Multiscale Dimensionality Reduction Based on Diffusion Wavelets

Chang Wang, Sridhar Mahadevan
Computer Science Department
University of Massachusetts
Amherst, Massachusetts 01003
{chwang, mahadeva}@cs.umass.edu

June 29, 2009

Abstract

Many machine learning data sets are embedded in high-dimensional spaces, and require some type of dimensionality reduction to visualize or analyze the data. In this paper, we propose a novel framework for multiscale dimensionality reduction based on diffusion wavelets. Our approach is completely data driven, computationally efficient, and able to directly process non-symmetric neighborhood relationships without ad-hoc symmetrization. The superior performance of our approach is illustrated using several synthetic and real-world data sets.

1 Introduction

In this paper, we propose a novel framework for multiscale dimensionality reduction, which has the added benefit of being able to handle non-symmetric neighborhood relationships. Similar to Laplacian eigenmaps [1] and LPP [5], our approach also represents the set of instances by vertices of a graph, where an edge is used to connect instances x and y using a distance measure, such as if y is among the k-nearest neighbors of x. The weight of the edge is specified typically using either a symmetric measure, such as the heat kernel or a non-symmetric measure, such as a transition reward in a state space graph. Such pairwise similarities can be used to derive a transition probability matrix for a random walk P ($P = D^{-1}W$), where W is the weight matrix, and D is a diagonal "valency" matrix of the row-sums of W. In contrast to many previous graph-based methods, we do not require W to be symmetric. Our approach thus addresses the problem of learning multiscale low dimensional embeddings from directed graphs (undirected graphs are a special case of directed graphs) without symmetrizing them, as many previous approaches require.

Most existing dimensionality reduction approaches follow a similar idea: First, basis functions are computed that span some subspace of the original problem space that contains the functions of interest; Then, the data is re-represented using a "selected" set of basis functions, which preserve a desired relationship (like neighborhood or covariance). Classically, in Euclidean spaces, two types of "fixed" bases are widely used: wavelet bases used in wavelet analysis, and Fourier bases used in Fourier analysis. For problems on non-Euclidean spaces, which include discrete spaces such as graphs and continuous spaces such as manifolds, these fixed bases may not be effective. In particular, as the geometry of the space may be unknown and needs to be reconstructed from the data, the bases themselves need to be learned from the data. Laplacian eigenmaps constructs basis functions using the eigenvectors of the graph Laplacian [2], which extends Fourier analysis to graphs and manifolds. Recently, diffusion wavelets [4] provides a similar extension of classical wavelet analysis to graphs and manifolds.

Laplacian eigenmaps and its linear approximation LPP are well studied graph Laplacian based dimensionality reduction approaches. One limitation of these approaches is that they only yield a "flat" embedding but not a multiscale embedding. Another problem is that such eigenvector methods cannot handle the relationships characterized by directed graphs without some ad-hoc symmetrization. Some typical examples where non-symmetric matrices arise are when k-nearest neighbor relationships are used, web information retrieval based on network topology, and state space transitions in a Markov decision process. For a general weight matrix W representing the edge weights on a directed graph, its eigenvalues and eigenvectors are not guaranteed to be real. Many current approaches to this problem convert the directed graphs to undirected graphs. A simple solution is setting W to be $W + W^T$ or WW^T. A more sophisticated symmetrization method uses the directed Laplacian [3], where the symmetrization uses the Perron-Frobenius theorem. It is more desirable to find an approach that handles directed graphs without the need for symmetrization.

We present a framework for multiscale dimensionality reduction (defined in Section 3) based on diffusion wavelets. Our method (called *Diffusion Projections (DP))* automatically reveals the geometric structure of the data at different scales, and provides multiscale embedding for both symmetric and non-symmetric relationship matrices. For the symmetric case, our approach can automatically identify the most appropriate dimensions for embedding, and the resulting p_j-dimensional embedding at level j is the same as that from eigenvector based approaches (using top p_j eigenvectors) up to a rotation. For the non-symmetric case, DP is directly applied to the matrix and no symmetrization step is needed. In this paper, we apply the cost functions used in Laplacian eigenmaps and LPP to explain diffusion projections. Our algorithm also generalizes to other dimensionality reduction approaches based on eigenvectors, like PCA.

The rest of this paper is as follows. In Section 2 we motivate multiscale analysis using diffusion wavelets. In Section 3 we describe our algorithm. Section 4 contains a theoretical analysis of the algorithm. We present experimental results in Section 5. Section 6 provides some concluding remarks.

