

Manifold Learning Algorithms and Their Mathematical Foundations

Ying Zheng

Department of Computer Science

Duke University

yuanqi@cs.duke.edu

Instructor: Pankaj Agarwal

December 8, 2008

Abstract

This is the final project report for CPS234¹. In this paper, we study several recently developed manifold learning algorithms or more specifically: Isomap, Laplacian Eigenmap and Diffusion Map. We motivate the importance of these algorithms through real applications such as dimensionality reduction, clustering and classification. We also give the basic mathematical justifications for these methods and see how they have connections with differential geometry, partial differential equations and spectral graph theory. In order to see the real power of these manifold learning methods, we also did several experiments in terms of dimensionality reduction both on synthetic data and real data. The experimental results show that these methods can work pretty well at least in our toy examples. Last, we summarize the ideas of manifold learning and concludes this article with some possible future directions.

1 Introduction

Manifold-based high dimensional data analysis have applications in many areas such as data mining and computer vision which require analysis on data sets with a number of features or dimensions. Managing a large number of features - some of which might be irrelevant or even misleading - is typically a burden to real applications. For example, suppose we have a collection of frames taken from a video of a person rotating his or her head. The dimensionality of the data set is equal to the number of pixels in a frame, which is generally very large. However, the images are actually governed by only a couple of degrees of freedom (such as the angle of rotation). Therefore, it is reasonable to expect that the real content of the images can be represented using much fewer dimensions. For another example, suppose we are to classify an email as spam or not spam. A typical approach to this problem would be to represent an email as a vector of counts of the words

¹CPS234, Computational Geometry, Fall 2008. Instructor: Pankaj Agarwal.

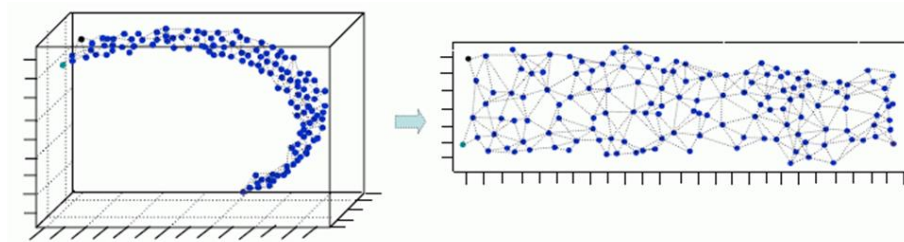


Figure 1: A example of embedding from 3-d space to 2-d space

appearing in the email. The dimensionality of this data can easily be in the hundreds, yet an effective dimensionality reduction technique may reveal that there are only a few exceptionally telling features, such as the word “Viagra”.

Manifold learning is a recent popular approach to nonlinear dimensionality reduction. Algorithms for this task are based on the idea that the dimensionality of many data sets is only artificially high; though each data point consists of perhaps thousands of features, it may be described as a function of only a few underlying parameters. That is, the data points are actually samples from a low-dimensional manifold that is embedded in a high-dimensional space. Manifold learning algorithms attempt to uncover these parameters in order to find a low-dimensional representation of the data. As an example, Figure 1 shows data points lying on a surface which is embedded in 3-dimensional space and its unrolling into 2-dimensional space. Notice that the structure of data points is preserved when mapped to 2-dimensional space.

In this paper, we focus mainly on three manifold learning methods: Isomap [TdSL00, BdSLT00], Laplacian Eigenmap [BN03, BN07] and Diffusion Map [NLCK05, LLL06]. All the three methods share some similarity with each other while the mathematical motivation and justifications for these methods are quite different. One of the central topic behind is the spectral graph theory [Chu] which links the eigenvectors and eigenvalues with various areas in mathematics such as graph theory, differential geometry and partial differential equations. A through discussion of these methods will require much deeper knowledge of high dimensional manifolds and differential geometry. As an alternative, we present the algorithm for these methods and give intuitive ideas behind the algorithms along with a brief description of their mathematical foundations.

Outline: The remainder of this article is organized as follows. Section 2 gives a thorough description of the algorithm of Isomap and its mathematical foundations. Section 3 will describe Laplacian method and its mathematical justifications. Section 4 describes diffusion map and its intuitive ideas behind. In Section 5 we present our experimental results using different manifold learning algorithms both on synthetic data and our toy examples in real application. Finally Section 6 summarizes the ideas of manifold learning and concludes this article with some possible future directions.

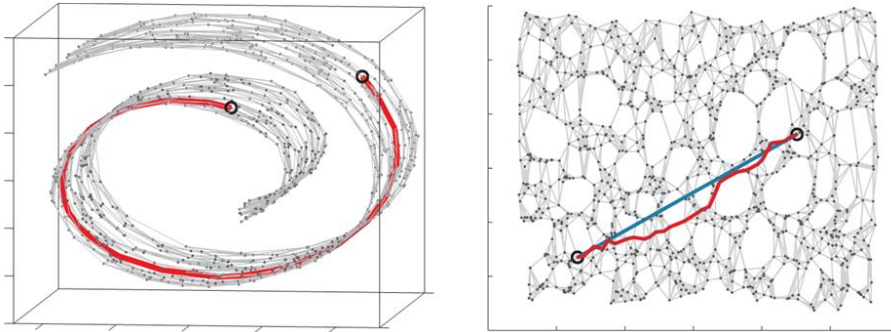


Figure 2: Illustration of the idea of Isomap

2 Isomap

Among all the manifold learning methods, Isomap [TdSL00, BdSLT00] - short for isometric feature mapping - is perhaps the most well-known and popular method. Its idea is simple and its implementation is straightforward. The main idea of Isomap is to approximate an N -dimensional manifold using n -dimensional Euclidean space (where n is far less than N) such that the geodesic distance between arbitrary pair of points in the original manifold is preserved in the transformed Euclidean space. See Figure 2 for illustration.

More formally speaking, given $x_1, x_2, \dots, x_k \in M$ lying in a N -dimensional manifold, we want to find $f : M \rightarrow \mathbb{R}^n, n \ll N$ s.t. $\forall x_i, x_j \in M, \|f(x_i) - f(x_j)\|_{Euclidean} = \|x_i - x_j\|_{geodesics}$, where $\|\cdot\|_{Euclidean}$ and $\|\cdot\|_{geodesics}$ means distance in Euclidean and geodesic sense respectively. The question is how to approximate the geodesic distance between arbitrary pair of points on the manifold M . The underlying assumption behind Isomap is that for points very close to each other on the manifold, we can approximate the geodesic distance by the Euclidean distance while for points which are far from each other on the manifold, we can approximate the geodesic distance by finding the shortest path linking these two points.

The theoretical guarantees for the shortest path as the approximation for the geodesic distance is presented in [BdSLT00]. In their work, the main result can be stated in the following theorem:

Theorem 1: Asymptotic Convergence Suppose that the data points are chosen randomly, with a certain density function α . Given $\lambda_1, \lambda_2, \mu > 0$, which can be arbitrarily small, then for α sufficiently large, the inequalities hold:

$$\mathbb{P} \left\{ 1 - \lambda_1 < \frac{\text{graph distance}}{\text{geodesic distance}} < 1 + \lambda_2 \right\} \geq 1 - \mu$$

The proof for theorem 1 is actually very lengthy. For simplicity we omit the proof for the convergence theorem and refer those readers who are interested in the rigorous proof to [BdSLT00]. As an alternative, we prove the following weak theorem which gives a rough bound for the shortest path on the graph as an approximation.

