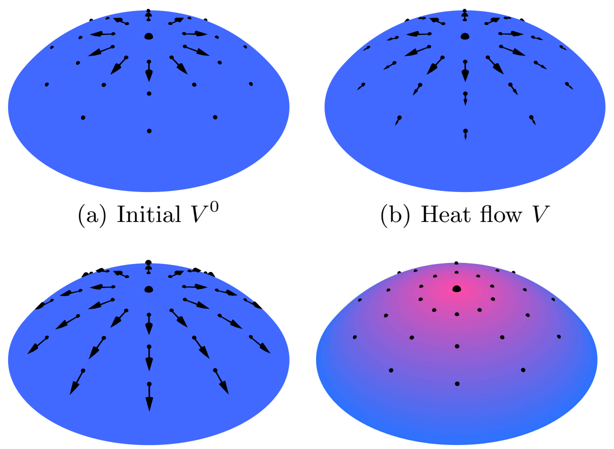
We show in the last section that the distance function can be characterized by its gradient field. Based on our theoretical analysis, we propose to first learn the gradient field of the distance function and then learn the distance function itself.

3.1. Geodesic Distance Learning

Let $(\mathcal{M}, g)$ be a d-dimensional Riemannian manifold embedded in a much higher dimensional Euclidean space $\mathbb{R}^m$, where g is a Riemannian metric tensor on $\mathcal{M}$. Given a point p on the manifold, we aim to learn the distance function $f_p(x) = d(p,x)$. Let $U_\epsilon := \{x : d(p,x) \le \epsilon\} \subset \mathcal{M}$ be a geodesic ball around p and let f^0 denote a local distance function on U. That is, $f^0(x) = d(p,x)$ if $p \in U$ and 0 otherwise. Let V^0 denote the gradient field of f^0, i.e., $V^0 = \nabla f^0$. Now we are ready to summarize our Geodesic Distance Learning (GDL) algorithm as follows:

- Learn a vector field V by transporting V^0 to the whole manifold using heat flow:

$$\min_V E(V) := \int_{\mathcal{M}} \|V - V^0\|^2 dx + t \int_{\mathcal{M}} \|\nabla V\|^2_{\mathrm{HS}} dx, \tag{1}$$



(a) Initial V^0 (b) Heat flow V

(c) Normalized $\hat{V}$ (d) Distance function f

Figure 2. Algorithm overview. The base point is on the top of the manifold. (a) shows the initial vector field V^0. (b) shows the vector field V after transporting V^0 to the whole manifold using heat flow on vector fields. (c) shows the normalized vector field $\hat{V}$ of V. (d) shows the final distance function learned via requiring its gradient field to be close to $\hat{V}$, where the red color indicates small distance function values and the blue color indicates large distance function values.

where $\|\cdot\|_{\mathrm{HS}}$ denotes the Hilbert-Schmidt tensor norm (Defant & Floret, 1993) and $t > 0$ is a parameter.

- Learn a normalized vector field $\hat{V}$ via normalizing V at each point x: set $\hat{V}_x = V_x/\|V_x\|$ when $x \neq p$ and set $\hat{V}_x = 0$ when $x = p$. Here V_x denotes the tangent vector at x of V.

- Learn the distance function f via solving the following equation:

$$\min_f \Phi(f) := \int_{\mathcal{M}} \|\nabla f - \hat{V}\|^2 dx, \ \text{ s.t.} f(p) = 0. \quad (2)$$

The above algorithmic steps are illustrated in Fig. 2.

The theoretical justification of the above algorithm is given in the appendix. Our analysis indicates that solving Eq. (1) is equivalent to transporting the initial vector field to the whole manifold via heat flow on vector fields. By asymptotic analysis of the heat kernel, the learned vector field is approximately parallel to the gradient field of the distance function at each point. Thus, the gradient field of the distance function can be obtained via normalization. Finally, the geodesic distance function can be obtained by requiring its gradient field to be close to the normalized vector field. Our analysis also indicate the factors of controlling the quality of the approximation. It mainly relies on two factors: the distance to the base point and the cut locus of the base point. If the data point is not in the cut locus of the base point, the smaller the distance between the data point and the base point is, the better the approximation would be. If the data point is in the cut locus, the approximation might fail since the vector field around the cut locus varies dramatically. Note that the measure of the cut locus is zero (Lin et al., 2014), thus the approximation would fail only in a zero measure set.