2 A Geometric Interpretation of Diffusion Wavelets Construction

Many kinds of relationship between data instances can be represented as graphs. Given a graph, the structure of the space of functions on the graph can be revealed by spectral analysis of random walks on the graph. Let $P_{x,y}$ represent the probability of transition in one time step from node x to node y which is proportional to the weight $W_{x,y}$. For $t > 0$, the probability of transition from x to y in t time steps is given by $P^t_{x,y}$. Running the chain forward in time will allow us to integrate the local geometry and therefore will reveal relevant geometric structures of data at different scales. An illustrative example is shown in Figure 1. We generated a set of 240 points in $\mathcal{R}^2$. The set is comprised of 4 clusters, each of which has 60 points. From this set, we built a graph with the heat kernel $e^{-\|x-y\|^2/2}$, and formed the related Markov random walk matrix P. In this figure, we plot P (1-D), P^8 (1-E) and P^{32} (1-F), where the block structure of these powers clearly shows the multiscale structure of the data. At scale P, the data set appears to have 4 clusters (1-A). At scale P^8, the neighborhood clusters have merged, and the data set has 2 clusters (1-B). Then at scale P^{32}, all clusters have merged together (1-C). This example shows that running the chain forward in time will reveal the intrinsic geometric structures of the data at different scales. Moreover, the effective rank of P^t decreases as t increases in a way that can be quantified: in this example the effective rank of P is 4, that of P^8 is 2 and that of P^{32} is 1, and therefore the correspondence between effective rank and number of clusters at different scales is apparent.

Figure 2 shows the diffusion wavelets routine. It constructs a compressed representation of the given matrix by representing powers of the matrix not in terms of the original (unit vector) basis, but rather a set of newly generated bases. The matrix T represents one step transition probability between data points. The t step transition probability is given by T^t. At scale j, the $\mathcal{QR}$ (a modified QR decomposition) subroutine decomposes $T^{2^{j-1}}$ into an orthogonal matrix Q and a triangular matrix R up to a precision ϵ by filtering out some high frequency information, where $T^{2^{j-1}} =_\epsilon QR$. Columns in Q are orthonormal basis functions spanning the column space of $T^{2^{j-1}}$ at scale j. RQ is the new representation of $T^{2^{j-1}}$ regarding the space spanned by the columns of Q (based on matrix invariant subspace theory). Compared to dimensionality of $T^{2^{j-1}}$'s column space, we usually get fewer basis functions, since some high frequency information has already been filtered out. $\mathcal{DWT}$ then computes the 2^j time step transition using the low frequency representation of $T^{2^{j-1}}$ resulting in a new representation of T^{2^j} and the procedure repeats.

Running $\mathcal{DWT}$ is equivalent to running a Markov chain on the input data forward in time. At scale j, the representation of $T^{2^{j-1}}$ is compressed based on the amount of remaining information and the desired precision. Two sets of basis functions are constructed: "scaling" functions span the column space of the input matrix at a given level; "wavelet" functions span the orthogonal complement space. The notations of "Scaling" and "Wavelet" functions are the

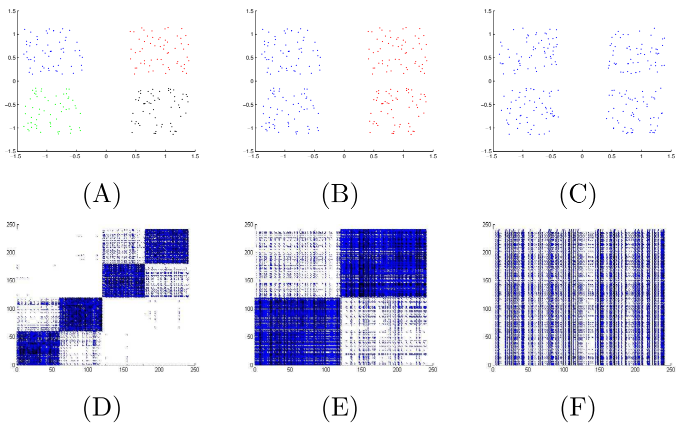


Figure 1: Diffusion at times $t = 1, 8, 32$ over a set of 4 clusters. The first 3 figures represent the set, and the colors show the cluster labels. The next 3 figures include plots of the transition matrices P (D), P^8 (E) and P^{32} (F). Points are ordered so that the first 60 are in the first cluster, the next 60 are in the second cluster and so on. The colors at position (i, j) of the matrix P^t represents the value of $P^t_{i,j}$.

```
$\{[\phi_j]_{\phi_0}, [\psi_j]_{\phi_0}\} = \mathcal{DWT}(T, \phi_0, J, \epsilon)$
{
    $For\ j = 0\ \ to\ \ J - 1$
    {
        $([\phi_{j+1}]_{\phi_j}, [T^{2^j}]_{\phi_j}^{\phi_{j+1}}) \leftarrow \mathcal{QR}([T^{2^j}]_{\phi_j}^{\phi_j}, \epsilon);$
        $[T^{2^{j+1}}]_{\phi_{j+1}}^{\phi_{j+1}} = ([T^{2^j}]_{\phi_j}^{\phi_{j+1}}[\phi_{j+1}]_{\phi_j})^2;$
        $[\phi_{j+1}]_{\phi_0} = [\phi_{j+1}]_{\phi_j}[\phi_j]_{\phi_0};$
        $[\psi_j]_{\phi_j} \leftarrow \mathcal{QR}(I < \phi_j > -[\phi_{j+1}]_{\phi_j}[\phi_{j+1}]_{\phi_j}^T, \epsilon);$
    }
}
```