Theorem 2: Weak Bound Let ϵ and δ be positive with $4\delta < \epsilon$. Let $d_M(x, y)$ be the geodesic distance between point x and y and $d_S(x, y) \triangleq \min_P(d_M(x_0, x_1) + \dots + d_M(x_{p-1}, x_p))$ where P varies over all paths along the edges of the graph connecting $x(=x_0)$ and $y(=x_p)$. Suppose that the graph contains all edges xy for which $d_M(x, y) \leq \epsilon$ and for every point q in M there is a data point x_i for which $d_M(q, x_i) \leq \delta$. Then for all pairs of data points x, y we have:

$$d_M(x, y) \leq d_S(x, y) \leq \left(1 + \frac{4\delta}{\epsilon}\right)d_M(x, y)$$

Proof: If $d_M(x, y) \leq \epsilon - 2\delta$, then x and y are connected with an edge which we can use as our path. Otherwise we can decompose $d_M(x, y)$ as $d_M(x, y) = d_M(\gamma_0, \gamma_1) + d_M(\gamma_1, \gamma_2) + \dots + d_M(\gamma_{p-1}, \gamma_p)$, where $\gamma_0 = x, \gamma_p = y$ and $d_M(\gamma_i, \gamma_{i+1}) = \epsilon - 2\delta$ for $i = 1, 2, \dots, p-2$ and $\epsilon - 2\delta \geq d_M(\gamma_0, \gamma_1) = d_M(\gamma_{p-1}, \gamma_p) \geq (\epsilon - 2\delta)/2$. The assumption says that for each point γ_i there exists a point x_i which lies within the distance δ of γ_i .

We will show that the path $x \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_{p-1} \rightarrow y$ is the desired path satisfying the requirement. Notice that $d_M(x_i, x_{i+1}) \leq d_M(x_i, \gamma_i) + d_M(\gamma_i, \gamma_{i+1}) + d_M(\gamma_{i+1}, x_{i+1}) \leq \delta + \epsilon - 2\delta + \delta = \epsilon = d_M(\gamma_i, \gamma_{i+1})\epsilon/(\epsilon - 2\delta)$. Similarly we have: $d_M(x, x_1) \leq d_M(\gamma_0, \gamma_1)\epsilon/(\epsilon - 2\delta)$ and $d_M(x_{p-1}, y) \leq d_M(\gamma_{p-1}, \gamma_p)\epsilon/(\epsilon - 2\delta)$. Therefore, $d_M(x, y) \leq d_S(x, y) \leq d_M(x_0, x_1) + \dots + d_M(x_{p-1}, x_p) \leq \frac{\epsilon}{\epsilon - 2\delta}(d_M(\gamma_0, \gamma_1) + \dots + d_M(\gamma_{p-1}, \gamma_p)) = \frac{\epsilon}{\epsilon - 2\delta}d_M(x, y) < d_M(x, y)(1 + \frac{4\delta}{\epsilon})$ where the last inequality is hold due to the Taylor expansion. ■

The convergence theorem says that the shortest path approximation to the geodesic distance on the original manifold is close to perfect provided that the data points are densely sampled from the underlying manifold. As an example, the blue straight line in the right image in Figure 2 represents the true geodesic curve between two points on the graph while the red curve represents the shortest path between the same fixed two points.

Given that the geodesic distance between each pair of points can be approximated by the length of the shortest path on the graph which can further be represented as a Euclidean distance matrix, the remaining question is how to really embed data points in high dimensions to data points living in low dimensional spaces. This turns out to be a solved problem by singular value decomposition (SVD) and the basis of low dimensional spaces are actually reconstructed by eigenvectors of the distance matrix or its transformed matrix [JI86, CCR03].

What SVD means is that any real matrix $A_{m \times n}$ which has m rows and n columns can be written as $A = U\Sigma V^T$ where U is an $m \times m$ orthogonal matrix and V is a $n \times n$ orthogonal matrix. Here Σ is a $m \times n$ diagonal matrix whose nonzero elements all live in its diagonal. The benefit of SVD lies in its stability in numerical computation and its adaptiveness to arbitrary matrix. To perform embedding we need another theorem which allows us to transform a Euclidean matrix to a Gram Matrix which is further a positive semi-definite matrix. The theorem is formally stated as follows:

Isomap	classical Multidimensional Scaling (cMDS)
input: $x_1, \dots, x_n \in \mathbb{R}^D, k$	input: $D \in \mathbb{R}^{n \times n} (D_{ii} = 0, D_{ij} \geq 0), d \in \{1, \dots, n\}$
<ol style="list-style-type: none"> 1. Form the k-nearest neighbor graph with edge weights $W_{ij} := \ x_i - x_j\$ for neighboring points x_i, x_j. 2. Compute the shortest path distances between all pairs of points using Dijkstra's or Floyd's algorithm. Store the squares of these distances in D. 3. Return $Y := \text{cMDS}(D)$. 	<ol style="list-style-type: none"> 1. Set $B := -\frac{1}{2}HDH$, where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ is the centering matrix. 2. Compute the spectral decomposition of B: $B = U\Lambda U^T$. 3. Form Λ_+ by setting $[\Lambda_+]_{ij} := \max\{\Lambda_{ij}, 0\}$. 4. Set $X := U\Lambda_+^{1/2}$. 5. Return $[X]_{n \times d}$.

Figure 3: Algorithm for Isomap

Theorem 3: A nonnegative symmetric matrix $D \in \mathbb{R}^{n \times n}$, with zeros on the diagonal, is an Euclidean distance matrix if and only if $B \triangleq -\frac{1}{2}HDH$, where $H = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, is positive semi-definite. Furthermore, B will be the Gram matrix for a mean centered configuration with inter-point distances given by D .

Theorem 3 gives us a way to transform our Euclidean matrix D to a Gram matrix B which, according to the definition, can be represented as $B = XX^T$. Since B can also be represented as $B = U\Sigma U^T$ by SVD, we have: $X = U\Sigma^{\frac{1}{2}}$, which is the required embedding of the original data points. Notice that the columns of the orthogonal matrix U actually forms a set of new orthogonal basis and the number of nonzero terms in the diagonal of Σ determines the intrinsic dimensions of the original manifold. Being able to estimate the intrinsic dimensions of the manifold can be viewed as another advantage of Isomap.

Due to the noise and the inaccurate estimation in real applications, the distance matrix D is not perfectly Euclidean and as a consequence, the eigenvalues of B are not all positive or zero. In order to impose the constraints for D and in the mean time get rid of the inaccurate estimation, one useful method is to set all negative values in Σ to be zero and for convenience we sort all the eigenvalues in decreasing order. We now have the flexibility to chose the number of dimensions for the embedding. The method of embedding used in Isomap is called multidimensional scaling. [CCR03]

As an example for the embedding, consider a 10×10 Euclidean distance matrix D and its associative Gram Matrix B , we may end up with the eigenvalues for B : 10,8,7,6,2,1,0,0,-1,-2. The existence of negative eigenvalues indicates that the estimation is not perfectly accurate. We first set all negative values to be zero and this give us all the transformed eigenvalues: 10,8,7,6,2,1,0,0,0,0. We can now get rid of the zero terms and only preserve eigenvalues: 10,8,7,6,2,1. This also indicates that the number of intrinsic dimensions is equal to 6. Furthermore, notice that 2 and 1 are relatively small compared to other eigenvalues, we can further discard eigenvalues 2 and 1 without introducing too much reconstruction error. Therefore, the ultimate embedding requires only eigenvalues: 10,8,7,6 and their corresponding eigenvectors. In general, users can specify the number of dimensions d for the embedding and what Isomap does is simply to preserve the d most significant eigenvalues along with the corresponding eigenvectors. The algorithm for Isomap along with its embedding is presented in Figure 3 and the time complexity is $O(n^3)$.