3.2. Implementation

Given n data points $x_i, i = 1, \ldots, n$, on the d-dimensional manifold $\mathcal{M}$ where $\mathcal{M}$ is embedded in the high dimensional Euclidean space $\mathbb{R}^m$. Let x_q denote the base point. We aim to learn the distance function $f : \mathcal{M} \to \mathbb{R}$ based at x_q, i.e., $f(x_i) = d(x_q, x_i)$, $i = 1, \ldots, n$.

We first construct an undirected nearest neighbour graph by either ϵ-neighbourhood or k nearest neighbours. It might be worth noting for a k-nn graph that the degree of a vertex will typically be larger than k since k nearest neighbour relationships are not symmetrical. Let $x_i \sim x_j$ denote that x_i and x_j are neighbors. Let w_{ij} denote the weight which can be approximated by the heat kernel weight or the simple 0-1 weight. For each point x_i, we estimate its tangent space $T_{x_i}\mathcal{M}$ by performing PCA on its neighborhood. Before performing PCA, we mean-shift the neighbor vectors using their true mean. Let $T_i \in \mathbb{R}^{m\times d}$ denote the matrix whose columns are constituted by the d principal components. Let V be a vector field on the manifold. For each point x_i, let V_{x_i} denote the tangent vector at x_i. Recall from the definition of the tangent vector that each tangent vector V_{x_i} should be in the corresponding tangent space $T_{x_i}\mathcal{M}$, we can represent V_{x_i} as $V_{x_i} = T_i v_i$, where $v_i \in \mathbb{R}^d$. We will abuse the notation f to denote the vector $f = (f(x_1), \ldots, f(x_n))^T \in \mathbb{R}^n$ and use V to denote the vector $V = \left(v_1{}^T, \ldots, v_n{}^T\right)^T \in \mathbb{R}^{dn}$. We propose to first learn V and then learn f.

Set an initial vector field V^0 as follows:

$$v_j^0 = \begin{cases} \dfrac{T_j^T(x_j - x_q)}{\|T_j T_j^T(x_j - x_q)\|}, & \text{if } j \sim q \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Note that the vector $T_j^T(x_j - x_q)/\|T_j T_j^T(x_j - x_q)\|$ is a unit vector at x_j pointing outward from the base point x_q (please see Fig. 2(a)). Following (Lin et al., 2011), the discrete form of our objective functions can be given as follows:

$$\begin{aligned} E(V) &= V^TV - 2{V^0}^TV + {V^0}^TV^0 + tV^TBV, \\ \Phi(f) &= 2f^TLf + \hat{V}^TG\hat{V} - 2\hat{V}^TCf, \end{aligned} \quad (4)$$

Algorithm 1 GDL (Geodesic Distance Learning)

Require: Data sample $X = (x_1, \dots, x_n) \in \mathbb{R}^{m\times n}$ and a base point x_q, $1 \le q \le n$.
Ensure: $f = (f_1, \dots, f_n) \in \mathbb{R}^n$
  for $i = 1$ to n **do**
    Compute the tangent space coordinates $T_i \in \mathbb{R}^{m\times d}$ by using PCA
  end for
  Set an initial vector field V^0 via Eq. (3) and construct sparse block matrices B and C
  Solve $(I + tB)V = V^0$ to obtain V
  Normalize each vector in V to obtain $\hat{V}$
  Solve $2Lf = C^T\hat{V}$ to obtain f
  return f