Figure 2: Using diffusion wavelets to learn basis functions spanning T's column space at different scales. $[\phi_j]_{\phi_0}$: "Scaling" basis functions at scale j; $\phi_0 = [\phi_0]_{\phi_0}$ is an identity matrix; $[\psi_j]_{\phi_0}$: "Wavelet" basis functions at scale j. J: the maximum number of levels to compute. ϵ: desired precision. $\mathcal{QR}$: A modified QR decomposition. The notation $[T]_{\phi_a}^{\phi_b}$ denotes matrix T whose column space is represented using basis ϕ_b at scale b, and row space is represented using basis ϕ_a at scale a. The notation $[\phi_b]_{\phi_a}$ denotes basis ϕ_b represented on the basis ϕ_a. At an arbitrary scale j, we have p_j basis functions, and length of each function is l_j. $[T]_{\phi_a}^{\phi_b}$ is a $p_b \times l_a$ matrix, $[\phi_b]_{\phi_a}$ is an $l_a \times p_b$ matrix.

same as that used in regular wavelet transform. Scaling functions at each level are orthonormal to each other. The space spanned by the scaling functions at level j is orthogonal to the space spanned by the wavelet functions at the same level. Scaling functions at the top level and the wavelet functions over all levels together span the column space of the original matrix. Columns of $[\phi_j]_{\phi_0}$ are basis functions spanning T's column space at scale j, and are used in this paper to compute multiscale embedding results.

3 The Main Algorithm

Notations used in Section 3 and 4 are as follows: $X = [x_1, \cdots, x_n]$ is a $p \times n$ matrix representing n instances defined in a p dimensional space. W is an $n \times n$ matrix, where $W_{i,j} = e^{-\|x_i - x_j\|^2/\delta^2}$. D is a diagonal matrix, where $D_{i,i} = \sum_j W_{i,j}$. $\mathcal{W} = D^{-0.5} W D^{-0.5}$. $\mathcal{L} = I - \mathcal{W}$, where I is an identity matrix. $XX^T = FF^T$, where F is a $p \times r$ matrix of rank r. A trivial way to compute F from X is singular value decomposition. $(\cdot)^+$ represents matrix pseudo inverse.

The problem: Assuming the data instances are collected from a data manifold, graph Laplacian type dimensionality reduction approaches map the data to lower dimensional spaces preserving the local geometry (of the manifold). When we write down the cost function as $\sum_{i,j}(y_i - y_j)^2 \mathcal{W}_{i,j}$, which guarantees the neighbors in the original space are still neighbors in the new space. The solution leads to *Laplacian eigenmaps*, where the c dimensional embedding is provided by eigenvectors of $\mathcal{L}x = \lambda x$ corresponding to the c smallest non-zero eigenvalues. When we require the mapping function to be a linear mapping, the cost function becomes $\sum_{i,j}(f^T x_i - f^T x_j)^2 \mathcal{W}_{i,j}$, and the solution leads to *LPP*, where c dimensional embedding is achieved by eigenvectors of $X\mathcal{L}X^T x = \lambda XX^T x$ corresponding to the c smallest non-zero eigenvalues. As suggested by Figure 1, many data sets have non-trivial regularities at multiple levels, which correspond to their underlying intrinsic structures. A key problem that has not been addressed so far is how to compute embedding results leveraging the underlying manifold structure, resulting in multilevel results. Recall that Laplacian eigenmaps and LPP only compute embedding at one level. The first problem we solve in this paper is: given X, compute $Y_k = [y_k^1, \cdots, y_k^n]$ at level k (Y_k is a $p_k \times n$ matrix) to minimize $\sum_{i,j}(y_k^i - y_k^j)^2 \mathcal{W}_{i,j}$. Here $k = 1, \cdots, L$ represents each level of the underlying manifold hierarchy. The second problem is similar but with a linear mapping constraint (like LPP): compute multiscale linear mapping f to minimize $\sum_{i,j}(f^T x_i - f^T x_j)^2 \mathcal{W}_{i,j}$. The **algorithm** to learn multiscale lower dimensional embedding is described in Figure 3.