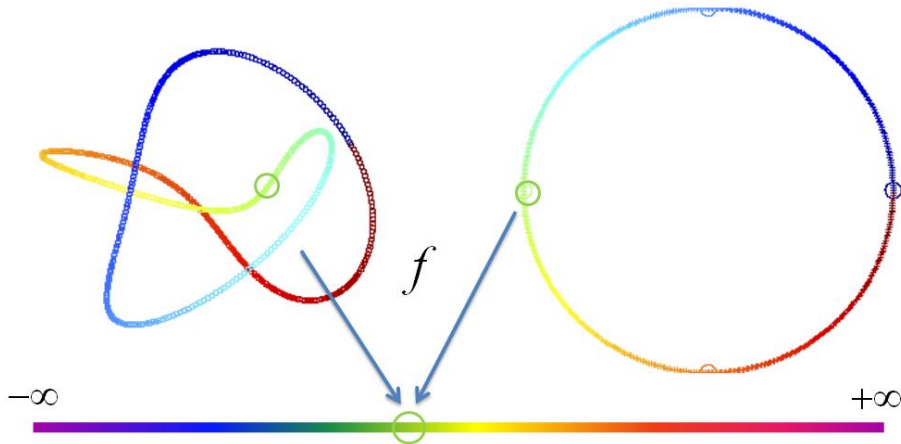


Figure 4: Laplacian Eigenmap tries to preserve the locality

3 Laplacian Eigenmap

Different from Isomap which is indeed an algorithm that tries to preserve the global geodesic distance between any pair of points, Laplacian Eigenmap only tries to preserve the locality, that is, for points which lie very close to each other in the original manifold, the corresponding points after embedding should still lie very close to each other. Figure 4 illustrates the idea of preserving locality. It has been shown [BN07] that the embedding provided by the Laplacian Eigenmap algorithms preserves the local information optimally in a certain sense. What's more amazing is that the justification for the Laplacian Eigenmap actually involves an interconnection between several areas of mathematics such as the Laplace-Beltrami Operator on the manifolds, the heat equations on the graph and the general eigenvalue problems in linear algebra. We will explore the connections in more detail in this section.

3.1 The Laplace Beltrami Operator

The Laplace-Beltrami operator Δ_M is a key geometric object associated to a Riemannian manifold. Given a differentiable function $f : M \rightarrow \mathbb{R}$, let $\nabla_M f$ be the gradient vector field on the manifold. The Laplace-Beltrami operator Δ_M (which acts on functions) can be defined as the divergence of the gradient: $\Delta_M f \triangleq -\nabla_M \cdot \nabla_M f$, where the negative sign is put for ease of representation. In the Euclidean space, the Laplace-Beltrami operator is the ordinary Laplacian:

$$\Delta f = - \sum_i \frac{\partial^2 f}{\partial x_i^2}$$

On a k -dimensional manifold M , in a local coordinate system (x_1, \dots, x_n) with a metric tensor g_{ij} , the Laplace-Beltrami operator applied to a function $f(x_1, \dots, x_k)$ is given by:

$$\Delta f = \frac{1}{\sqrt{\det(g)}} \sum_j \frac{\partial}{\partial x_j} \left(\sqrt{\det(g)} \sum_i g^{ij} \frac{\partial f}{\partial x_i} \right)$$

where g^{ij} are the components of the inverse of the metric tensor G^{-1} .

Now let's consider two neighboring points $x, z \in M$. They are mapped to $f(x)$ and $f(z)$ in the real line respectively. Let $c(t)$ be the geodesic curve parameterized by arclength such that $x = c(0)$ and $z = c(l)$ where $l = \text{dist}_M(x, z)$ is the geodesic distance between x and z . As an analogous to the Taylor expansion in the Euclidean space,

$$|f(z) - f(x)| = \left| \int_0^l df(c'(t)) dt \right| = \left| \int_0^l \langle \nabla f(c(t)), c'(t) \rangle dt \right| \quad (1)$$

$$\leq \int_0^l \|\nabla f(c(t))\| \|c'(t)\| dt = \int_0^l \|\nabla f(c(t))\| dt \quad (2)$$

$$\leq l \|\nabla f(x)\| + o(l) \approx \|\nabla f(x)\| \|z - x\| + o(\|z - x\|) \quad (3)$$

Equation(2) is valid due to the Schwartz inequality and the fact that $c(t)$ is parameterized by arclength and hence $\|c'(t)\| = 1$. Equation(3) is hold since we assume that M is isometrically embedded in \mathbb{R}^n and $l = \text{dist}_M(x, z) \approx \|x - z\|_{\mathbb{R}^n}$. Equation(3) tells that $\|\nabla f(x)\|$ provides us with an estimate of how far apart f maps nearby points. We therefore look for a map that best preserves locality on average by trying to find

$$\operatorname{argmin}_{\|f\|_{L^2(M)}=1} \int_M \|\nabla f(x)\|^2 dx$$

where the integral is taken with respect to the standard measure on a Riemannian manifold. The constraint $\|f\|_{L^2(M)} = 1$ is necessary because it avoids the trivial case that $f(x) \equiv 0$. It turns out that minimizing $\int_M \|\nabla f(x)\|^2 dx$ is equivalent to finding eigenfunctions of the Laplace Beltrami operator Δ_M . This is due to the fact that the divergence $-\nabla \cdot$ and ∇ are formally adjoint operators, i.e. :

$$\int_M \|\nabla f(x)\|^2 dx = \int_M \nabla f(x) \cdot \nabla f(x) dx = \int_M \Delta_M f(x) f(x) dx \quad (4)$$

$$= \lambda \int_M f^2(x) dx = \lambda \|f\|_{L^2(M)}^2 = \lambda \quad (5)$$

where λ is the eigenvalue of the Laplace Beltrami operator Δ_M and f is the corresponding eigenfunction. It is immediately seen that the minimization problem is equivalent to finding the smallest eigenvalue and its eigenvector associated to the Laplace Beltrami operator. However, the smallest eigenvalue is equal to zero for the case that f may collapse every point to a single point. To avoid this trivial case, we look for the second smallest eigenvalue and its corresponding eigenvector for the Laplace Beltrami operator.

3.2 Approximations: Connections with Heat Equations

Before we proceed to present the algorithm for the Laplacian Eigenmap, it would be better if we first find the connections with the heat kernel which will later be used to justify the choice of the weight associated with each edge on the graph in the formal algorithm. By this we can see how the Laplace Beltrami operator is approximated. The Laplace

Beltrami operator on differentiable functions on a manifold M is intimately related to the heat flow. Let $f : M \rightarrow \mathbb{R}$ be the initial heat distribution, $u(x, t)$ be the heat distribution at time t with the initial condition $u(x, 0) = f(x)$. The heat equation can be described as $(\partial_t + \Delta_M)u = 0$. The solution is given by $u(x, t) = K(x, t) * f(x)$ where $*$ represents the operator of convolution and $K(x, t)$ is called the heat kernel defined on the manifold M . On the manifold of dimension m , we can write the heat kernel approximately as a Gaussian kernel²: $K(x, t) \triangleq (4\pi t)^{-\frac{m}{2}} e^{-\frac{\|x\|^2}{4t}}$ and notice that as t tends to 0, the heat kernel $K(x, t)$ tends to be a Dirac's δ -function, i.e., $u(x, 0) = \lim_{t \rightarrow 0} K(x, t) * f(x) = f(x)$.

