Manifold Learning and Dimensionality Reduction with Diffusion Maps

Richard Socher
Supervisor: Prof. Matthias Hein

July 20, 2008

Abstract

This report gives an introduction to diffusion maps, some of their underlying theory, as well as their applications in spectral clustering. First, the shortcomings of linear methods such as PCA are shown to motivate the use of graph-based methods. We then explain Locally Linear Embedding [9], Isomap [11] and Laplacian eigenmaps [1], before we give details on diffusion maps and anisotropic diffusion processes.

Contents

1 Introduction **2**
1.1 Principal Component Analysis . . . 2
1.2 Graph-Based Algorithms . . . 3
1.3 Locally Linear Embedding . . . 4
1.4 Isomap . . . 5
1.5 Laplacian Eigenmaps . . . 6
1.5.1 Graph Laplacians . . . 6
1.5.2 The Algorithm . . . 7
1.5.3 Algorithm Justification . . . 7
1.5.4 Eigenmaps - Conclusion . . . 8

2 Diffusion Maps **8**
2.1 Intuition . . . 9
2.2 Diffusion Distance . . . 9
2.3 Embedding . . . 11
2.4 Conclusion . . . 11

3 Anisotropic Diffusion **12**
3.1 Family of Anisotropic Diffusions . . . 12
3.2 Laplace-Beltrami Operator . . . 13
3.3 Influence of Density and Geometry . . . 15

4 Conclusion **16**

A PCA code and mapping to Eigenvector **17**

B Rayleigh Ritz Proof **19**

C Implementation of Laplacian Eigenmaps **19**

D Implementation of Anisotropic Diffusion **20**

E Implementation of Eigenfunctions for Symmetric and Random Walk Laplacian **20**

1 Introduction

High dimensional data such as characters or image sets are hard to cluster or interpret. However, these complex data sets may be realizations of only a few intrinsic parameter changes. A set of images depicting faces for instance might be governed by three parameters: horizontal rotation, vertical rotation and lighting changes. Such features may be recovered by non-linear dimensionality reduction techniques.

In contrast to linear methods such as PCA or LDA ([4]), non-linear methods do not ignore protrusion or concavity of the data and are therefore able to handle a broader range of data sets. In this report we will focus on unsupervised manifold learning for dimensionality reduction and clustering using diffusion maps. For the most part, the presented methods assume that the data lies on a low-dimensional manifold in a high-dimensional observation space. The goal is to find a mapping from the original D-dimensional data X to a d-dimensional space Y in which local distances are preserved as much as possible and $d < D$:

$$\Psi : X \in \mathbb{R}^D \to Y \in \mathbb{R}^d \tag{1}$$

This section covers some of the basic definitions, underlines the shortcomings of linear methods and gives a quick introduction to similar methods. Section 2 will throw light on the embedding defined by diffusion maps and section 3 describes anisotropic diffusion, a technique to recover manifold geometry regardless of the sampling distribution on it.

While most presented methods (except Isomap) work on any graph structure or Euclidean space they are often successfully used in the field of manifold learning. Furthermore, some results such as the convergence of the graph Laplacian to the Laplace-Beltrami operator hold for general manifolds and not only in Euclidean space.

Definition 1.1 (Manifold)**.** *We define a manifold as a topological space that is locally Euclidean but might have a more complicated structure globally. It is seen as an image of a lower-dimensional domain. The input points of the D-dimensional observation space are samples taken from this domain.*

1.1 Principal Component Analysis

In order to demonstrate the shortcomings of purely linear methods, let us consider Principal Component Analysis (PCA). The goal of PCA is to find an optimal subspace that best preserves the variance of the data. The input and output of PCA are defined as in equation 1, given N input points. The algorithm performs the following steps:

1. Calculate the empirical mean vector for each dimension $\mu[dim] = \frac{1}{N}\sum_{i=1}^{N} X[dim, i]$
2. Subtract μ from each column of the $M \times N$ input matrix X: $\mathbf{B} = \mathbf{X} - \mu \cdot \mathbf{1}$ where $\mathbf{1}$ is a $1 \times N$ vector of 1's.
3. Compute the $D \times D$ covariance matrix $\mathbf{C} = \frac{1}{N-1}\mathbf{B} \cdot \mathbf{B}^T$
4. Solve the eigenvector problem to find the matrix $\mathbf{V}$ of eigenvectors, so that $\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D}$ with $\mathbf{D}$ being the matrix in which the decreasing eigenvalues (corresponding to their eigenvectors) are on the diagonal and $V^T = V^{-1}$. All eigenvectors are orthogonal and form an orthonormal basis.
5. Project the data onto the new d-dimensional subspace, using the first d columns of V, where d is chosen according to some measure (data energy or highest variance): $\mathbf{Y} = \mathbf{V}^{(d)} \cdot \mathbf{X}$

Appendix A lists the Matlab code that performs PCA and creates the following figures which demonstrates one example for which PCA works well and one for which it cannot perform well since it ignores the non-linear geometry of the manifold.

Figure 1 shows a Gaussian distribution together with the first (and only) two principal components, calculated by the method described above. The vectors are therefore the eigenvectors of the matrix C. The coloring is linearly dependent on values of x_1 and x_2. The right side shows the projection on the eigenvector corresponding to the largest eigenvalue. As one can see, very little information is lost through this transformation.

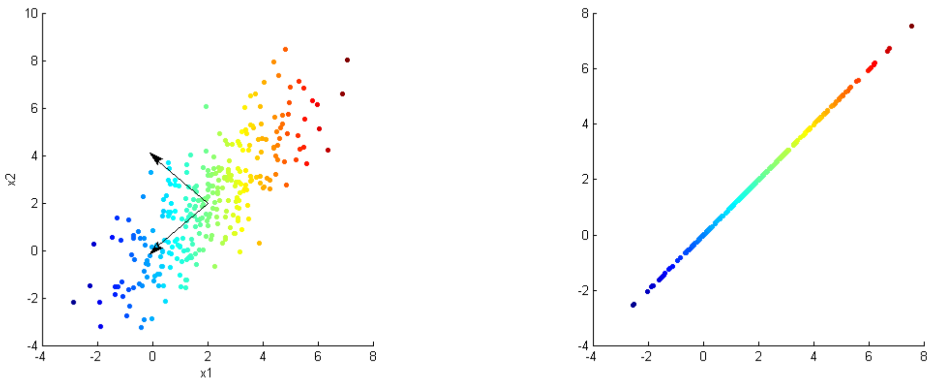


Figure 1: Working example of PCA. The left image shows a Gaussian distribution together with the two principal components. The coloring is dependent on values of x_1 and x_2. The right side shows the projection on the eigenvector corresponding to the largest eigenvalue.