Some **advantages** of the proposed multiscale approach are described below:

(1) Multiscale Dimensionality Reduction: our algorithm can automatically identify the hierarchy of the data set and the best dimensionality for embedding (leveraging the underlying structure of the data). As the procedure in Figure 2

1. **Construct relationship matrix T characterizing the given data set:**
   - $T = I - \mathcal{L}$.
   - Linear mapping only: $T = (F^+ X \mathcal{L} X^T (F^T)^+)^+$.
2. **Apply diffusion wavelets to explore the intrinsic structure of the data:**
   $\{[\phi_j]_{\phi_0}, [\psi_j]_{\psi_0}\} = \mathcal{DWT}(T, I, J, \epsilon)$, where the function $\mathcal{DWT}$ is described in Section 2,
   - The resulting $[\phi_j]_{\phi_0}$ is an $n \times p_j$ matrix.
   - Linear mapping only: the resulting $[\phi_j]_{\phi_0}$ is a $p \times p_j$ matrix.
3. **Compute lower dimensional embedding (at level j):**
   - The embedding $x_i \to y_i =$ row i of $[\phi_j]_{\phi_0}$.
   - Linear mapping only: $\mathcal{A} = (F^T)^+[\phi_j]_{\phi_0}$ is a $p \times p_j$ matrix. The embedding $x_i \to y_i = \mathcal{A}^T x_i$.

Figure 3: The main algorithm (diffusion projections).

suggests, the data set is spanned by varying numbers of basis functions at different levels. These numbers reveal the dimensions of the relevant geometric structures of the data at different scales. Such a multiscale representation is the main target of both hierarchical clustering and hierarchical topic modeling in information retrieval. Laplacian eigenmaps and LPP cannot do multiscale dimensionality reduction (note: eigenvectors of T^j and T are the same).

(2) Directional Dimensionality Reduction: In contrast to all eigenvector-based approaches, diffusion projections does not require the input relationship matrix to be symmetric. For the non-symmetric case, $[\phi_j]_{\phi_0}$ spans the same space, up to a precision ϵ, spanned by the columns of T^{2^j}, which is the space of probability distributions of the random walk at time 2^j. Compared to using the directed Laplacian, which also provides low-dimensional embeddings for directed graphs, diffusion projections offer two advantages: (1) Diffusion projections is faster, since for using the directed Laplacian we need to add the cost of calculating the Perron vector of the random walk matrix P. Usually an iterative technique will be applied. When the random walk matrix is large, this step is time consuming. (2) We have one fewer parameter to tune. In the directed Laplacian, we need to carefully select the value of a "teleportation" probability parameter η used to define a strongly connected reversible transition matrix. This parameter is necessary for both correctness of the algorithm and stability.

(3) Diffusion Scaling Functions are Sparse: From [4], we know that for an input matrix T, columns of $[\phi_j]_{\phi_0}$ and $\{\xi_i : |\lambda_i|^{2^j} > \epsilon\}$ span the same space, where ξ_i and λ_i are the eigenvector and corresponding eigenvalue of T. However, these diffusion scaling functions and eigenvectors are quite different. Eigenvec-

tors have a potential drawback in that they capture global "smoothness" well, but poorly model functions which are not globally smooth but only piecewise-smooth, or have different smoothness in different regions. Unlike the "global" nature of eigenvectors, the columns of $[\phi_j]_{\phi_0}$ are usually sparse. This property makes diffusion scaling functions "interpretable" in some applications like topic modeling and object recognition. In those applications, we are interested in both the lower dimensional embedding and the "meaning" of the embedding.

4 Theoretical Results

There is an interesting connection between our algorithm, Laplacian eigenmaps and LPP. Theorem 1 shows our embedding result at level j is the same as that from Laplacian eigenmaps with top p_j eigenvectors. A similar connection between our algorithm and LPP is shown in Theorem 3. This means diffusion projections is also optimal regarding the cost functions used in previous approaches. Note that our algorithm does not use eigenvalue decomposition at all. It computes multiscale embedding with diffusion wavelets by exploring the intrinsic structure of the given data set.

Theorem 1. In the main algorithm, Laplacian eigenmaps (with eigenvectors corresponding to p_j smallest non-zero eigenvalues) and diffusion projections (at level j) return the same p_j dimensional embedding up to a rotation Q.
Proof:
In Laplacian eigenmaps, we use row i of $V_{1:p_j}$ to represent p_j dimensional embedding of x_i, where $V_{1:p_j}$ is an $n \times p_j$ matrix representing the p_j smallest eigenvectors of $\mathcal{L}$. When $T = I - \mathcal{L}$, the largest eigenvectors of T are exactly the smallest eigenvectors of $\mathcal{L}$.

Let $[\phi_j]_{\phi_0}$ represent the scaling functions of T at level j, then $V_{1:p_j}$ and $[\phi_j]_{\phi_0}$ span the same space, i.e. $V_{1:p_j}V_{1:p_j}^T = [\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T$ [4]. Since the columns of both $V_{1:p_j}$ and $[\phi_j]_{\phi_0}$ are orthonormal, it is easy to verify that $V_{1:p_j}^T V_{1:p_j} = [\phi_j]_{\phi_0}^T[\phi_j]_{\phi_0} = I$ (I is a $p_j \times p_j$ identity matrix).
So $V_{1:p_j} = V_{1:p_j}I = V_{1:p_j}V_{1:p_j}^T V_{1:p_j} = [\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T V_{1:p_j} = [\phi_j]_{\phi_0}([\phi_j]_{\phi_0}^T V_{1:p_j})$.