where L is the graph Laplacian matrix (Chung, 1997), B is a $dn\times dn$ block matrix, G is a $dn\times dn$ block diagonal matrix and C is a $dn\times n$ block matrix. Let B_{ij} ($i \neq j$) denote the ij-th $d\times d$ block, G_{ii} denote the i-th $d\times d$ diagonal block of G, and C_i denote the i-th $d\times n$ block of C. We have: $B_{ii} = \sum_{j\sim i} w_{ij}(Q_{ij}Q_{ij}^T + I)$, $B_{ij} = -2w_{ij}Q_{ij}$, $G_{ii} = \sum_{j\sim i} w_{ij}T_i^T(x_j - x_i)(x_j - x_i)^T T_i$, and $C_i = \sum_{j\sim i} w_{ij}T_i^T(x_j - x_i)s_{ij}^T$, where $Q_{ij} = T_i^T T_j$ and $s_{ij} \in \mathbb{R}^n$ is a selection vector of all zero elements except for the i-th element being -1 and the j-th element being 1. The matrix Q_{ij} transports from the tangent space $T_{x_j}\mathcal{M}$ to $T_{x_i}\mathcal{M}$ which approximates the parallel transport (Lin et al., 2014) from x_j to x_i. It might be worth noting that one can also approximate the parallel transport by solving a singular value decomposition problem (Singer & Wu, 2012). The block matrix B provides a discrete approximation of the connection Laplacian operator, which is a symmetric and positive semi-definite matrix.

Now we give our algorithm in the discrete setting. By taking derivatives of $E(V)$ with respect to V, V can be obtained via the following sparse linear system:

$$(I + tB)V = V^0. \tag{5}$$

Then we learn a normalized vector field $\hat{V}$ via normalizing V at each point: $\hat{v}_i = v_i/\|v_i\|$ if $i \neq q$ and $\hat{v}_i = 0$ if $i = q$. The final distance function can be obtained via taking derivatives of $\Phi(f)$ with respect to f:

$$2Lf = C^T\hat{V}, \tag{6}$$

where we restrict $f_q = 0$ when solving Eq. (6). A direct way is to plug the constraint $f_q = 0$ into Eq. (6). It is equivalent to removing the q-th column of L and the q-th element of f on the left hand side of Eq. (6). For each point x_q, we have corresponding vectors V, V^0,$\hat{V}$ and$f$. If x_q varies, we can write V, V^0, $\hat{V}$ and f in matrix form where each column is a vector field or a distance function. Then the complete distance function $d(\cdot,\cdot)$ can be obtained via solving the corresponding matrix form linear systems of Eq. (5) and Eq. (6). We summarize our algorithm in Algorithm 1.

3.3. Computation Complexity Analysis

The computational complexity of our proposed Geodesic Distance Learning (GDL) algorithm is dominated by three parts: searching for k-nearest neighbors, computing local tangent spaces, computing Q_{ij} and solving the sparse linear system Eq. (5). For the k nearest neighbor search, the complexity is $O((m+k)n^2)$, where $O(mn^2)$ is the complexity of computing the distance between any two data points, and $O(kn^2)$ is the complexity of finding the k nearest neighbors for all the data points. The complexity for local PCA is $O(mk^2)$. Therefore, the complexity for computing the local tangent space for all the data points is $O(mnk^2)$. Note that the matrix B is not a dense matrix but a very sparse block matrix with at most kn non-zero $d \times d$ blocks. Therefore the computation complexity of computing all Q_{ij}'s is $O(knmd^2)$. We use LSQR package[2] to solve Eq. (5). It has a complexity of $O(Iknd^2)$, where I is the number of iterations. In summary, the overall computational cost for one base point is $O((m+k)n^2 + mndk + kmnd^2 + Iknd^2)$. For p base points, the extra cost is to solve Eq. (5) by adding $p-1$ columns which has a complexity of $O(pIknd^2)$. Empirically, the manifold dimension d and the number of nearest neighbors k are usually much smaller than the ambient dimension m and the number of data points n. So the total computational cost for p base points could be $O(mn^2 + pIn)$. There are several ways to further reduce the computational complexity. One way is to select anchor points and construct the graph using these anchor points. Another possible way is to learn the distance functions of nearby points simultaneously.