Figure 2 shows that PCA cannot handle non-linear datasets. The left image shows a spiral distribution (2-d Swiss role) together with the two principal components. The coloring is dependent on values of t, where the function is given as $f(t) = (t\cos(t), t\sin(t))$. The right side shows the overlapping projection on the eigenvector corresponding to the largest eigenvalue. One can observe that blue, red and yellow points are all overlapping in the center of the projected line. This means that most geometric information of the data is lost through this projection.

Another problem of PCA is that it tries to preserve large distances between data points. However, in most cases distances are only meaningful in local neighborhoods. The following section presents local methods which address this problem.

1.2 Graph-Based Algorithms

Graph-based algorithms perform three steps.

1. Build undirected similarity graph $G = (V, E)$.
2. Estimate local properties, i.e. define the weight matrix W to define the weighted similarity graph $G = (V, E, W)$, where $w_{ij} \geq 0$ represents the weight for the edge between vertex i and j. Weights are obtained by means of a kernel, a term we define below. A weight of 0 means that the vertices are not connected.
3. Derive an optimal global embedding Ψ which preserves these local properties.

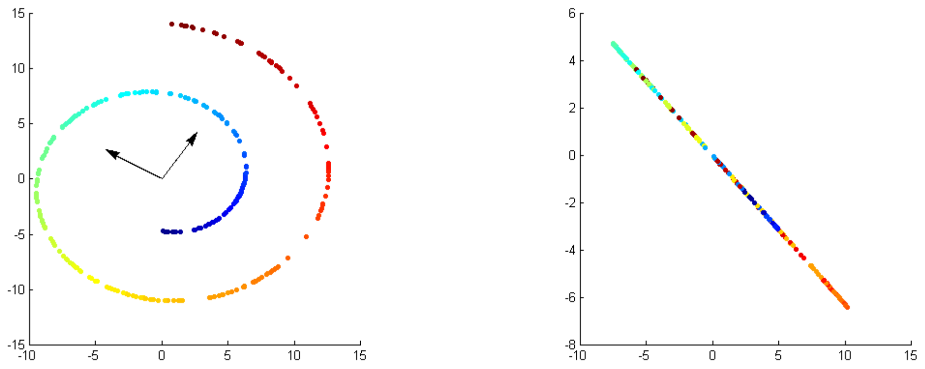


Figure 2: PCA cannot handle non-linear datasets. The left image shows a spiral distribution (2-d Swiss role) together with the two principal components. The coloring is dependent on values of t, where the function is given as $f(t) = (t\cos(t), t\sin(t))$. The right side shows the overlapping projection on the eigenvector corresponding to the largest eigenvalue.

There are three often used techniques for building the similarity graph G[1]. First, there is the ϵ-neighborhood graph which connects all vertices with distance $||x_i - x_j||^2$ smaller than ϵ. The ϵ graph is naturally symmetric.

Contrary to this local connection is the fully connected graph which uses a similarity function that incorporates local neighborhood relations such as the Gaussian function: $w_{ij} = \exp(-||x_i - x_j||^2/(2\sigma^2))$. This leads directly to the third step, since it implicitly defines the weights.

k-nearest neighbor (kNN) graphs combine both worlds by connecting each vertex only to its k-nearest neighbors. If one needs a symmetric weight matrix one may either ignore the asymmetry of the neighborhood relation by adding an edge in both directions or only include mutual neighbors.

Definition 1.2 (Kernel)**.** *A kernel $k : X \times X \rightarrow \mathbb{R}$ on a data set X is a function that defines edge weights for matrix W in the weighted graph. It has the following properties:*

- *symmetric:* $k(x, y) = k(y, x)$
- *positivity preserving:*$k(x, y) \geq 0$
- *represents similarity between points in X*

1.3 Locally Linear Embedding

Locally Linear Embedding (LLE, [9], [10]) is an unsupervised learning algorithm which finds an embedding Ψ (see equation 1) which preserves neighborhood relations. The computation consists of an eigenvalue problem which can be computed efficiently.

Similar to the general case described above, LLE has the following three specific steps shown in figure 3.

1. Compute the similarity graph, using one of the three methods (ϵ,kNN,global). If several unconnected groups exist, perfrom the next two steps on each connected component.

[1]Also called adjacency graph, if it only indicates an edge by binary values.

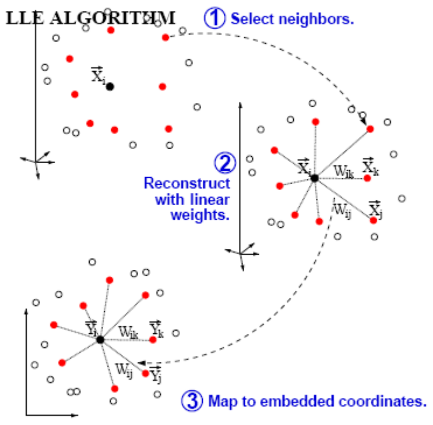


Figure 3: Visualization of the three steps of Locally Linear Embedding ([10])

2. For each data point X_i find the weights w_{ij} that best reconstruct, i.e. which minimize the constrained least squares problem:

$$E(W) = \sum_i |X_i - \sum_j X_j W_{ij}|^2$$

subject to:$\sum_j w_j = 1$ and $w_j > 0$ iff vertex j is a neighbor.

3. Find embedded vectors Y which are best reconstructed by the weights of step 2 by minimizing:

$$E(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2$$

[10] has more details on how this sparse $N \times N$ eigenvalue problem is solved.

Notice that in step 3 the weights are fixed and the output Y is optimized based on locally linear reconstruction errors. Hence, the geometry of nearby inputs is preserved, but the set of these neighborhoods overlap. If data points are weakly connected, exhibit noise or are under-sampled, the coupling between points which are far away is underestimated. This leads to points which are distant in the original space X, but nearby in the embedding space Y.

In contrast to LLE which tries to preserve local geometric properties, the Isomap method aims at preserving global properties of the manifold.

1.4 Isomap

Isomap ([11]) is a non-linear generalization of multidimensional scaling (MDS) where similarities are defined through geodesic distances, i.e. the path along the manifold. MDS ([3]) tries to find a low-dimensional projection that preserves pairwise distances by finding the eigenvectors of the distance matrix. Again, the algorithm consists of three steps:

1. Compute the similarity graph.
2. Use Dijkstra's algorithm to compute the shortest path for all pairs of points.
3. Apply MDS to embed data into d-dimensional space Y such that geodesic distances are preserved.