Next, we show $Q = [\phi_j]_{\phi_0}^T V_{1:p_j}$ is a rotation matrix. $Q^TQ = V_{1:p_j}^T[\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T V_{1:p_j} = V_{1:p_j}^T V_{1:p_j} V_{1:p_j}^T V_{1:p_j} = I$. $QQ^T = [\phi_j]_{\phi_0}^T V_{1:p_j} V_{1:p_j}^T [\phi_j]_{\phi_0} = [\phi_j]_{\phi_0}^T[\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T[\phi_j]_{\phi_0} = I$. Since $det(Q^TQ) = (det(Q))^2 = 1$, $det(Q) = 1$, Q is a rotation matrix. □

Theorem 2. Solution to generalized eigenvalue decomposition $X\mathcal{L}X^Tv = \lambda XX^Tv$ is given by $((F^T)^+x, \lambda)$, where x and λ are eigenvector and eigenvalue of $F^+X\mathcal{L}X^T(F^T)^+x = \lambda x$.
Proof:
We know $XX^T = FF^T$, where F is a $p \times r$ matrix of rank r.

Case 1: When XX^T is positive definite:
It is trivial to see that $r = p$. This implies that F is an $p \times p$ full rank matrix: $F^{-1} = F^+$.
$X\mathcal{L}X^Tv = \lambda XX^Tv$
$\Longrightarrow X\mathcal{L}X^Tv = \lambda FF^Tv \Longrightarrow X\mathcal{L}X^Tv = \lambda FF^T(F^T)^{-1}F^Tv$
$\Longrightarrow X\mathcal{L}X^Tv = \lambda F(F^Tv) \Longrightarrow X\mathcal{L}X^T(F^T)^{-1}(F^Tv) = \lambda F(F^Tv)$
$\Longrightarrow F^{-1}X\mathcal{L}X^T(F^T)^{-1}(F^Tv) = \lambda(F^Tv)$
$\Longrightarrow$ Solution to $X\mathcal{L}X^Tv = \lambda XX^Tv$ is given by $((F^T)^+x, \lambda)$, where x and λ are eigenvector and eigenvalue of $F^+X\mathcal{L}X^T(F^T)^+x = \lambda x$.

Case 2: When XX^T is positive semi-definite, but not positive definite:
In this case, $r < p$ and F is a $p \times r$ matrix of rank r.
Since X is a $p \times n$ matrix, F is a $p \times r$ matrix, there exits a matrix G such that $X = FG$. This implies $G = F^+X$.
$X\mathcal{L}X^Tv = \lambda XX^Tv$
$\Longrightarrow FG\mathcal{L}G^TF^Tv = \lambda FF^Tv \Longrightarrow FG\mathcal{L}G^T(F^Tv) = \lambda F(F^Tv)$
$\Longrightarrow (F^+F)G\mathcal{L}G^T(F^Tv) = \lambda(F^Tv) \Longrightarrow G\mathcal{L}G^T(F^Tv) = \lambda(F^Tv)$
$\Longrightarrow F^+X\mathcal{L}X^T(F^T)^+(F^Tv) = \lambda(F^Tv)$
$\Longrightarrow$ One solution to $X\mathcal{L}X^Tv = \lambda XX^Tv$ is $((F^T)^+x, \lambda)$, where x and λ are eigenvector and eigenvalue of $F^+X\mathcal{L}X^T(F^T)^+x = \lambda x$. Note that eigenvector solution to Case 2 is not unique. $\square$

Theorem 3. For any instance u, its embedding result with LPP (using top p_j eigenvectors) is the same as its embedding result with diffusion projections (at level j) up to a rotation.
Proof:
It is well known that the normalized graph Laplacian $\mathcal{L}$ is positive semi-definite (PSD), so $F^+X\mathcal{L}X^T(F^T)^+$ is also PSD, and all its eigenvalues are ≥ 0. This implies that eigenvectors corresponding to $F^+X\mathcal{L}X^T(F^T)^+$'s smallest non-zero eigenvalues are the same as eigenvectors corresponding to $(F^+X\mathcal{L}X^T(F^T)^+)^+$'s largest eigenvalues.

Let $T = (F^+X\mathcal{L}X^T(F^T)^+)^+$, $[\phi_j]_{\phi_0}$ (a $p \times p_j$ matrix) represent the diffusion scaling functions of T at level j. From Theorem 1, we have $V_{1:p_j} = [\phi_j]_{\phi_0}Q$, where $V_{1:p_j}$ is a $p \times p_j$ matrix, represents the p_j smallest eigenvectors of $F^+X\mathcal{L}X^T(F^T)^+$ and Q is a rotation.