We are now equipped with all the tools to derive the approximation for the Laplace Beltrami operator on discrete domain. Suppose we are facing a cloud of n points: x_1, x_2, \dots, x_n which lie in the manifold M and are sampled uniformly and densely as well. We now consider the Laplace Beltrami operator applied to a point $x \in M$, for small t , from the definition of the partial derivative with respect to the t we have:

$$\Delta_M f(x) = \Delta_M u(x, 0) = -\partial_t u(x, t) = \lim_{t \rightarrow 0} \frac{u(x, 0) - u(x, t)}{t} \quad (6)$$

$$= \lim_{t \rightarrow 0} \frac{f(x) - u(x, t)}{t} \approx \frac{f(x) - K(x, t) * f(x)}{t} \quad (7)$$

$$= \frac{1}{t} \left[f(x) - (4\pi t)^{-\frac{m}{2}} \int_M e^{-\frac{\|x-y\|^2}{4t}} f(y) dy \right] \quad (8)$$

$$= \frac{1}{t} \left[f(x) (4\pi t)^{-\frac{m}{2}} \int_M e^{-\frac{\|x-y\|^2}{4t}} dy - (4\pi t)^{-\frac{m}{2}} \int_M e^{-\frac{\|x-y\|^2}{4t}} f(y) dy \right] \quad (9)$$

$$= \frac{(4\pi t)^{-\frac{m}{2}}}{t} \int_M e^{-\frac{\|x-y\|^2}{4t}} (f(x) - f(y)) dy \quad (10)$$

$$\approx \frac{(4\pi t)^{-\frac{m}{2}}}{nt} \left(f(x) \sum_i e^{-\frac{\|x-x_i\|^2}{4t}} - \sum_i e^{-\frac{\|x-x_i\|^2}{4t}} f(x_i) \right) \quad (11)$$

$$= \frac{(4\pi t)^{-\frac{m}{2}}}{nt} \mathbf{L}_n^t(f)(x) \quad (12)$$

where $\mathbf{L}_n^t(f)(x_i) \triangleq f(x_i) \sum_j e^{-\frac{\|x_i-x_j\|^2}{4t}} - \sum_j e^{-\frac{\|x_i-x_j\|^2}{4t}} f(x_j)$ is called the graph Laplacian operator on f . Given a sample of n points x_1, x_2, \dots, x_n from M , we can construct the complete weighted graph associated to that point cloud by taking x_1, x_2, \dots, x_n as vertices of the graph and taking the edge weights to be $w_{ij} = e^{-\frac{\|x_i-x_j\|^2}{4t}}$. Thus the corresponding graph Laplacian matrix \mathbf{L}_n^t associated with the graph Laplacian operator is given by:

$$(\mathbf{L}_n^t)_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_k w_{ik} & \text{if } i = j \end{cases}$$

From the derivation (6) - (12), we see that the graph Laplacian operator or its equivalent matrix representation on the graph approximates the Laplace Beltrami operator applied to the underlying manifold up to a constant scalar factor. This fact combined with the analysis of the Laplace Beltrami operator itself in section 3.1 give us a mathematical foundation as well as an elegant algorithm for the dimensionality reduction.

²In the Euclidean space, the heat kernel is exactly the Gaussian function

3.3 Put it Together: The Algorithms for Laplacian Eigenmap

From section 3.1 we know that to embed data points on an arbitrary manifold to a real line is equivalent to finding the second smallest eigenvalue as well as its corresponding eigenvector associated with the Laplace Beltrami operator. From section 3.2 we know that the Laplace Beltrami operator on the manifold can be approximated by the graph Laplacian matrix \mathbf{L}_n^t . Combing the results in both sections we see that to embed the data points on high dimensional manifold to a real line is equivalent to finding the second smallest eigenvalue as well as its corresponding eigenvector associated with the graph Laplacian matrix. More generally, suppose that we want to embed data points to d dimensional space, we shall preserve the d least significant eigenvalues and eigenvectors for the graph Laplacian matrix \mathbf{L}_n^t . One subtle issue here is that we need to impose the constraints for the eigenfunctions such that they won't collapse every point to a single point. Let's define the weight matrix W as $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{4t}}$ for $i \neq j$ and $W_{ii} = 0$ in the diagonal. Let D be the diagonal weight matrix such that $D_{ii} = \sum_j W_{ji}$. By this the graph Laplacian matrix \mathbf{L}_n^t is equal to $D - W$. As an approximation for the continuous case, the constraints for any eigenfunction f is $f^T D f = 1$. By integrating the constraint we actually transform the original problem to the generalized eigenvalue problem: $\mathbf{L}_n^t \mathbf{f} = \lambda D \mathbf{f}$. Following gives a formal description of the algorithm for the Laplacian Eigenmap for embedding $x_1, x_2, \dots, x_n \in \mathbb{R}^k$ to m dimensional Euclidean space:

Step 1: [Construct the Adjacency Graph]. Node i and node j are connected by an edge if $\|x_i - x_j\|^2 < \epsilon$ where the norm is the usual Euclidean norm in \mathbb{R}^k .

Step 2: [Choosing the Weights]. If node i and node j are connected, put $W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{4t}}$ where t is a given parameter. Otherwise put $W_{ij} = 0$.

Step 3: [Eigenmaps]. Assume the graph G , constructed above, is connected, otherwise proceed with step 3 for each connected component. Compute the eigenvalues and eigenvectors for the generalized eigenvector problem: $\mathbf{L}_n^t \mathbf{f} = \lambda D \mathbf{f}$ where the symbols are defined properly in the above section. We leave out the eigenvector corresponding to eigenvalue 0 and use the next m eigenvectors $\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_m$ for embedding in m -dimensional Euclidean space, that is, $x_i \rightarrow (\mathbf{f}_1(x_i), \mathbf{f}_2(x_i), \dots, \mathbf{f}_m(x_i))$.

This concludes our discussion of the method of Laplacian Eigenmap so far. Our final remark on the method is that the Laplacian on the graph links various areas of mathematics such as differential geometry, spectral graph theory and partial differential equations. Its algorithm is simple and its mathematical justifications are surprisingly beautiful. In the next section we will move on to another method of dimensionality reduction which takes a dramatically different perspective of view from Isomap and Laplacian Eigenmap.

4 Diffusion Map

Section 2 and Section 3 present two important manifold learning methods, Isomap and Laplacian Eigenmap. One of their essential parts lies in how to define the distance between each pair of points on the graph which approximates the distance on the manifold. Dif-

ferent from Isomap and Laplacian Eigenmap, Diffusion Map defines the distance from the perspective of random walk on the graph. Here, a random walk on a graph is a stochastic process which models a molecule jumping randomly from one point to another following certain path on the graph. In this section, we will first give basic background of random walk on a graph. We will show a necessary and sufficient condition for a unique stationary distribution given transition probability. After that, we define the distance function, denoted as *diffusion distance*, based on the random walk on the similarity graph. Finally, we show that low dimensional description of the data by the first few eigenvectors is the optimal embedding of high dimensional data onto a low dimensional Euclidean space.