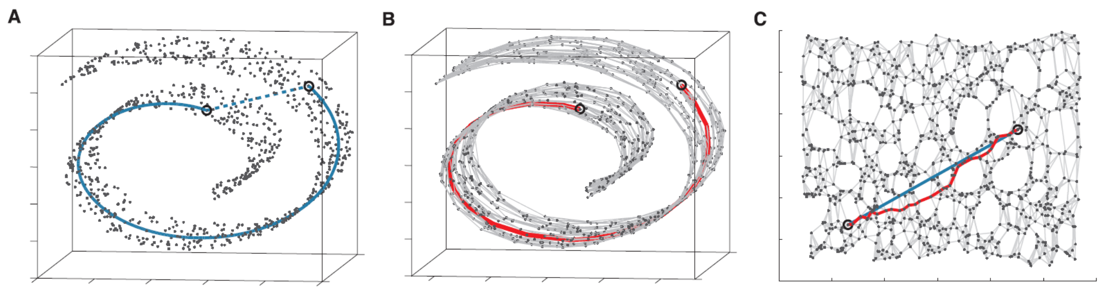


Figure 4: The Swiss roll data set. (A) shows that the Euclidean distance between two points does not reflect their similarity on the manifold. (B) shows the geodesic path calculated in step 2. (C) displays the 2-dimensional embedding defined by Isomap.

In contrast to LLE, Isomap is governed by the geodesic distances between distant points. In other words, the embedding Ψ preserves the distances of even faraway points. This often leads to distortions in local neighborhoods.

Another disadvantage of Isomap is its speed which is quite low due to the complexity of MDS.

1.5 Laplacian Eigenmaps

Laplacian Eigenmaps ([1]) are similar to LLE in that they try to preserve distance relations and that they can be solved by one sparse eigenvalue problem. However, they additionally reflect the geometric structure of the manifold by approximating the Laplace-Beltrami operator using the weighted Laplacian of the similarity graph. It has to be noted that this holds only, if the data on the manifold is uniformly sampled.

Let us first define graph Laplacians which are needed to understand the algorithm. Following the structure in [1], we explain the algorithm and show in what sense the embedding Ψ is optimal.

1.5.1 Graph Laplacians

Graph Laplacians are the central tool of spectral graph theory. There are several definitions in the literature, we will focus on the following two. Both are based on the similarity graph build in the first step of all the presented non-linear algorithms.

Definition 1.3 (Unnormalized graph Laplacian)**.**

$$L = D - W \tag{2}$$

W is defined similarly in section 1.2 as the symmetric weight matrix with positive entries for edge weights between vertices. If $w_{ij} = 0$, then vertices i and j are not connected.

The degree matrix D is (by the entries on its) diagonal: $d_{ii} = \sum_{j=1}^{n} w_{ij}$ and $d_{ij} = 0 \quad \forall i \neq j$

The multiplicity of the eigenvalue 0 of L equals the number of connected components of the graph (see [12] for a proof).

Definition 1.4 (Normalized graph Laplacian (random walk))**.**

$$L_{rw} = D^{-1}L = I - D^{-1}W \tag{3}$$

Important properties are:

(i) λ is an eigenvalue of L_{rw} with eigenvector v if and only if λ and v solve the generalized eigenproblem $Lv = \lambda Dv$ (See appendix B for proof.)

(ii) L_{rw} is positive semi-definite with the first eigenvalue $\lambda_1 = 0$ and the constant one vector $\mathbb{1}$ the corresponding eigenvector.

(iii) All eigenvectors are real and it holds that: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

1.5.2 The Algorithm

Again, we compute an embedding Ψ in three steps:

1. Build undirected similarity graph $G = (V, E)$.
2. Choose a weight matrix W either by simply setting $W_{ij} = 1$ for all connected vertices or using a heat kernel with parameter t: $w_{ij} = \exp(-||x_i - x_j||^2/t)$ If the graph is not fully connected, proceed with step 3 for each connected component.
3. Find the eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$ and eigenvectors $v_1, \ldots, v_n$ of the generalized eigenvalue problem: $Lv = \lambda Dv$. Define the embedding: $\Psi : x_i \to (v_2(i), \ldots, v_d(i))$

1.5.3 Algorithm Justification

The goal of the embedding function Ψ is to leave relative distances intact. Nearby points in D dimensions, or points connected by a strongly weighted edge in G should be mapped to a d-dimensional Euclidean space in which they are also close.

Let us now consider the case where $d = 1$ (i.e. $Y = \mathbb{R}$) and all points in the graph are connected. A reasonable optimality criterion to minimize would be:

$$\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 w_{ij} \tag{4}$$

This objective function penalizes points that are close in observation space (and hence have a large weight) but are mapped far apart in Y. Remember that y_i is just a real in this case.

Proposition 1.5. *This objective function can be rewritten in a shorter form where $y \in \mathbb{R}^n$*

$$y^T L y = \frac{1}{2}\sum_{i,j}(y_i - y_j)^2 w_{ij} \tag{5}$$

Which shows that L is positive semidefinite.

Proof. By definition, the unnormalized Laplacian is $L = D - W$. Hence,

$$y^T L y = y^T (D - W) y = y^T D y - y^T W y.$$

Recall also that $d_i = \sum_{j=1}^n w_{ij}$. Therefore,

$$= \sum_{i=1}^n y_i^2 d_i - \sum_{i,j=1}^n y_i y_j w_{ij} = \frac{1}{2}\sum_{i=1}^n y_i^2 d_i - \sum_{i,j=1}^n y_i y_j w_{ij} + \frac{1}{2}\sum_{j=1}^n y_i^2 d_j$$

$$= \frac{1}{2}\left(\sum_{i=1}^n y_i^2 \sum_{j=1}^n w_{ij} - 2\sum_{i,j=1}^n y_i y_j w_{ij} + \sum_{i=1}^n y_i^2 \sum_{j=1}^n w_{ij}\right)$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2.$$

Since the weights are all non-negative, we see that

$$y^T L y \geq 0,$$

, which proves that L is positive semidefinite. $\square$

We conclude that the following minimization problem will lead to an optimal embedding.

$$\underset{\substack{y \\ y^T D y = 1}}{\operatorname{argmin}}\, y^T L y \tag{6}$$

Where we constrained one degree of freedom by the additional constraint in order to remove an arbitrary scaling in the embedding. The larger d_{ii}, the more edges vertex i has and the more important it becomes for the minimization.

This optimization problem is solved by the eigenvector corresponding to the smallest eigenvalue of the generalized eigenvalue problem,

$$Ly = \lambda D y. \tag{7}$$

See appendix B for a proof.

Now that we proved the case for $d = 1$ let us consider the general case of $d = m$. The embedding is then defined as the matrix $Y \in \mathbb{R}^{k \times m}$ where the ith row corresponds to the embedding of the ith vertex. Since now $y_i \in \mathbb{R}^m$, we need to take the squared distance to find the optimal embedding

$$trace(Y^T L Y) = \sum_{i,j} ||y^{(i)} - y^{(j)}||^2 w_{ij}. \tag{8}$$

Adding a constraint that prevents an embedding into a space of less than $m-1$ dimensions, we get

$$\underset{Y^T D Y = I}{\operatorname{argmin}}\ trace(Y^T L Y),$$

where $rank(Y) = m$. Hence, the problem is reduced to the same eigenvalue problem as before.

1.5.4 Eigenmaps - Conclusion

Laplacian Eigenmaps are a special case of diffusion maps. This special case handles only manifolds from which the data is sampled uniformly, something that rarely happens in real machine learning tasks. Since the Laplacian only converges to the Laplace-Beltrami operator, if this condition is met, we will defer the discussion of the continuous case to the next section.