Given an instance u ($p \times 1$ vector):
its embedding result with LPP is
$((F^T)^+V_{1:p_j})^Tu = V_{1:p_j}^TF^+u$;
its embedding result with diffusion projections is
$((F^T)^+[\phi_j]_{\phi_0})^Tu = [\phi_j]_{\phi_0}^TF^+u = QV_{1:p_j}^TF^+u$.
So these two embeddings are the same up to a rotation Q. $\square$

5 Experimental Results

In this section, we first use a toy example from computer vision to illustrate what multiscale scaling function $[\phi_j]_{\phi_0}$ look like. This helps us better understand how multiscale embedding is achieved. Our theorems show that for the undirected case, the low-dimensional embedding results from Laplacian eigenmaps, LPP (with top p_j eigenvectors) and diffusion projections (at level j) are the same up to a rotation. A "toroidal helix" example in Section 5.2 verifies this result. We claimed that our approach offers many advantages over the other approaches for the directed case. A "punctured sphere" example in Section 5.3 and a "citation analysis" example in Section 5.4 give credence to this argument. In all our experiments, diffusion projections computes c dimensional embedding by using top c basis functions of $[\phi_j]_{\phi_0}$, where level j has $\geq c$ basis functions but level $j+1$ does not. δ used in heat kernel is always 1.

5.1 A Toy Example: DiffusionFaces

In this experiment, we use a toy example to illustrate what multiscale scaling functions look like. The counterpart of our approach (diffusionfaces) in the eigenvector system is the eigenfaces [6]. Given m face images $I_1, \cdots I_m$, each of which is represented by an $n \times n$ matrix, the "eigenfaces" algorithm works as follows: (1) Convert each image I_i to a $n^2 \times 1$ vector Γ_i; (2) Compute the average face vector: $\overline{\Gamma} = \sum_{i=1}^{m} \Gamma_i / m$; (3) Normalize each image vector: $\Phi_i = \Gamma_i - \overline{\Gamma}$; (4) Compute the covariance matrix $C = [\Phi_1, \cdots, \Phi_m][\Phi_1, \cdots, \Phi_m]^T$; (5) Each eigenvector of C is an "eigenface". Using this approach, each image can be written as a linear combination of eigenfaces. In our approach, we start with the same covariance matrix C, but we use diffusion wavelets instead of applying eigenvectors. Each column of $[\phi_j]_{\phi_0}$ is used as a diffusionface.

We used the "Olivetti Faces" data in our test. This data set includes 400 face images, each of which is represented by 8-bit grayscale color and stored in a 64×64 matrix. Diffusion wavelets identifies a 4 level hierarchy of diffusionfaces, and dimensionality of each level is: 200, 53, 9, 2. We plot all 9 diffusionfaces at level 3 in Figure 4, the top 24 diffusionfaces at level 2 in Figure 5. We also plot the top 24 eigenfaces in Figure 6. It is clear that these two types of basis are quite different: eigenvectors are global, and almost all such bases model the whole face. Diffusion faces are defined at multiple scales, where the fine scale (e.g. Figure 5) characterizes the details about each image, while the coarser scales (e.g. Figure 4) skip some of the details and only keep the lower frequency information. Diffusion scaling functions are usually sparse (especially those at low levels), and most of them focus on just one particular feature on the face, like eyes, noses. So they are easier to interpret. Given an image written as a summation of diffusionfaces, we can estimate what the image looks like by checking the coefficients (contributions) of each type of eyes, noses, etc.

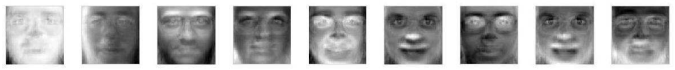


Figure 4: Diffusionfaces (level 3).

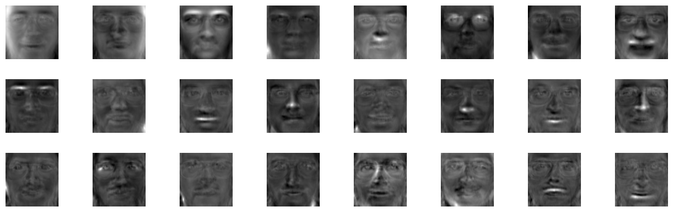


Figure 5: Diffusionfaces (level 2).

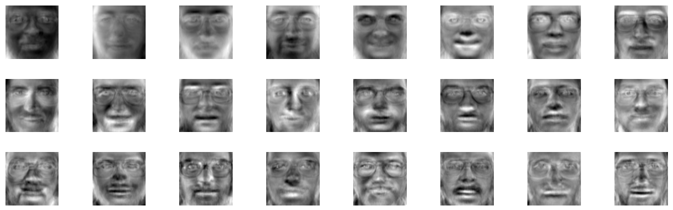


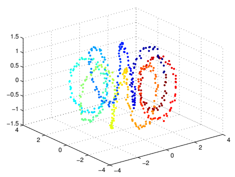
Figure 6: Eigenfaces.