3.4. Related Work and Discussion

Our approach is based on the idea of vector field regularization which is similar to Vector Diffusion Maps (VDM, Singer & Wu 2012). Both methods employ vector fields to discover the geometry of the manifold. However, VDM and our approach differ in several key aspects: Firstly, they solve different problems. VDM tries to preserve the vector diffusion distance by dimensionality reduction while we try to learn the geodesic distance function directly on the manifold. It is worth noting that the vector diffusion distance is a

[2] http://www.stanford.edu/group/SOL/software/lsqr.html

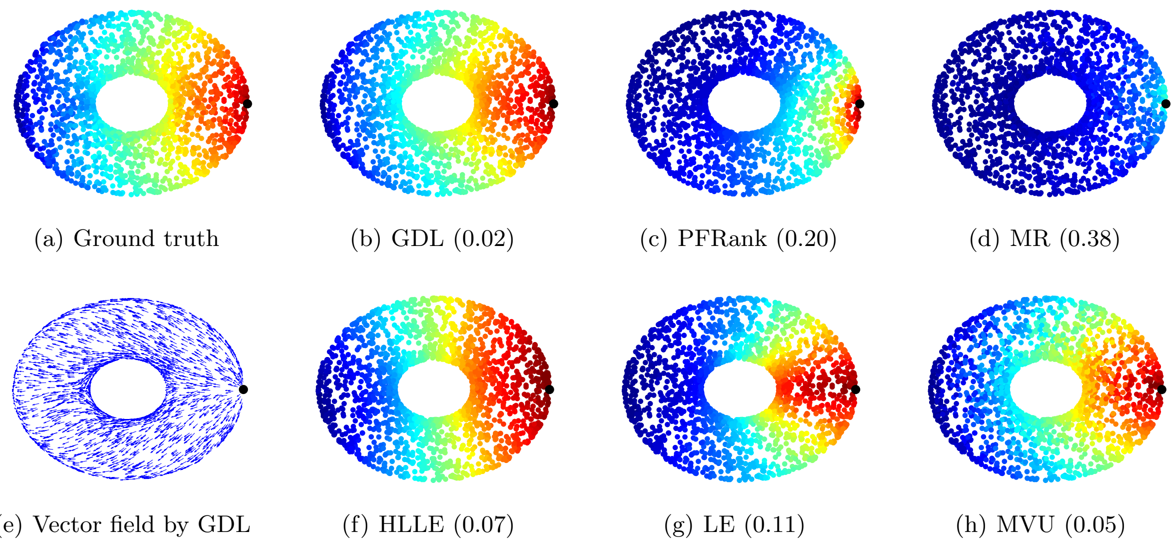


Figure 3. The base point is marked in black. (a) shows the ground truth geodesic distance function. (e) shows the vector field of the distance function learned by GDL. (b)-(d) and (f)-(h) visualize the distance functions learned by different algorithms. Different colors indicates different distance values. The number in the brackets measures the difference between the learned distance function and the ground truth.

variation of the geodesic distance. Secondly, they use different approximation methods. VDM approximates the parallel transport by learning an orthogonal transformation and we simply use projection adopted from (Lin et al., 2011). GDL can also be regarded as a generalization of the heat method (Crane et al., 2013) on scalar fields. Both methods employ heat flow to obtain the gradient field of distance function. The algorithm proposed in (Crane et al., 2013) first learns a scalar field by heat flow on scalar fields and then learns the desired vector field by evaluating the gradient field of the obtained scalar field. Our method tries to learn the desired vector field directly by heat flow on vector fields. Note that the scalar field is zero order and the vector field is first order. It is expected that the first order approximation of the vector field might be more effective for high dimensional data.

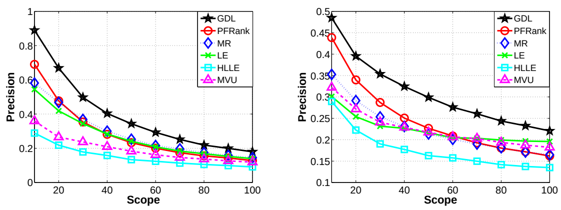
4. Experiments

In this section, we empirically evaluate the effectiveness of our proposed Geodesic Distance Learning (GDL) algorithm in comparison with three representative distance metric learning algorithms: Laplacian Eigenmaps (LE, Belkin & Niyogi 2001), Maximum Variance Unfolding (MVU, Weinberger et al. 2004) and Hessian Eigenmaps (HLLE, Donoho & Grimes 2003) as well as two state-of-art ranking algorithms: Manifold Ranking (MR, Zhou et al. 2003) and Parallel Field Rank (PFRank, Ji et al. 2012). As LE, MVU and HLLE cannot directly obtain the distance function, we compute the embedding first and then compute the Euclidean distance between data points in the embedded Euclidean space.