2 Diffusion Maps

Diffusion maps ([2])are another technique for finding meaningful geometric descriptions for data sets even when the observed samples are non-uniformly distributed. Coifman and Lafon provide a new motivation for normalized graph Laplacians by relating them to diffusion distances. These distances give different multiscale geometries depending on how often the random walk matrix is iterated.

[2] uses the same notion of kernels (defined in section 1.2) and almost the same random walk graph Laplacians as defined in previous sections. There is one difference though. Before we

defined $L_{rw} = D^{-1}L = I - D^{-1}W$, we now call $P = D^{-1}W$ the diffusion operator[2], where each entry $p_{ij} = k(x_i, x_j)/d(i)$ is viewed as the transition kernel of the Markov chain on G. In other words p_{ij} defines the transition probability of going from state i to j in one time step. Thereby P defines the entire Markov chain and P^t gives the probability of transition from each point to another in t times steps.

2.1 Intuition

The idea is that the transition probabilities that are defined by P reflect the local geometry of the data. The higher t, the power of P, the further a probability weight can diffuse to other points further away. In such a framework, a cluster is a region in which the probability of escaping this region is low. Figure 5 illustrates this idea. It shows a matrix P at different scales and on the left one line of this matrix. One can see that the larger t the coarser the implicit clustering.

2.2 Diffusion Distance

The goal is to relate the spectral properties of a Markov chain (more precisely its matrix and its eigenvalues and eigenvectors) to the geometry of the data. For this purpose, we define the diffusion distance D.

Definition 2.1 (Diffusion Distance)**.** *A family of diffusion distances $\{D_t\}_{t\in\mathbb{N}}$ at time t is defined as:*

$$D_t^2(x,y) \triangleq \|p(z,t|x) - p(z,t|y)\|_w^2 = \sum_z \left(p(z,t|x) - p(z,t|y)\right)^2 w(z), \tag{9}$$

where $p(z,t|x)$ is the probability that the random walk that started at x arrived at z after t steps.

This definition is very intuitive. Two points are closer, the more short paths (with large weights) connect them. By subtracting the posterior inside the square we gain a probabilistically defined distance measure.

$D_t(x,y)$ will be small, if there is a large number of short paths between x and y. This is shown in figure 6, where we have a small diffusion distance between B and C, but a large one between A and B. Intuitively, if an edge was a pipe and its weight the width of the pipe, then heat would be more likely to diffuse to point C than to point A. Additionally, it would spread quickly inside a cluster but not leave the cluster easily.

As shown in [2], diffusion distances can be computed using eigenvectors ψ_l and eigenvalues λ_l of P:

$$D_t(x,y) = \left(\sum_{l\geq 1} \lambda_l^{2t}(\psi_l(x) - \psi_l(y))^2\right)^{\frac{1}{2}}. \tag{10}$$

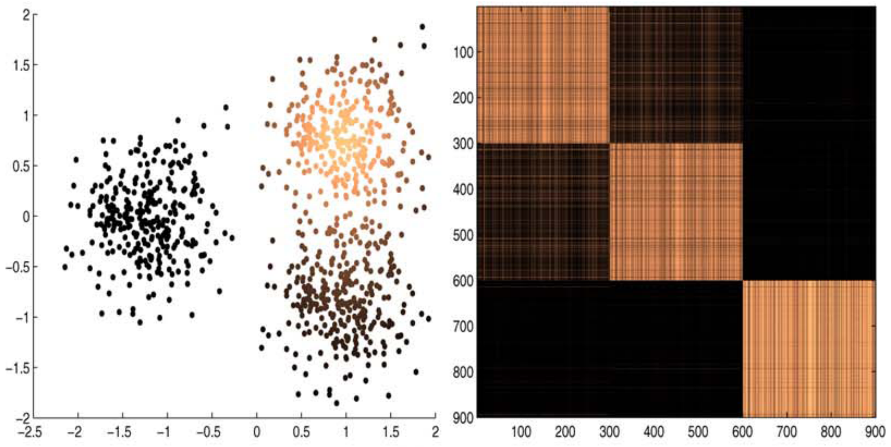
The proof uses the spectral theorem in the Hilbert space and the fact that the eigenfunctions of P are orthonormal[3]. Exploiting the fact that

$$1 = \lambda_0 > |\lambda_1| \geq |\lambda_2| \geq \dots \tag{11}$$

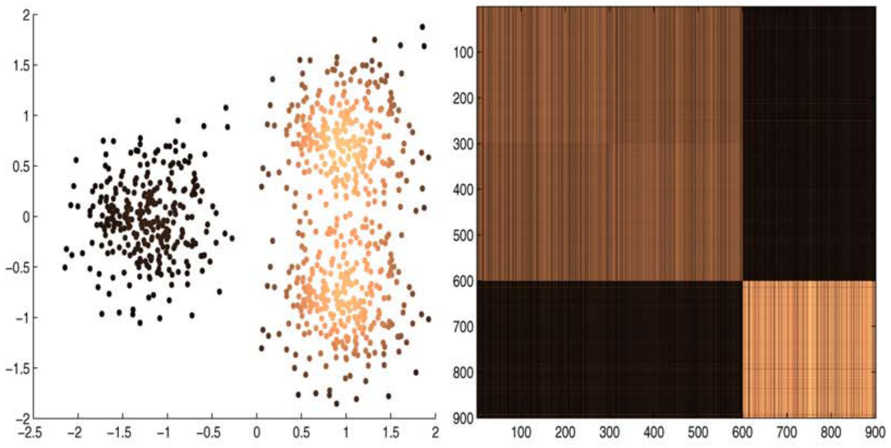
the distance may be approximated with the first s eigenvalues.

[2]Hence, the eigenvalues λ_P of P are $\lambda_P = 1 - \lambda_L$, where λ_L are the eigenvalues of the random walk graph Laplacian. This follows from $(I - P)v = \lambda v$

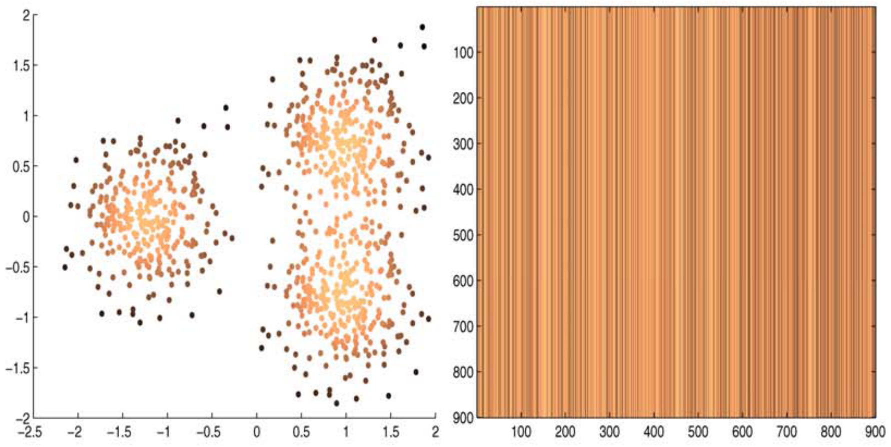
[3]In the next section we will investigate the continuous case and analyze such eigenfunctions.