5.2 Toroidal Helix example

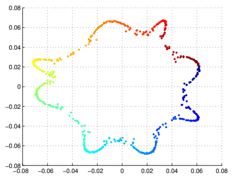
The "Toroidal Helix example" in Figure 7(A) uses 500 samples. We use the heat kernel to generate a weight matrix, and for each point, we compute the weights for its 20 nearest neighbors (in each row). This results in a non-symmetric matrix W. We symmetrize the weight matrix W: $\overline{W} = (W + W')/2$, and then apply Laplacian eigenmaps (Figure 7(B)) and LPP (Figure 7(E)) on $\overline{W}$. For the purpose of comparison, we also apply diffusion projections (DP) directly on $\overline{W}$ (Figure 7(C)) and W (Figure 7(D)). To verify Theorem 3, we also apply diffusion projections to learn LPP type mappings from $\overline{W}$ (Figure 7(F)). For all diffusion projection tests, we use the top level of the data hierarchy (in 2D spaces). The results show that for symmetrized graphs, the low dimensional embeddings achieved by Laplacian eigenmaps (or LPP) and diffusion projections are the same up to a rotation. It might be argued that embedding of directed graphs can be done by simply symmetrizing the matrix and doing eigenvalue decomposition. However, we will show in the next example this is not true for the general case.

5.3 Punctured Sphere Example

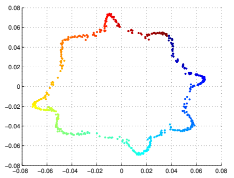
Consider the punctured sphere in Figure 7(G) based on 800 samples. We use the heat kernel to generate its weight matrix, and for each point, we compute the weights for 20 nearest neighbors (in each row). This results in a non-symmetric matrix W. We symmetrize W as usual, and then apply Laplacian eigenmaps and diffusion projections (DP) on $\overline{W}$ (Figure 7(H)(I)). Directed Laplacian (Figure 7(J)) and diffusion projections (Figure 7(K)) are also applied directly on W. To numerically compare the low dimensional embeddings, given any point x, we compute the probability that x's K nearest neighbors in the original sphere are still among its K nearest neighbors in the low dimensional embeddings (Figure 8). The figures show that diffusion projections on the original weight matrix W can successfully reconstruct the original structure, while both approaches based on symmetrized W fail. We know many graphs used in real world domains are directed. If we simply symmetrize such graphs, this example illustrates that the results may not be as good as using diffusion projections. The reason that symmetrization does not work is that for the points (red) on the rim of the sphere, their 20 neighbors are mostly red points. For the points (yellow) in the middle, some of their 20 neighbors are red, since the yellow points are sparse. Symmetrizing the relationship matrix will add links from the red to the yellow. This is equal to reinforcing the relationship between the red and yellow, which further forces the red to be close to the yellow in the low dimensional space. The above process weakens the relationship between the red points. So in the 3D embedding, we see some red points are far away from each other, while the red-yellow relationship is well preserved. Directed Laplacian also fails to generate good embeddings in this task. Figure 7(L) shows the spectrums of $T = \overline{W}$ and its higher powers. The high powers have a spectrum that decays much more rapidly than the low powers. We usually call a matrix



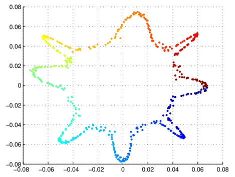
(A) Toroidal Helix



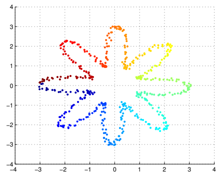
(B) Laplacian



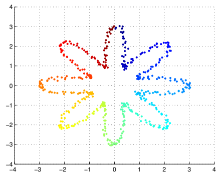
(C) DP undirected



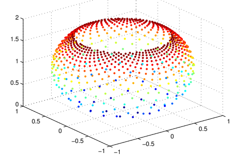
(D) DP directed



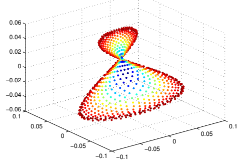
(E) LPP



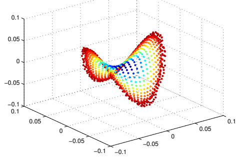
(F) DP-LPP



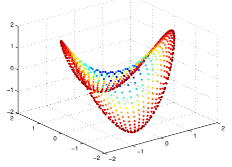
(G) Puncture Sphere



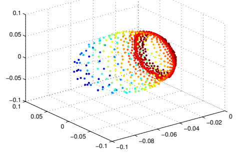
(H) Laplacian



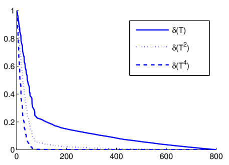
(I) DP undirected



(J) Directed Lap



(K) DP directed



(L) Spectrum

Figure 7: Toroidal Helix and Punctured Sphere Examples.

like this “diffusion like” matrix. In fact, most relationship matrices generated by k nearest neighbor method are “diffusion like”. Number of basis functions spanning $\overline{W}$'s column space at each of level is: 800, 741, 347, 63, 38, 23, 12, 6, 3, 1.