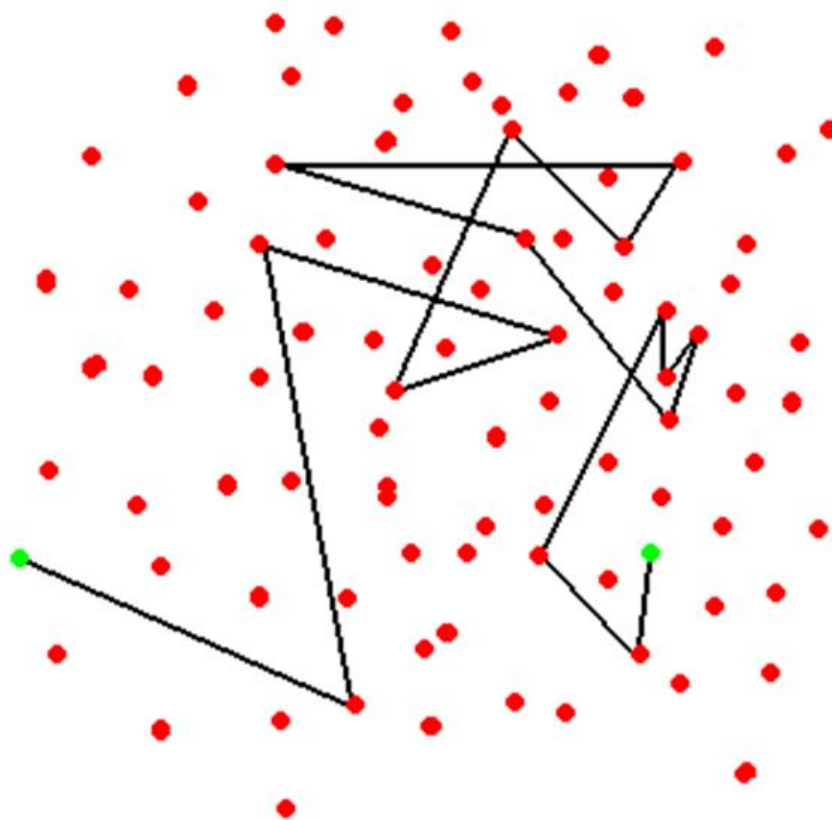


Figure 5: Random walk on the graph

4.1 Random Walk

Given a graph G , let $E(G)$ denote the set of edges of G . A walk on G is a sequence of points (x_0, x_1, \dots, x_s) with $\{x_{i-1}, x_i\} \in E(G)$ for all $1 \leq i \leq s$. Figure 5 illustrates the idea of the random walk on a graph. A random walk is determined by the stationary transition probabilities: $P(u, v) = \mathbb{P}(x_{i+1} = v | x_i = u)$, which is the probability of jumping from point u to point v . Notice that here P is the transition matrix whose item in u^{th} row and v^{th} column is denoted as $P(u, v)$. Clearly, for each point u , $\sum_v P(u, v) = 1$.

For any initial distribution $f : V \rightarrow \mathbb{R}$ with $\sum_v f(v) = 1$, the distribution after k steps is fP^k (i.e., a matrix multiplication with f viewed as a row vector and P is the matrix of transition probabilities). The random walk is said to be *ergodic* if there is a unique *stationary distribution* $\pi(v)$ satisfying $\lim_{s \rightarrow \infty} fP^s(v) = \pi(v)$. It is easy to see that the necessary conditions for the ergodicity of P are: (i) *irreducibility*, i.e. for any $u, v \in V$, there exists some s such that $P^s(u, v) > 0$; (ii) *aperiodicity*, i.e., the greatest common divisor of the set $\{s : P^s(u, v) > 0\}$ is equal to 1. As it turns out, these are also the sufficient conditions [Chu].

For a weighted graph, let W denote the symmetric weight matrix whose item in the u^{th} row and v^{th} column, $w(u, v)$, is the weight associated with the edge from point u to point v . We further define the transition probability of a random walk on the graph: $P(u, v) = \frac{w(u, v)}{\sum_z w(u, z)}$. The two conditions for ergodicity are equivalent to the conditions that the graph be (i) connected and (ii) non-bipartite. Therefore, (i) and (ii) together deduce ergodicity. Notice that the transition matrix P defined above can be represented as: $P = D^{-1}W$ where D is the diagonal matrix such that $D_{ii} = \sum_j W(j, i)$, we have: $\mathbb{1}DP = \mathbb{1}W = \mathbb{1}D$, where $\mathbb{1}$ is the row vector of all ones. Therefore, $\mathbb{1}D$ is the stationary distribution up to a constant scalar factor. The regularized stationary distribution is therefore defined to be: $\pi = \mathbb{1}D / \sum_i D_{ii}$. From the perspective of eigenvalue problem, π is the left eigenvector of P with its corresponding eigenvalue equal to 1.

4.2 Diffusion Distance and Diffusion Map

Now, we will see how to utilize the property of random walk on a weighted graph to model the diffusion distance. Given n data points $\{x_i\}_{i=1}^n$ where each $x_i \in \mathbb{R}^p$, we define a pairwise similarity matrix between points. For example, as used in Laplacian Eigenmap, we can use a Gaussian kernel with width ϵ : $L_{i,j} = e^{-\frac{\|x_i - x_j\|^2}{2\epsilon}}$ and a diagonal normalization matrix D . Then, the matrix $M = D^{-1}L$ is a normalized graph Laplacian with all row sums equal to one, and thus can be interpreted as defining a random walk on the graph. Under this perspective of view, $M_{i,j}$ denotes the transition probability from the point x_i to the point x_j in one time step ϵ . Thus, we have the conditional probability:

$$\mathbb{P}\{x(t + \epsilon) = x_j | x(t) = x_i\} = M_{i,j}$$

For ϵ large enough, all points in the graph are connected. Furthermore, the graph is not bipartite since for any two points there will have a weighted edge. According to Section 4.1, the random walk on the graph with transition probability M is ergodic, and has a unique stationary distribution: $\pi = \frac{\mathbb{1}D}{\sum_i D_{i,i}}$. If we let φ_j, ψ_j be the left and right eigenvectors of M with eigenvalue λ_j , where $\lambda_0 = 1 \geq \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{n-1} \geq 0$, then π is exactly φ_0 .

Based on the above observation, we now look at the probability distribution of a random walk landing at location y at time t , $p_t(y|x)$, given a starting location at x when the time is equal to zero. First, for infinite time t , we have

$$\lim_{t \rightarrow \infty} p_t(y|x) = \pi(y) = \varphi_0(y).$$

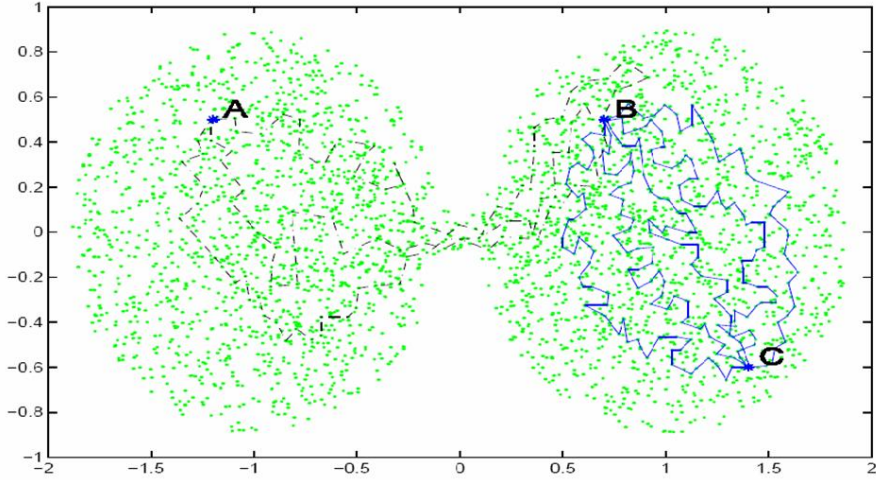


Figure 6: Random walk on the graph

This shows that when t tends to infinity, the starting point x doesn't impact the long-term distribution, and $\varphi_0(y)$ can also be viewed as a density estimate (importance) at the point y . For finite time t , the probability distribution can be decomposed using eigenbasis $\{\varphi_j\}$:

$$p_t(y|x) = \varphi_0(y) + \sum_{j \geq 1} \psi_j(x) \lambda_j^t \varphi_j(y)$$

To calculate the diffusion distance, we not only consider the evolution of their probability distributions, but also weight every point by its probability density (or importance). Thus we can define the diffusion distance $D_t(x_0, x_1)$ between any two points x_0 and x_1 as below:

$$D_t^2(x_0, x_1) = \sum_y (p_t(y|x_0) - p_t(y|x_1))^2 \frac{1}{\varphi_0(y)} \quad (13)$$

$$= \sum_y \left(\sum_{j \geq 1} \psi_j(x_0) \lambda_j^t \varphi_j(y) - \sum_{j \geq 1} \psi_j(x_1) \lambda_j^t \varphi_j(y) \right)^2 \frac{1}{\varphi_0(y)} \quad (14)$$

$$= \sum_y \left(\sum_{j \geq 1} \lambda_j^t (\psi_j(x_0) - \psi_j(x_1)) \varphi_j(y) \right)^2 \frac{1}{\varphi_0(y)} \quad (15)$$

$$= \sum_{j \geq 1} \lambda_j^{2t} (\psi_j(x_0) - \psi_j(x_1))^2 \quad (16)$$

$$= \|\Psi_t(x_0) - \Psi_t(x_1)\|^2, \quad (17)$$

where $\Psi_t(x) = [\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), \lambda_3^t \psi_3(x), \dots]$.

The derivations (13) - (17) show that the diffusion distance is equal to the Euclidean distance in the Ψ space. Since in many practical applications the spectrum of the matrix M has a spectral gap with only a few eigenvalues close to one and all additional eigenvalues much smaller than one, the diffusion distance at a large enough time t can be well approximated by only the first few k eigenvectors $\psi_1(x), \dots, \psi_k(x)$, denoted as *diffusion*

map, with a negligible error of the order of $O((\lambda_{k+1}/\lambda_k)^t)$. This provides a theoretical justification for dimensionality reduction using these eigenvectors.

Take Figure 6 as an example, where the data are uniformly distributed in a region composed of two large clusters connected by a narrow bridge. Intuitively, the probability distributions of point B and C in the same cluster are more similar than B and A which lie in different clusters. And it takes longer time for point A to arrive at points in the right cluster (e.g. point C) in comparison with point B. Therefore, the diffusion distance between B and A will be greater than the distance between B and C. By approximating the diffusion distances, Diffusion Map provides a low-dimensional description of the data points combining the geometry of the underlying data set.

5 Experiments

In this section we present our experimental results both on synthetic data and real applications such as optical character recognition and face classification. The MATLAB implementation of various manifold learning algorithms, including the Isomap, Laplacian Eigenmap and Diffusion Map presented above, can be found in [Wit05].

Figure 7 shows the dimensionality reduction from a three dimensional spiral curve attached to a torus to two dimensional space. It can be seen that all methods perform pretty good in the sense that the locality is preserved well. Figure 8 shows the dimensionality reduction from data points living in a Swiss-roll shaped surface to 2-dimensional plane. In this case, some linear methods like PCA fail to preserve the underlying structure while Isomap, Laplacian Eigenmap and Diffusion Map still work very good.

We now move on to some real applications. The manifold learning algorithm we use for the real applications is Isomap. The first application we show is called the optical digit recognition. In this scenario, we are given 2000 images of written digits from 0 to 9. The requirement is that computers can automatically cluster these images based on their content information, namely 0, 1, 2, ..., 9. The benefit of this automatic clustering is obvious: given a new image of a certain written digit, machines can recognize the cluster it belongs to and hence assign the corresponding label. By this we enable the machine to have the ability of recognizing human written digits. The role of dimensionality reduction is to extract the most distinguished features of these digits and improve the clustering and classification efficiency. Notice that the original image is 64×64 , that is, it potentially lives in a 4096 dimensional manifold. Figure 9 gives a sample of the written digits whose shape and style vary from image to image. Figure 10 shows the result of clustering digits of 3,4 and 7 by only preserving 2 dimensions. For ease of visualization, we only display 30 images for each digit where in the graph each point represents an image of a digit whose label is specified by the color. It can be seen that by even preserving only 2 dimensions, the result of clustering is still pretty good. Figure 11 further shows the result of clustering by preserving 3 dimensions.

As an another application, we apply Isomap algorithm to the human face classification problem. Figure 12 shows a sample of human faces belonging to 4 different people. The image size is 92×112 . It can be seen that the expression as well as the pose of the head varies from image to image in the samples. In our experiment, we collect 10 images

of faces for each person and the clustering result by preserving only 2 dimensions is displayed in Figure 13 where each dot represents an image of a human face whose label is specified by the color. Although there exists some ambiguity between the blue and yellow category where both person wear glasses, the general clustering result is still very good. Furthermore, Figure 14 shows the result of clustering by preserving 3 dimensions. It can be seen that the original ambiguity is partially solved by including one more dimension.

6 Conclusions

In this paper, we introduce three manifold learning methods including Isomap, Laplacian Eigenmap and Diffusion Map along with their intuitive ideas and the mathematical justifications. All the three methods share the common assumption that data points lie in a low dimensional manifold which is embedded in a high dimensional space. The common objective is to find a mapping from high dimensional space to low dimensional space while preserving the “distance” among points. The central theme here is that all three algorithms employ singular value decomposition to achieve the dimensionality reduction and the embedding as well. The difference lies in the definition and interpretation of the distance on the manifold: Isomap tries to preserve the geodesic distance between each pair of points; Laplacian Eigenmap tries to preserve the local distance between each pair of nearby points; and Diffusion map tries to preserve the diffusion distance between every two points based on the random walk on a graph.

It is interesting to realize that these manifold learning methods have deep connections with several areas of mathematics. First, they all borrow various concepts from differential geometry: Isomap adopts the idea of geodesics; Laplacian Eigenmap tries to approximate the Laplace Beltrami operator on the manifolds; and Diffusion Map applies the stochastic analysis on the manifold. Second, the justifications for these methods involve many other branches of mathematics. The Laplacian Eigenmap uses the heat flow on the graph to justify the approximation; the analysis of Isomap and Diffusion Map also involves large amount of probability analysis. Third, one emerging topic linking all three methods is called the spectral graph theory which aims to study the eigenvectors and eigenvalues of a graph. It is an amusing trip to find the connections while exploring the mathematical background of these methods.

In addition to the theoretical explorations of these three methods, we also perform several experiments both on synthetic data and real applications including the optical digit recognition and human face classification. The experiment result shows that these dimensionality reduction algorithms work pretty well at least in our toy examples.

While a bunch of manifold learning algorithms have been developed, how to evaluate the results of manifold learning (i.e. how successful are these algorithms at uncovering manifold structures) is still open. Although there are a couple of theoretical performance results for these algorithms, the primary evaluation methodology has been to run the algorithm on some (often artificial) data sets and see whether the results are intuitively pleasing [Cay03]. We may also ask the following questions. 1) Is the assumption of manifold structure reasonable? What if there are noises in the data and how to make the algorithm more robust to noises? Can we combine some ideas from stochastic analysis

on manifold? 2) Since almost all the manifold learning algorithms use SVD as a basic decomposition tool, the complexity is also an issue for the scalability of the algorithms. Furthermore, how to design incremental manifold learning algorithms is also worthy of exploring. Several efforts on this direction include [LJ04] and [KOP05]. And, are there any other alternatives of matrix factorization (e.g. non-negative matrix factorization) that may give us some other meaningful and useful results? 3) For complex data sets containing multiple manifolds of possibly different dimensionalities, it is unlikely that the existing manifold learning approaches can discover all the interesting lower-dimensional structures. Can we develop a hierarchical manifolds learning framework to discover a variety of the underlying low dimensional structures?