(a) $t = 8$



(b) $t = 64$



(c) $t = 1024$

Figure 5: Diffusion at time $t = 8$, $t = 64$, $t = 1024$. Left: color from one row of P^t. All clusters have merged after 1024 time steps

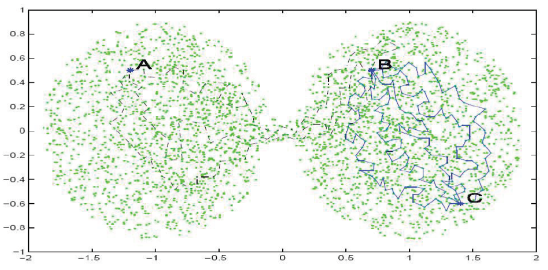


Figure 6: Example paths for diffusion distance [8]

2.3 Embedding

Using equation 10 and 11, we can now define the embedding Ψ for diffusion maps:

Definition 2.2 (Diffusion Map)**.** *A diffusion map $\Psi_t(x) : X \to \mathbb{R}^s$ is defined by:*

$$\Psi_t(x) \triangleq \begin{pmatrix} \lambda_1^t \psi_1(x) \\ \lambda_2^t \psi_2(x) \\ \vdots \\ \lambda_s^t \psi_s(x) \end{pmatrix}. \tag{12}$$

Proposition 2.3. *The diffusion map $\Psi_t : X \to Y$ embeds the data into the Euclidean space $Y = \mathbb{R}^s$ in which the distance is approximately the diffusion distance:*

$$\|\Psi_t(x) - \Psi_t(y)\| = D_t(x, y) + O(t, s), \tag{13}$$

where the big O-notation means that the embedding is equal up to a relative accuracy.

Compared to Laplacian eigenmaps, we notice only one difference in the final embedding: the scaling of each eigenvector by its corresponding eigenvalue. This leads to a smoother mapping since higher eigenvectors are attenuated.

Appendix C shows an implementation of diffusion maps in Matlab.

2.4 Conclusion

To conclude this section, let us quickly sketch the application of diffusion maps to spectral clustering algorithms.

1. Construct similarity graph
2. Compute normalized Laplacian
3. Solve generalized eigenvector problem, $Lu = \lambda Du$.
4. Define the embedding into k-dimensional Euclidean space **via diffusion maps**
5. Cluster points $y_i \in \mathbb{R}^k$ with k-means

In this section, we have investigated diffusion maps as an alternative to Laplacian eigenmaps. The actual difference of the embedding is only a weighting of the eigenvectors. [2] provides a new interpretation of the random walk graph Laplacian by relating it to diffusion processes.

3 Anisotropic Diffusion

While diffusion maps work on any kind of graph, we now turn to the special case where data sets approximate Riemannian submanifolds in $\mathbb{R}^D$. Our goals are to (i) find a low-dimensional embedding Ψ and (ii) to recover the manifold structure regardless of the sampling distribution. In particular, the question that anisotropic diffusion will answer is: *How do the density of sampled points and the geometry of the manifold on which the data points are assumed to lie influence the embedding given by the spectrum and eigenfunctions of the diffusion?*

Since Laplacian Eigenmaps use the assumption that the data is sampled uniformly on the manifold their findings are not easily applied to general machine learning tasks. Figure 7 illustrates a manifold with a nonuniform density.

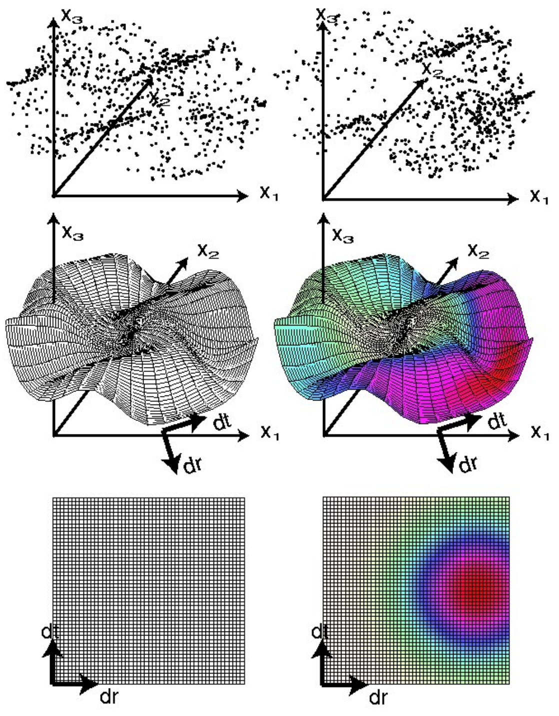


Figure 7: Manifold with nonuniform density [7], Top: samples, middle: the continuous manifold, botton: the sampling density.

We will present a new set of weights for the normalized graph Laplacian introduced by [2], which will solve this problem by adjusting the influence of the density by means of one parameter. After presenting the solution for the discrete case we give the interpretation for the continuous setting where we see the random sample as an approximation of the continuous manifold. Since the graph Laplacian approximates the Laplace-Beltrami operator under certain conditions, the geometry and the density can be completely decoupled, allowing to recover the geometry regardless of the given density of the data.

3.1 Family of Anisotropic Diffusions

To facilitate the understanding of the continuous case, let us first introduce a new family of anisotropic diffusion processes parameterized by one parameter α. We compare the new case to the standard one for creating the Laplacian:

For building the standard normalized graph Laplacian, we perform the following steps:

1. Create neighborhood graph and fix kernel $k(x_i, x_j)$
2. Build initial weight matrix W: $w_{ij} = k(x_i, x_j)$ if ith and jth vertex are connected, otherwise $w_{ij} = 0$
3. $d_i = \sum_{j=1}^{n} w_{ij}$
4. $P = D^{-1}L$

For creating an anisotropic normalized graph Laplacian, we have an additional renormalization step in 3. Otherwise, the method does not change.

1. Create neighborhood graph and fix kernel $k(x_i, x_j)$
2. Build initial weight matrix W: $w_{ij} = k(x_i, x_j)$ if ith and jth vertex are connected, otherwise $w_{ij} = 0$
3. Renormalize weight into new anisotropic kernel matrix $W^{(\alpha)}$:

$$q_i = \sum_{j=1}^{m} k(x_i, x_j) \tag{14}$$

$$w_{ij}^{(\alpha)} = \frac{w_{ij}}{q_i^{\alpha} q_j^{\alpha}} \tag{15}$$

4. $d_i^{(\alpha)} = \sum_{j=1}^{n} w_{ij}^{(\alpha)}$
5. $P^{(\alpha)} = D^{-1}L$

See appendix D for a matlab implementation of this extra step.