We empirically set $t = 1$ for GDL in all experiments as GDL performs very stable when t varies. The dimension of the manifold d is set to 2 in the synthetic example. For real data, we perform cross-validation to choose d. Specifically, $d = 9$ for the CMU PIE data set and $d = 2$ for the Corel data set. We use the same nearest neighbor graph for all six algorithms. The number of nearest neighbors is set to 16 on both synthetic and real data sets and the weight is the simple $0-1$ weight.

4.1. Geodesic Distance Learning

A simple synthetic example is given in Fig. 3. We randomly sample 2000 data points from a torus. It is a 2-dimensional manifold in the 3-dimensional Euclidean space. The base point is marked by the black dot on the right side of the torus. Fig. 3(a) shows the ground truth geodesic distance function which is computed by the shortest path distance. Fig. 3(b)-(d) and (f)-(h) visualize the distance functions learned by different algorithms respectively. To better evaluate the results, we compute the error by using the equation $\frac{1}{n}\sum_{i=1}^{n}|f(x_i)-d(x_q,x_i)|$, where $f(x_i)$ represents the learned distance and $\{d(x_q,x_i)\}$ represents the ground truth distance. To remove the effect of scale, $\{f(x_i)\}$ and $\{d(x_q,x_i)\}$ are rescaled to the range $[0,1]$. As can be seen from Fig. 3, GDL better preserves the distance



(a) Precision-scope curves on PIE (b) Precision-scope curves on Corel

Figure 4. Precision-scope curves.

metric on the torus. Although MVU is comparable to GDL in this example, GDL is approximately thirty times faster than MVU. It might be worth noting that both MR and PFRank achieve poor performance since they are deigned to preserve the ranking order but not the distance.

4.2. Image Retrieval

In this section, we apply our GDL algorithm to the image retrieval problem in real world image databases. Two real world data sets are used in our experiments. The first one is from the CMU PIE face database (Sim et al., 2003), which contains 32×32 cropped face images of 68 persons. We choose the frontal pose (C27) with varying lighting conditions, which leaves us 42 images per person. The second data set contains 5,000 images of 50 semantic categories, from the Corel database. Each image is extracted to be a 297-dimensional feature vector. Both of the two image data sets we use have category labels. For each data set, we randomly choose 10 images from each category as queries, and average the retrieval performance over all the queries.

We use precision, recall, and Mean Average Precision (MAP, Manning et al. 2008) to evaluate the retrieval results of different algorithms. Precision is defined as the number of relevant presented images divided by the number of presented images. Recall is defined as the number of relevant presented images divided by the total number of relevant images in our database. Given a query, let r_i be the relevance score of the image ranked at position i, where $r_i = 1$ if the image is relevant to the query and $r_i = 0$ otherwise. Then we can compute the Average Precision (AP):

$$\mathrm{AP} = \frac{\sum_i r_i \times \mathrm{Precision@}i}{\#\text{ of relevant images}}. \tag{7}$$

MAP is the average of AP over all the queries.

Fig. 4(a) and Fig. 4(b) show the average precision-scope curves of various methods on the two data sets, respectively. The scope means the number of top-ranked images returned to the user. The precision-scope curves describe the precision with various scopes, and therefore provide an overall performance evaluation of the algorithms. As can be seen from Fig. 4(a) and Fig. 4(b), our proposed GDL algorithm outperforms all the other algorithms. We also present the recall and MAP scores of different algorithms on the two data sets in Table 1 and Table 2, respectively. MAP provides a single figure measure of quality across all the recall levels. Our GDL achieves the highest MAP, indicating reliable performance over the entire ranking list. We also performed comprehensive t-test with 99% confidence level. The improvements of GDL compared to PFRank and other algorithms are significant with most of the p-values less than 10^{-3}, including those in Fig. 4(a), Fig. 4(b), Table 1 and Table 2. These results indicate that learning the distance function directly on the manifold might be better than learning the distance function after embedding.