References

- [BdSLT00] M. Bernstein, V. de Silva, J. Langford, and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Stanford University, Stanford, 2000.
- [BN03] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.
- [BN07] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian based manifold methods. *Journal of Computer and System Sciences*, 2007.
- [Cay03] Lawrence Cayton. Algorithms for manifold learning. Technical report, University of California - San Diego, California, 2003.
- [CCR03] T. F. Cox, M. A. A. Cox, and B. Raton. Multidimensional scaling. *Technometrics*, 45(2):182–182, May 2003.
- [Chu] Fan Rong K Chung. *Spectral Graph Theory*.
- [Jl86] Jolliffe and I.T. *Principal Component Analysis*. Springer-Verlag, 1986.
- [KOP05] Olga Kouropteva, Oleg Okun, and Matti Pietikäinen. Incremental locally linear embedding. *Pattern Recognition*, 38(10):1764–1767, 2005.
- [LJ04] Martin H. C. Law, Nan Zhang 0002, and Anil K. Jain. Nonlinear manifold learning for data stream. In *SDM*, 2004.
- [LLL06] S. Lafon, S. Lafon, and A.B. Lee. Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. 28(9):1393–1403, 2006.
- [NLCK05] Boaz Nadler, Stphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Advanced Neural Information Processing System (NIPS)*, 18, 2005.
- [TdSL00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[Wit05] Todd Wittman. Mani fold learning matlab demo, 2005.