For $\alpha = 0$, we get the random walk Laplacian as before. The interesting case arises for $\alpha = 1$, where the influence of the distribution no longer influences the embedding and the geometry of the dataset may be recovered. It is this case, for which the discrete samples converge to the Laplace-Beltrami operator, given several conditions.

Let us now investigate how points on a submanifold of $\mathbb{R}^n$ may be used for an approximation of the Laplace-Beltrami operator. This case is especially fascinating, because it proves that we may recover the Riemannian geometry of the data set, no matter the sampling distribution.

3.2 Laplace-Beltrami Operator

Before we go into the details of Laplace-Beltrami operators and how we arrive at those from discrete samples, we need to introduce additional terms.

Definition 3.1 (Hilbert Space)**.** *A Hilbert space is an inner product space X on a space S that is complete under the norm $\|f\| = \sqrt{\langle f, f\rangle}$ defined by the inner product $\langle \cdot, \cdot \rangle$ and where $\|f\|_2 < \infty$.*

One example for such a norm is the L_2 norm: $\langle f, g\rangle = \int_S f(x)g(x)dx$. Given such a space, we now have the ability to define functions f using a function basis (Φ_i):

$$f = \sum \alpha_i \Phi_i(x). \tag{16}$$

The notion of an orthonormal basis is similar to that in a vector space: $||\Phi_i|| = 1, \ \ \forall i = 1, \ldots, n$ and $\langle \Phi_i, \Phi_j \rangle = 0, \ \ \forall i \neq j$.

An operator $L : X \to X$ is a function of functions. One example of such an operator is the Laplace operator.

Definition 3.2 (Eigenfunction)**.** *An eigenfunction of an operator is defined like the eigenvector of a matrix in vector space:*

$$Lf = \lambda f. \tag{17}$$

Figure 8 shows an example of such eigenfunctions for a dumb-bell shaped manifold.

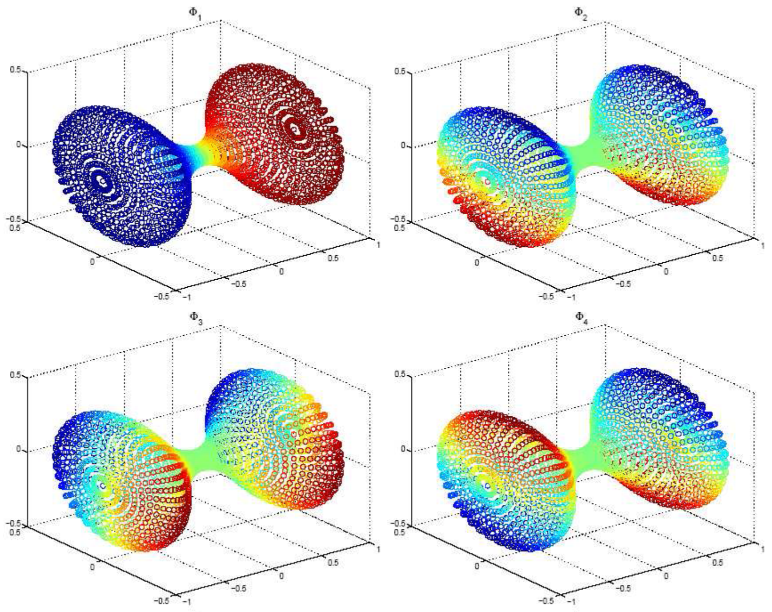


Figure 8: First 4 eigenfunctions of a dumb-bell shaped manifold and corresponding diffusion map ([8]).

Definition 3.3 (Hermitian Operator)**.** *A linear operator is Hermitian (symmetric) if*

$$\langle Lf, g \rangle = \langle f, Lg \rangle$$

Eigenfunctions of Hermitian operators form an orthonormal basis of the Hilbert space X on a compact domain.

Definition 3.4 (Laplace Operator Δ)**.** *The Laplace operator is a differential operator defined as the divergence of the gradient. It is defined in the n-dimensional Euclidean space. Given a twice-differentiable real-valued function f its definition is:*

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

The Laplace operator possesses several interesting properties:

- Its first eigenfunction is constant
$$\frac{\partial^2 c}{\partial x_i^2} = 0$$
- Sine and cosine are the second eigenfunctions, since they are only scaled when the operator is applied to them:
$$(\sin(\omega x))'' = -\omega^2 \sin(\omega x)$$
and they change signs so:
$$\langle c, \sin \rangle = 0,$$
where the sine fucntion was the argument.
- The eigenfunctions of Δ form an orthonormal basis in X.

Definition 3.5 (Laplace-Beltrami Operator)**.** *Laplace-Beltrami operator is an extension of normal Laplacians to manifolds.*

In order to make statements about the Laplace-Beltrami operators for a manifold, we have to overcome one major Xobstacle: We only have a finite sample from a probability measure p on an d-dimensional submanifold M in $\mathbb{R}^D$. Hence, we are usually in a discrete setting.

However, in [5, 6] Hein et. al. proved that given several technical conditions on the kind of submanifold, the kernel and the density:

Theorem 3.6. *If neighborhood $h \to 0$ and the number of data points in it $n \to \infty$ and $nh^{m+2}/\log n \to \infty$, then the random walk Laplacian converges to the weighted Laplace-Beltrami operator*
$$\lim_{n\to\infty} (L_n^{(rw)} f)(x) \sim -(\Delta_s f)(x), \tag{18}$$
where the weighted Laplace-Beltrami operator
$$\Delta_s = \Delta_M + \frac{s}{p}\langle \nabla p, \nabla f \rangle, \tag{19}$$
is the natural generalization of the Laplace-Beltrami operator. It is used, if the manifold has a non-uniform probability distribution. In these cases, it induces an anisotropic diffusion towards or away from increasing density depending on s.

To relate the weighted Laplace-Beltrami operator to the anisotropic diffusion introduced before, we note that $s = 2(1-\alpha)$. Hence, the influence of the density is zero, if $\alpha = 1 \leftrightarrow s = 0$.

3.3 Influence of Density and Geometry

Now that we have established the connection between the random walk Laplacian and the Laplace-Beltrami operator we can analyze how both are related. Their common trait is that they generate a diffusion process. The graph Laplacian generates it on a graph, while the Laplace-Beltrami operator does the same on a manifold.

We recall that the general Laplacian eigenmaps minimize
$$y^T L y = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2. \tag{20}$$

For the general normalized Laplacian, the continuous counterpart is:
$$S(f) = \int_M ||\nabla f||^2 dV. \tag{21}$$

By a similar application of the Rayleigh-Ritz theorem, we use the eigenfunction which minimizes:

$$\lambda_{min} = \underset{f}{\operatorname{argmin}} \frac{\int \|\nabla f\|^2 dx}{\int f^2 dx}. \tag{22}$$

In section 3.1 we have established that the second eigenfunction of the Laplace operator is the sine or cosine function. In order to get a better intuition, figure 9 (left) shows such an eigenfunction, created from the symmetric normalized graph Laplacian of two Gaussians. In this figure, the connection between points of the second eigenvector suggests that we approximate the eigenfunction through enough sampled points.