Table 1. Recall and MAP on the PIE data set.

Recall	@10	@20	@50	MAP
GDL	**0.457**	**0.618**	**0.783**	**0.698**
PFRank	0.443	0.585	0.713	0.596
MR	0.323	0.524	0.698	0.507
LE	0.301	0.452	0.643	0.479
HLLE	0.162	0.234	0.357	0.245
MVU	0.228	0.333	0.565	0.338

Table 2. Recall and MAP on the Corel data set.

Recall	@10	@20	@50	MAP
GDL	**0.134**	**0.195**	**0.330**	**0.340**
PFRank	0.124	0.173	0.268	0.266
MR	0.098	0.148	0.250	0.263
LE	0.092	0.127	0.233	0.268
HLLE	0.099	0.134	0.213	0.220
MVU	0.099	0.137	0.239	0.272

5. Conclusion

In this paper, we study the geodesic distance from the vector field perspective. We provide theoretical analysis to precisely characterize the geodesic distance function and propose a novel heat flow on vector fields approach to learn it. Our experimental results on synthetic and real data demonstrate the effectiveness of the proposed method. The future work includes developing the machine learning theory and designing efficient algorithms using heat flow on vector fields as well as other general partial differential equations.

Acknowledgments

This work was supported in part by NIH (LM010730), NSF (IIS-0953662, CCF-1025177), National Basic Research Program of China (973 Program) under Grant 2012CB316400, National Natural Science Foundation of China under Grant 61125203 and Grant 61233011, National Program for Special Support of Top-Notch Young Professionals.

Appendix. Justification

We first show that solving Eq. (1) is equivalent to solving the heat equation on vector fields. According to the Bochner technique (Petersen, 1998), with appropriate boundary conditions we have $\int_{\mathcal{M}} \|\nabla V\|_{\mathrm{HS}}^2 dx = \int_{\mathcal{M}} g(V, \nabla^*\nabla V)dx$, where $\nabla^*\nabla$ is the *connection Laplacian* operator. Define the inner product $(\cdot,\cdot)$ on the space of vector fields as $(X,Y) = \int_{\mathcal{M}} g(X,Y)dx$. Then we can rewrite $E(V)$ as $E(V) = (V - V^0, V - V^0) + t(V, \nabla^*\nabla V)$. The necessary condition of $E(V)$ to have an extremum at V is that the functional derivative $\delta E(V)/\delta V = 0$ (Abraham et al., 1988). Using the calculus rules of the functional derivative and the fact $\nabla^*\nabla$ is a self-adjoint operator, we have $\delta E(V)/\delta V = 2V - 2V^0 + 2t\nabla^*\nabla V$. A detailed derivation can be found in the long version of this paper(Lin et al., 2014). Since $\nabla^*\nabla$ is also a positive semi-definite operator, the optimal V is then given by:

$$V = (I + t\nabla^*\nabla)^{-1}V^0, \tag{8}$$

where I is the identity operator on vector fields. Let $X(t)$ be a vector field valued function. That is, for each t, $X(t)$ is a vector field on the manifold. Given an initial vector field $X(t)|_{t=0} = X_0$, the heat equation on vector fields (Berline et al., 2004) is given by $\frac{\partial X(t)}{\partial t} + \nabla^*\nabla X(t) = 0$. When t is small, we can discrete it as follows: $\frac{X(t)-X_0}{t} + \nabla^*\nabla X(t) = 0$. Then $X(t)$ can be solved as

$$X(t) = (I + t\nabla^*\nabla)^{-1}X_0. \tag{9}$$

If we set $X_0 = V^0$, then Eq. (9) is exactly the same as Eq. (8). Therefore when t is small, solving Eq (1) is essentially solving the heat equation on vector fields.