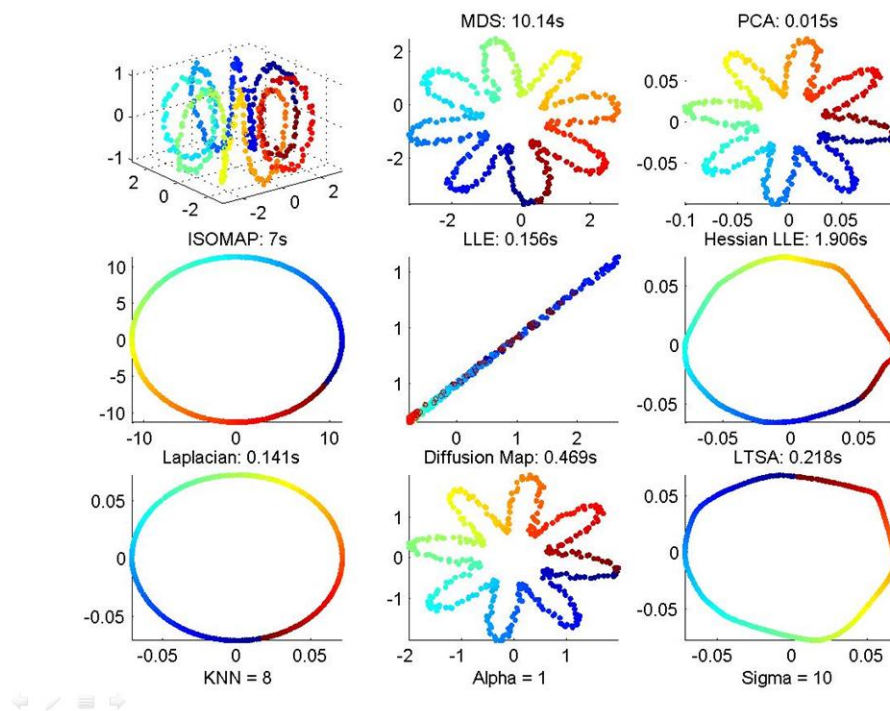


Figure 7: Comparisons between various dimensionality reduction algorithms

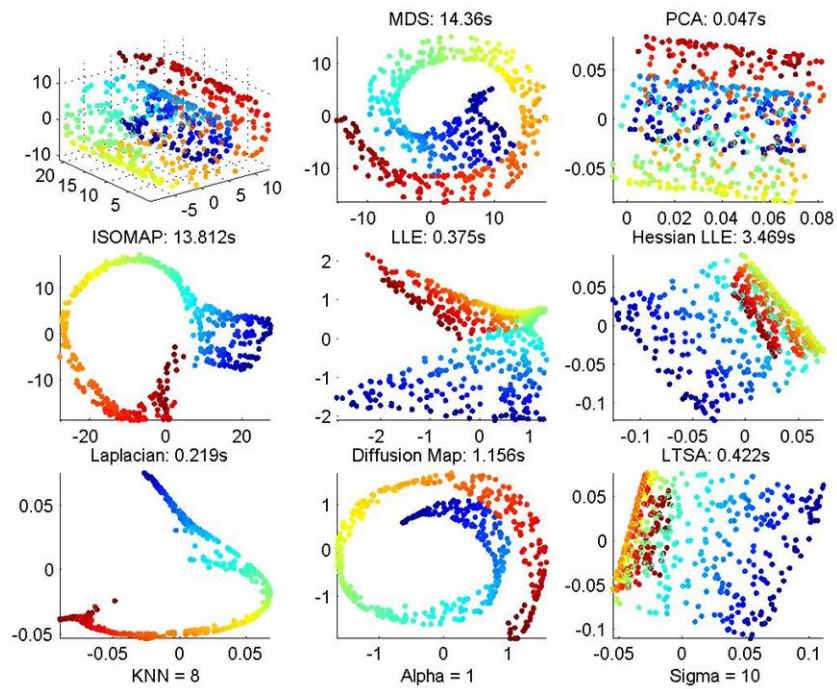


Figure 8: Comparisons between various dimensionality reduction algorithms



Figure 9: Samples of written digits

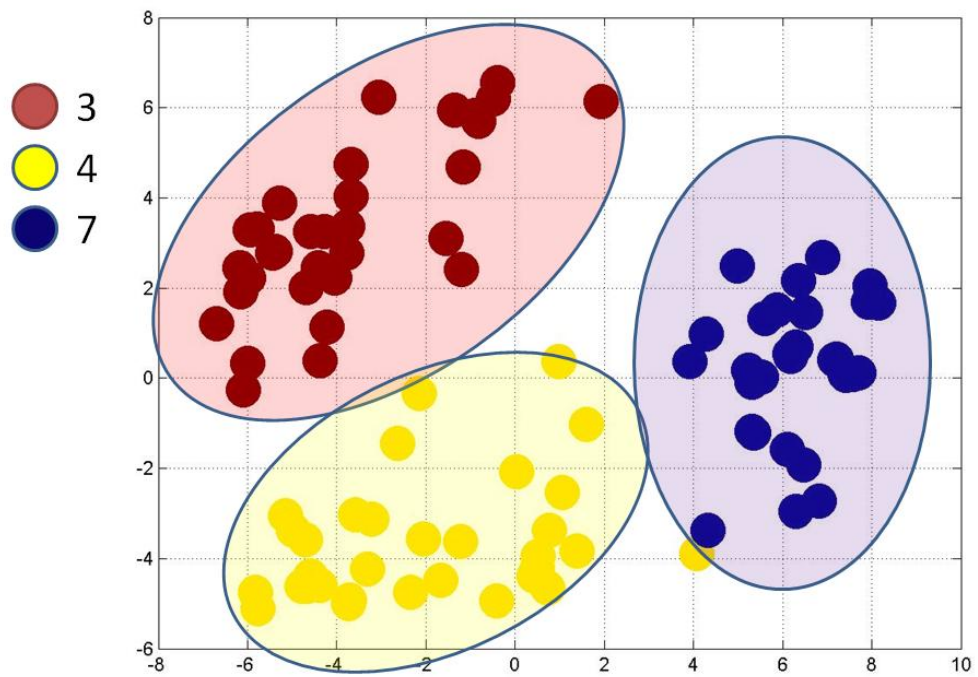


Figure 10: Result of dimensionality reduction to 2 dimensions for written digits

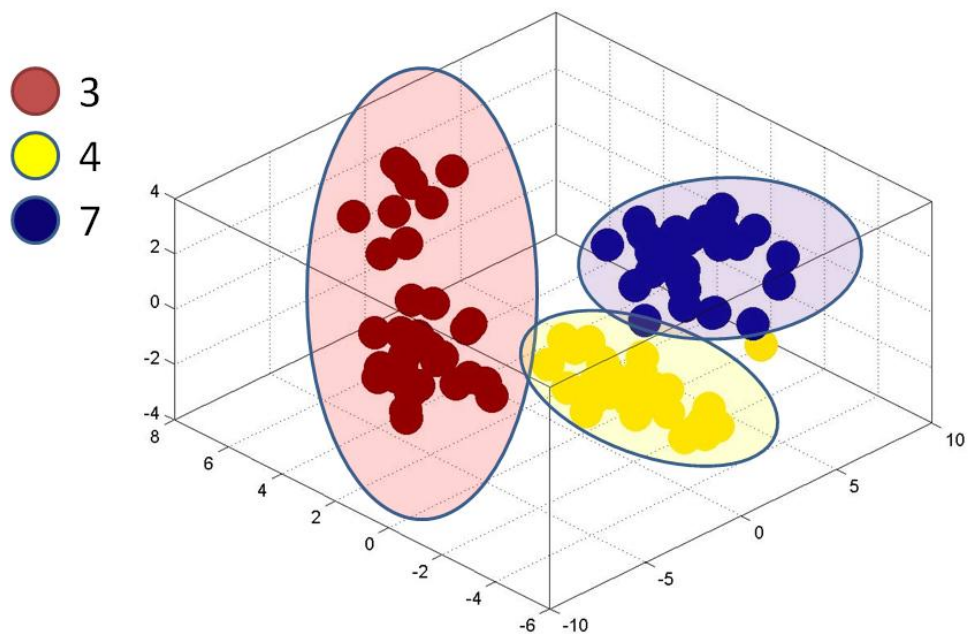


Figure 11: Result of dimensionality reduction to 3 dimensions for written digits

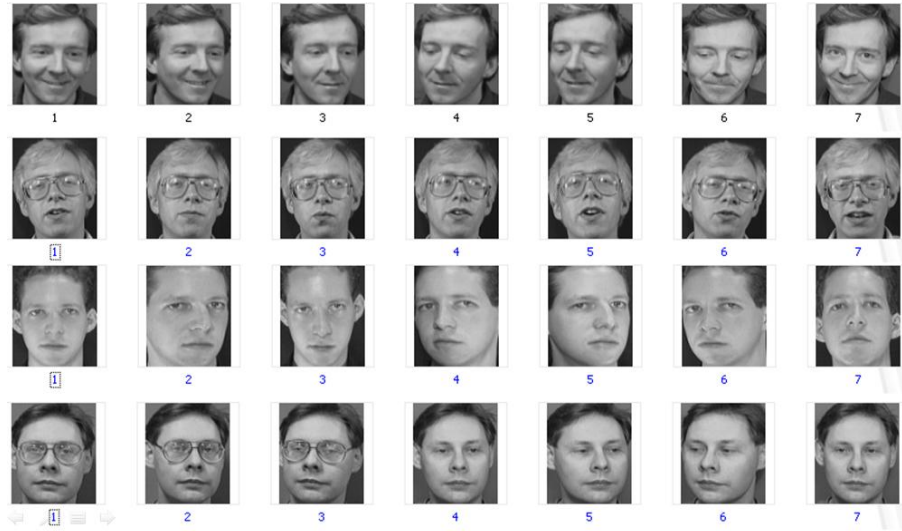


Figure 12: Samples of human faces

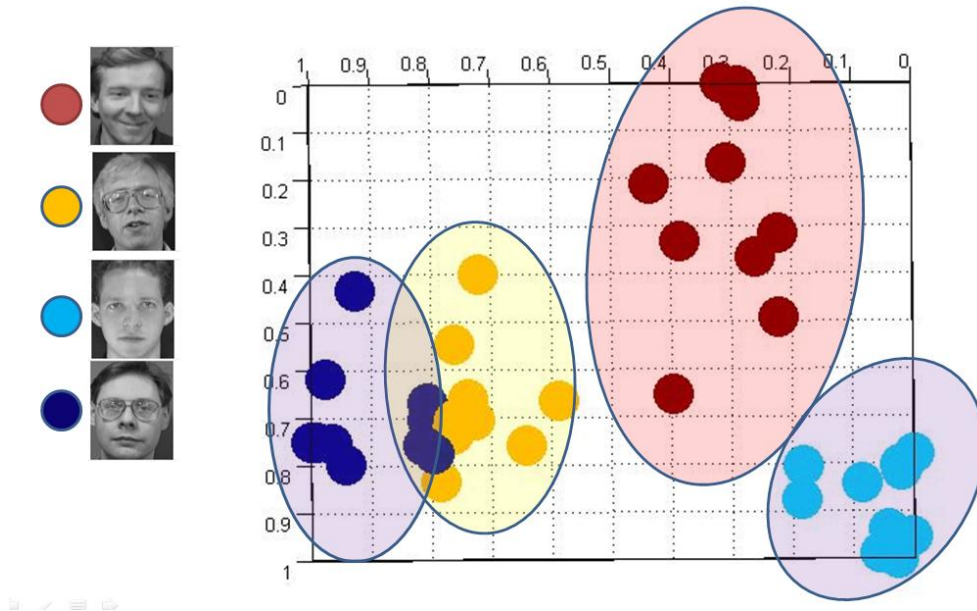


Figure 13: Result of dimensionality reduction to 2 dimensions for human faces

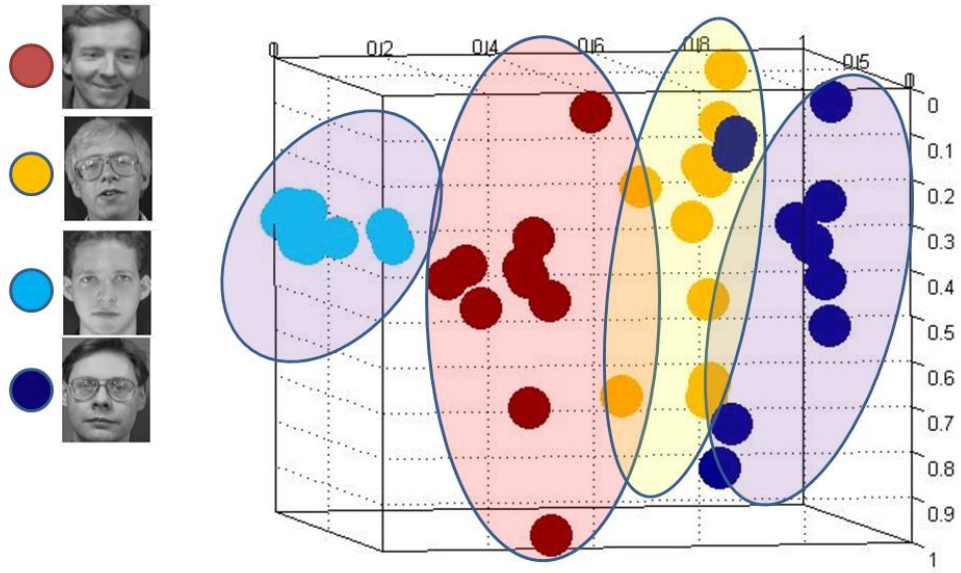


Figure 14: Result of dimensionality reduction to 3 dimensions for human faces