The interesting observation now is that weighted Laplace-Beltrami operator induces a different smoothness functional. Namely one that incorporates the density:

$$S(f) = \int_M ||\nabla f||^2 p^s dV \tag{23}$$

For $s > 0$ this functional heavily penalizes functions that are non-smooth in areas of high density. Figure 9 (right) shows exactly such a situation with the second eigenfunction of a random walk Laplacian. In the area of high density of both Gaussians, the function is extremely smooth.

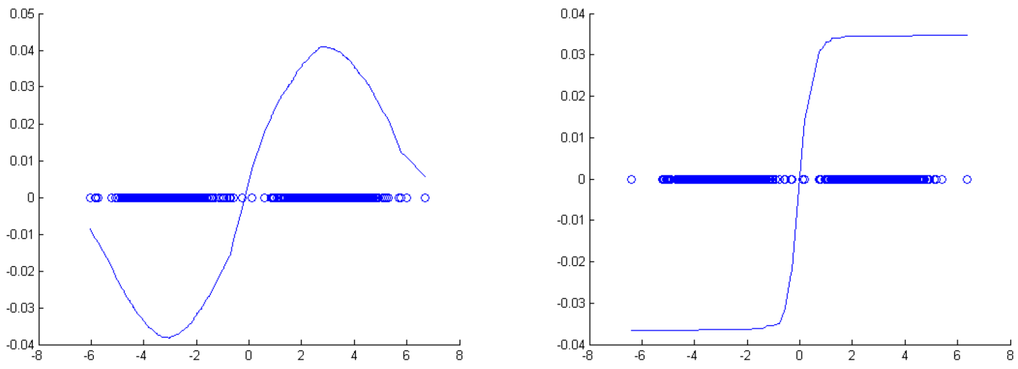


Figure 9: Simple example of 2 dense Gaussians. Left: the line shows the second eigenfunction of the corresponding symmetric graph Laplacian. Right: The random walk Laplacian considers the density of the points.

Through the parameter s in the second term of the Laplace-Beltrami operator $\Delta_s = \Delta_M + \frac{s}{p}\langle\nabla p, \nabla f\rangle$, one can induce an anisotropic diffusion, resulting in smooth leveled eigenfunctions in regions of high density. This is especially desirable in semi-supervised learning where one wants to find similar labels inside a cluster of high density.

Another example of how geometry and density influence the embedding with eigenvectors is given in [2]. Figure 10 shows how the geometry of the manifold is completely recovered, despite the different densities along the curves. See [2] for further details.

4 Conclusion

In this report, we gave an introduction to diffusion maps and anisotropic diffusion. We introduced a family of weighted Laplace-Beltrami operators that allow a scaling of the influence of the

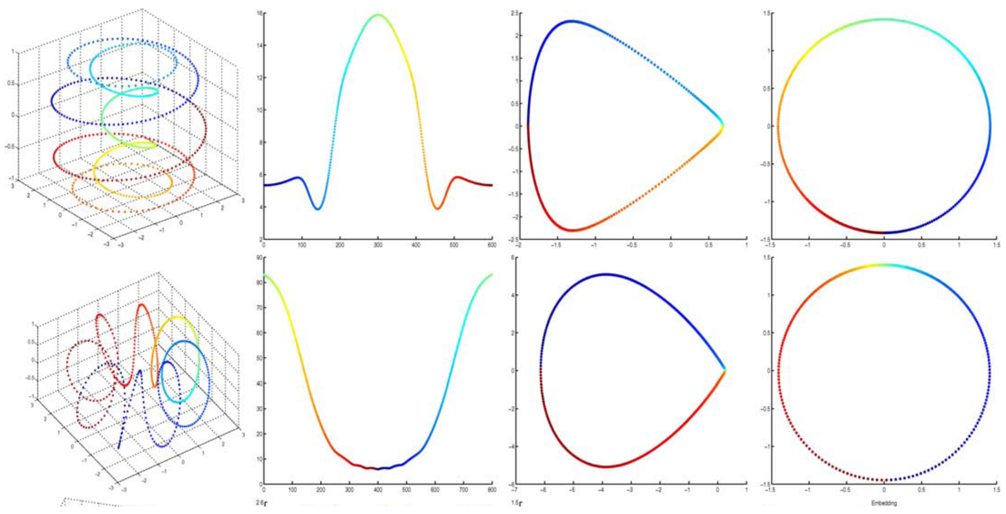


Figure 10: From left to right: original curves, the densities of points, the embeddings via the graph Laplacian ($\alpha = 0$) and the embeddings via the LaplaceBeltrami approximation ($\alpha = 1$). In the latter case, the curve is embedded as a perfect circle and the arclength parametrization is recovered.

density via one parameter s. This family of diffusion operators allows the complete separation of the distribution of the data from the geometry of the underlying manifold.

We have also shown the superiority of nonlinear methods for some data sets. Further assessment of these methods applied to difficult and noisy manifolds is necessary.

A PCA code and mapping to Eigenvector

Code for producing figure 1 and 2. Performs PCA and projection to optimal eigenvector.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Gaussian data points in x1 are generated with randn and
% mean vector: [2 2] and covariance: [4 4;2.5 4]
% for eigenvectors
[n, p] = size(x1);
% subtract mean from each row
B = x1 - repmat(mean(x1,1), n, 1);
[V,D] = EIGS(B * B' / (n - 1));
% [V,D] = EIGS(A) returns a diagonal
% matrix D of A's 6 largest magnitude
% eigenvalues and a matrix V whose
% columns are the corresponding eigenvectors.