Next we analyze the asymptotic behavior of $X(t)$ and show that the heat equation transfers the initial vector field primarily along geodesics. Let $x, y \in \mathcal{M}$, then $X(t)$ can be obtained via the heat kernel as $X(t)(x) = \int_{\mathcal{M}} k(t,x,y)X_0(y)dy$, where $k(t,x,y)$ is the heat kernel for the connection Laplacian. It is well known for small t, we have the asymptotic expansion of the heat kernel (Berline et al., 2004):

$$k(t,x,y) \approx (\frac{1}{4\pi t})^{\frac{d}{2}} e^{-d(x,y)^2/4t}\tau(x,y), \tag{10}$$

where $d(\cdot,\cdot)$ is the distance function, $\tau : T_y\mathcal{M} \to T_x\mathcal{M}$ is the parallel transport along the geodesic connecting x and y.

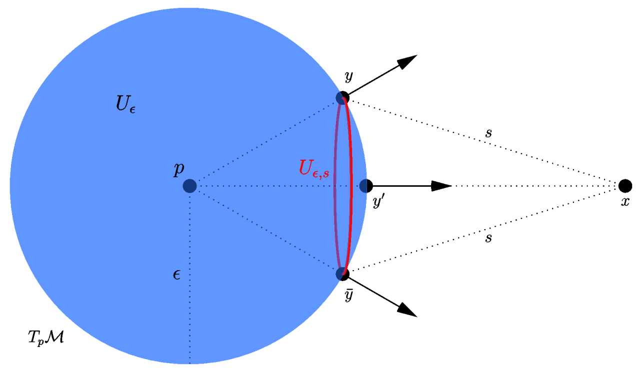


Figure 5. Illustration of the heat flow on vector fields.

Now we consider $X_0 = V^0$. By construction $X_0(y) = 0$ if $y \notin U_\epsilon$. Then the vector $X(t)(x) = \int_{U_\epsilon} e^{-d(x,y)^2/4t}\tau(x,y)X_0(y)dy$ up to a scale. To analyze what $X(t)(x)$ is, we first map the manifold $\mathcal{M}$ to the tangent space $T_p\mathcal{M}$ by using $\exp_p^{-1}$. Then U_ϵ becomes a ball in $T_p\mathcal{M}$; please see Fig. 5. In the following we will still use x and U_ϵ to represent $\exp_p^{-1}(x)$ and $\exp_p^{-1}(U_\epsilon)$ for simplicity of notation. Given any point $x \in T_p\mathcal{M}$, we can decompose the ball U_ϵ as $U_\epsilon = \cup_{\epsilon',s}U_{\epsilon',s}$ where $U_{\epsilon',s} := \{y|d(p,y) = \epsilon', d(x,y) = s\}, \epsilon' \le \epsilon$ and $0 \le s \le \infty$. Then each section $U_{\epsilon',s}$ is a sphere centered at some point lying on the line segment connecting p and x. Therefore $U_{\epsilon',s}$ is symmetric with respect to the vector $x - p$. For any $y \in U_{\epsilon',s}$, there is a unique reflection point $\bar{y}$ such that $\tau(x,y)X_0(y) + \tau(x,\bar{y})X_0(\bar{y})$ is parallel to $\tau(x,y')X_0(y')$ where $y' = \arg\min_{y\in U_\epsilon} d(x,y)$. Note that the weight $e^{-d(x,y)^2/4t}$ is the same on the section $U_{\epsilon',s}$. We conclude that $\int_{U_{\epsilon',s}} e^{-d(x,y)^2/4t}\tau(x,y)X_0(y)dy$ is parallel to $\tau(x,y')X_0(y')$. Since $\int_{U_\epsilon} = \int_{\epsilon'}\int_s\int_{U_{\epsilon',s}}$, $X_0(x) \approx \int_{U_\epsilon} e^{-d(x,y)^2/4t}\tau(x,y)X_0(y)dy$ is parallel to $\tau(x,y')X_0(y')$. In other words, the vector field flows primarily along geodesics. Therefore given an initial distance vector field around the base point, solving the heat equation will get a vector field which is approximately parallel to the gradient field of the distance function at each point. We can further normalize the vector field at each point to obtain the gradient field of the distance function. From this heat equation point of view, it also provides guidance of the algorithm setting. Specifically, we should set the initial vector field uniformly around the base point and set a small t.