5.4 Citation Graph Mining

We now study a problem from citation graph mining. The citation data set in KDD Cup 2003 consists of scientific papers (about $29,000$ documents) from arXiv.org. These papers are from high-energy physics. They cover the period from 1992 through 2003. We sampled 3,369 documents from the data set and created a citation graph, i.e. a set of pairs of documents, showing that one paper references another. To evaluate the methods, we need to assign each document

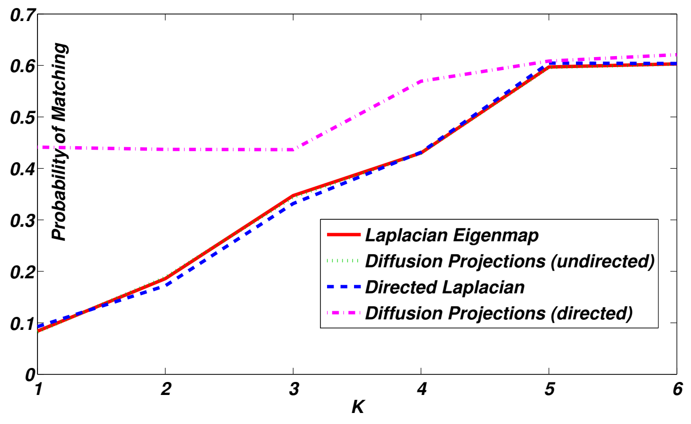


Figure 8: Comparison of punctured sphere embeddings.

a class type. To compute this, we first represent each paper using a TF-IDF vector based on the text of its abstract and the title, then we use the dot product to compute the similarity between any two papers. Hierarchy clustering is used to assign each document with a class. As a result, we get 8 classes.

We compare our approach to Laplacian eigenmaps using this data set. Our approach results in a 9 level hierarchy. Dimensionality of each level is: 3369, 1442, 586, 251, 125, 105, 94, 7, 1. Obviously, the citation graph is non-symmetric, and to apply Laplacian eigenmaps, we symmetrize the graph as before. A leave-one-out test is used to compare the low dimensional embeddings. We first map the data to a c dimensional space (we run 10 tests: $c = 10, 20, 30 \cdots 100$) using both diffusion projections and Laplacian eigenmaps. For each document in the new space, we check whether at least one document from the same class is among its K nearest neighbors (we use this as correctness). Diffusion projections performs better than Laplacian eigenmaps in all 10 tests. We also plot the average performance of these tests in Figure 9. The reason why Laplacian eigenmaps does a poor job is that the citation relationship is directed, and a paper that is cited by many other papers should be more important compared to a paper that cites many others but is not cited by others.

6 Conclusions

This paper presents a multiscale dimensionality reduction framework based on diffusion wavelets. In contrast to eigenvalue decomposition based approaches, which can only deal with symmetric relationships, our approach is also able to handle non-symmetric relationship matrices without ad-hoc symmetrization. The superior performance of our approach and some of its advantages are illustrated using several synthetic and real-world data sets.

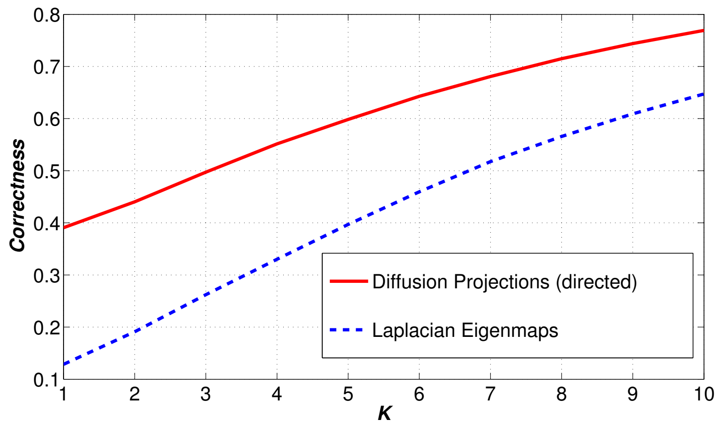


Figure 9: Comparison of citation graph embeddings.

References

[1] Belkin, M., Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15. 2003.

[2] Chung, F. Spectral graph theory. *Regional Conference Series in Mathematics*, 92. 1997.

[3] Chung, F. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics 9*. 2005.

[4] Coifman, R., Maggioni, R. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21. 2006

[5] He, X., Niyogi, P. Locality preserving projections. *NIPS 16*. 2003.

[6] Turk, M., Pentland, A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience* 3:71-86. 1991.