X=x1';
ColorVector = X(:,1)+X(:,2);
figure(1)
    hold on
    axis('square');
```

```
%     plot(x1(1,:),x1(2,:),'.r')
%     plot(x2(1,:),x2(2,:),'.g')
%     plot(x3(1,:),x3(2,:),'.b')
     scatter(X(:,1),X(:,2),12,X(:,1),'filled');
%     title('2 dimensional gaussian samples'),
     xlabel('x1'),ylabel('x2');
     %PCA eigenvectors
     arrow(middle+[0 0],middle+3.*V(:,1)',14,'BaseAngle',60);
     arrow(middle+[0 0],middle+3.*V(:,2)',14,'BaseAngle',60);
hold off;

% projection
P = V(:,2)*V(:,2)';
pX = P*X';
pX = pX';

figure(2)
hold on
axis('square');
scatter(pX(:,1),pX(:,2),12,ColorVector,'filled');
hold off;

%%%%%%%%%%%%%%%%%%%%%%%
% Roll
tt = (3*pi/2)*(1+2*rand(1,N));

X = [tt.*cos(tt); tt.*sin(tt)]';
ColorVector = tt';

[n, p] = size(X);
% subtract mean from each row
B = X - repmat(mean(X,1), n, 1);
[V,D] = EIGS(B' * B / (n - 1));

figure(3)
hold on
     axis('square');
     scatter(X(:,1),X(:,2),12,ColorVector,'filled');
     arrow([0 0],5.*V(:,1)',14,'BaseAngle',60);
     arrow([0 0],5.*V(:,2)',14,'BaseAngle',60);
hold off;

% projection
P = V(:,2)*V(:,2)';
pX = P*X';
pX = pX';

figure(4)
hold on
axis('square')
```

```
scatter(pX(:,1),pX(:,2),12,ColorVector,'filled');
hold off;
```

B Rayleigh Ritz Proof

Proposition B.1. *The optimization problem*

$$\underset{\substack{y \\ y^T D y = 1}}{\operatorname{argmin}} \, y^T L y \tag{24}$$

is solved by finding the eigenvector corresponding to the smallest eigenvalue of the general eigenvalue problem.

$$Ly = \lambda D y \tag{25}$$

If one multiplies y^T to the left of both side of equation 25, one immediately sees the connection between the original optimization problem and the general eigenvalue problem. However, for proving this proposition the original formulation is better.

We know L is symmetric and positive semidefinite. Hence, its eigenvectors u_i where $Lu_i = \lambda_i u_i$ form an orthonormal basis and L has the eigenvalue decomposition:

$$L = \sum_{i=1}^{n} \lambda_i u_i u_i^T$$

Without loss of generality, let us assume that $D = I$, the identity matrix. Therefore, we get $\|y\| = 1$ as the constraint. Hence,

$$\underset{\substack{y \\ y^T y = 1}}{\operatorname{argmin}} \, y^T L y = \underset{\substack{y \\ y^T y = 1}}{\operatorname{argmin}} \sum_{i=1}^{n} \lambda_i < u_i, y >^2 = \lambda_{min} \tag{26}$$

C Implementation of Laplacian Eigenmaps

All following code shows modifications to the *GraphDemo* of Matthias Hein and Ulrike von Luxburg: `http://www.ml.uni-saarland.de/GraphDemo/GraphDemo.html`.

Notice that we subtract the eigenvalues from 1, this is due to the different definition of the graph Laplacian. See section 2 for details.

```
%Compute eigenvectors of random walk graph Laplacian
[eigvecs,eigvals] = computeEigvectors(Lrw);

% ———————————————————————————————————————————
% Diffusion Maps
% diffMapT is the power of Lrw
% matrix and influences the diffusion distance.
% Larger diffMapT result in coarser clusters.
% Essentially, this weighs the eigenvectors
% with their corresponding eigenvalues.
  eigvalsT = (1-eigvals).^diffMapT;
   for i=1:num_eigvecs
```

```
      eigvecs(:,i)=eigvecs(:,i).*eigvalsT(i);
         end
% ----------------------------------------------------------------------
```

D Implementation of Anisotropic Diffusion

This is the extra step which can be implemented to change the influence of the density on the embedding.

```
% ----------------------------------------------------------------------
% Anisotropic Diffusion
% alpha=0: normal Laplacian, maximum influence of density
% alpha=0.5: Markov chain is an approximation
% of the diffusion of a Fokker-Planck equation
% alpha=1: approximation of Laplace-Beltrami operator,
% if data lies in submanifold
% Normalize kernel matrix W by dividing with the density
if (alpha>0)
    q=zeros(num_points,1);
    q=sum(W,2);
    Q = spdiags((1./q.^alpha),0,num_points,num_points);
    W = Q*W*Q;
end
% ----------------------------------------------------------------------
% Continue with general normalization of kernel
```

E Implementation of Eigenfunctions for Symmetric and Random Walk Laplacian

This matlab code samples from two Gaussians, creates the symmetric graph Laplacian, calculates its eigenvectors and draws the second eigenvector. It may also transform the eigenvectors from that of the symmetric Laplacian to the ones of the random walk.

```
numEach = 400;

r = -3 + randn(numEach,1);
s = 3 + randn(numEach,1);
t = [r;s]
t=sort(t);

dist2 =DistEuclideanPiotrDollar(t,t); % squared distances
K = exp(-1/(2*2^2)*dist2).*(dist2 < 1.5^2 & dist2~=0);

numPoints = length(t);
d=zeros(numPoints,1);
d=sum(K,2);
d(find(d==0)) = 1/numPoints;
```

```
% L_sym = D^{-1/2} L D^{-1/2}.
Dsqrt =spdiags(1./(d.^0.5), 0,numPoints,numPoints);
A = (speye(numPoints)-Dsqrt*K*Dsqrt);

[evecs,evals]=eig(A);
%--------------------
% to get the eigenvectors of L_rw and not L_sym
num_eigvecs = length(evals);
  for i=1:num_eigvecs
    evecs(:,i)=evecs(:,i)./sqrt(d);
    evecs(:,i)=evecs(:,i)/norm(evecs(:,i));
  end
%--------------------

hold on;
    scatter(t,zeros(numPoints,1));
    plot(t,evecs(:,2));
hold off;
```

References

[1] Mikhail Belkin and Partha Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Comput. **15** (2003), no. 6, 1373–1396.

[2] Ronald R. Coifman and Stephane Lafon, *Diffusion maps*, Applied and Computational Harmonic Analysis **21** (2006), no. 1, 5–30.

[3] T. F. Cox and M. A. A. Cox, *Multidimensional scaling*, Chapman & Hall, London, 1994.

[4] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning*, Springer, August 2001.

[5] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg, *Graph laplacians and their convergence on random neighborhood graphs*, Aug 2006.

[6] Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg, *Graph laplacians and their convergence on random neighborhood graphs*, J. Mach. Learn. Res. **8** (2007), 1325–1370.

[7] Erik G. Learned-Miller, *Manifold picture*, `http://www.cs.umass.edu/~elm/papers_by_research.html`.

[8] Mauro Maggioni, *Laplacian and wavelet bases for value function approximation and their connection to kernel methods*, ICML Workshop, June 2006.

[9] Sam T. Roweis and Lawrence K. Saul, *Nonlinear dimensionality reduction by locally linear embedding*, Science **290** (2000), no. 5500, 2323–2326.

[10] Lawrence K. Saul and Sam T. Roweis, *Think globally, fit locally: unsupervised learning of low dimensional manifolds*, J. Mach. Learn. Res. **4** (2003), 119–155.

[11] J. B. Tenenbaum, V. de Silva, and J. C. Langford, *A global geometric framework for nonlinear dimensionality reduction.*, Science **290** (2000), no. 5500, 2319–2323.

[12] Ulrike von Luxburg, *A tutorial on spectral clustering*, Tech. report, Max Plank Institute for Biological Cybernetics, August 2006.