References

Abraham, R., Marsden, J. E., and Ratiu, T. *Manifolds, tensor analysis, and applications*, volume 75 of *Applied Mathematical Sciences*. Springer-Verlag, New York, second edition, 1988.

Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pp. 585–591. 2001.

Berline, N., Getzler, E., and Vergne, M. *Heat kernels and Dirac operators*. Springer-Verlag, 2004.

Chung, Fan R. K. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997.

Coifman, Ronald R. and Lafon, Stphane. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 – 30, 2006. Diffusion Maps and Wavelets.

Crane, Keenan, Weischedel, Clarisse, and Wardetzky, Max. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph.*, 32(5): 152:1–152:11, 2013.

Defant, A. and Floret, K. *Tensor Norms and Operator Ideals*. North-Holland Mathematics Studies, North-Holland, Amsterdam, 1993.

Donoho, D. L. and Grimes, C. E. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences of the United States of America*, 100(10):5591–5596, 2003.

Ji, Ming, Lin, Binbin, He, Xiaofei, Cai, Deng, and Han, Jiawei. Parallel field ranking. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pp. 723–731, 2012.

Jin, Rong, Wang, Shijun, and Zhou, Yang. Regularized distance metric learning:theory and algorithm. In *Advances in Neural Information Processing Systems 22*, pp. 862–870. 2009.

Jolliffe, I. T. *Principal Component Analysis*. Springer-Verlag, New York, 1989.

Jost, Jürgen. *Riemannian Geometry and Geometric Analysis (5. ed.)*. Springer, 2008. ISBN 978-3-540-77340-5.

Lee, J. M. *Introduction to Smooth Manifolds*. Springer Verlag, New York, 2nd edition, 2003.

Lin, Binbin, Zhang, Chiyuan, and He, Xiaofei. Semi-supervised regression via parallel field regularization. In *Advances in Neural Information Processing Systems 24*, pp. 433–441. 2011.

Lin, Binbin, He, Xiaofei, Zhang, Chiyuan, and Ji, Ming. Parallel vector field embedding. *Journal of Machine Learning Research*, 14:2945–2977, 2013.

Lin, Binbin, Yang, Ji, He, Xiaofei, and Ye, Jieping. Geodesic distance function learning via heat flows on vector fields. *CoRR*, abs/1405.0133, 2014.

Manning, Christopher D., Raghavan, Prabhakar, and Schtze, Hinrich. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

Mantegazza, Carlo and Mennucci, Andrea Carlo. Hamilton-jacobi equations and distance functions on riemannian manifolds. *Applied Mathematics and Optimization*, 47(1):1–26, 2003.

McFee, Brian and Lanckriet, Gert. Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 775–782, 2010.

Mémoli, Facundo and Sapiro, Guillermo. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *Journal of Computational Physics*, 173 (2):730 – 764, 2001.

Petersen, P. *Riemannian Geometry*. Springer, New York, 1998.

Roweis, S. and Saul, L. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Sim, T., Baker, S., and Bsat, M. The CMU pose, illuminlation, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12): 1615–1618, 2003.

Singer, A. and Wu, H.-T. Vector diffusion maps and the connection Laplacian. *Communications on Pure and Applied Mathematics*, 65(8):1067–1144, 2012.

Tenenbaum, J., de Silva, V., and Langford, J. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

Weinberger, Kilian, Blitzer, John, and Saul, Lawrence. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 18*, pp. 1473–1480. 2006.

Weinberger, Kilian Q., Sha, Fei, and Saul, Lawrence K. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the twenty-first international conference on Machine learning (ICML-04)*, ICML '04, pp. 839–846, 2004.

Xing, Eric P., Ng, Andrew Y., Jordan, Michael I., and Russell, Stuart J. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pp. 505–512, 2002.

Zhou, Dengyong, Weston, Jason, Gretton, Arthur, Bousquet, Olivier, and Schölkopf, Bernhard. Ranking on data manifolds. In *Advances in Neural Information Processing Systems 16*, pp. 169–176. 2003.