Multiscale Manifold Learning

Chang Wang
IBM T. J. Watson Research Lab
1101 Kitchawan Rd
Yorktown Heights, New York 10598
wangchan@us.ibm.com

Sridhar Mahadevan
Computer Science Department
University of Massachusetts
Amherst, Massachusetts 01003
mahadeva@cs.umass.edu

Abstract

Many high-dimensional data sets that lie on a low-dimensional manifold exhibit nontrivial regularities at multiple scales. Most work in manifold learning ignores this multiscale structure. In this paper, we propose approaches to explore the deep structure of manifolds. The proposed approaches are based on the diffusion wavelets framework, data driven, and able to directly process directional neighborhood relationships without ad-hoc symmetrization. The proposed multiscale algorithms are evaluated using both synthetic and real-world data sets, and shown to outperform previous manifold learning methods.

Introduction

In many application domains of interest, from information retrieval and natural language processing to perception and robotics, data appears high dimensional, but often lies near or on low-dimensional structures, such as a manifold or a graph. By explicitly modeling and recovering the underlying structure, manifold learning methods (Belkin and Niyogi 2003; Roweis and Saul 2000; He and Niyogi 2003) have been shown to be significantly more effective than previous dimensionality reduction methods. Many existing manifold learning approaches are largely based on extending classical Fourier analysis to graphs and manifolds. In particular, spectral graph theory (Chung 1997) combined with classical differential geometry and global analysis on manifolds forms the theoretical basis for "Laplacian" techniques for function approximation and learning on graphs and manifolds, using the eigenfunctions of a Laplace operator naturally defined on the data manifold to reveal hidden structure. While Fourier analysis is a powerful tool for *global* analysis of functions, it is known to be poor at recovering multiscale regularities across data and for modeling local or transient properties (Mallat 1998). Consequently, one limitation of these techniques is that they only yield a "flat" embedding but not a multiscale embedding. However, when humans try to solve a particular problem (such as natural language processing), they often exploit their intuition about how to decompose the problem into sub-problems and construct multiple levels of representation. As a consequence, there has been rapidly growing interest in the problem of "deep learning", wherein learning methods are designed that construct multiple layers of latent representations from data (Hinton and Salakhutdinov 2006; Hinton, Osindero, and Teh 2006; Lee et al. 2007; Bengio 2009).

Another problem with such Fourier analysis based methods is that they 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, in information retrieval/data mining applications based on network topology (Shin, Hill, and Raetsch 2006), 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. It is more desirable to find an approach that handles directed graphs without the need for symmetrization.

To address the need for multiscale analysis and directional neighborhood relationships, we explore multiscale extensions of Fourier analysis based approaches using *wavelet* analysis (Mallat 1998). Classical wavelets in Euclidean spaces allow a very efficient multiscale analysis much like a highly flexible tunable microscope probing the properties of a function at different locations and scales. Diffusion wavelets (DWT) (Coifman and Maggioni 2006) extends the strengths of classical wavelets to data that lie on graphs and manifolds. The term *diffusion wavelets* is used because it is associated with a diffusion process that defines the different scales, allows a multiscale analysis of functions on manifolds and graphs. We focus on multiscale extensions of Laplacian eigenmaps (Belkin and Niyogi 2003) and LPP (He and Niyogi 2003). Laplacian eigenmaps constructs embeddings of data using the low-order eigenvectors of the graph Laplacian as a new coordinate basis (Chung 1997), which extends Fourier analysis to graphs and manifolds. Locality Preserving Projections (LPP) is a linear approximation of Laplacian eigenmaps.

Our paper makes the following specific contributions: (1) We investigate the relationships between DWT and (multiscale) Laplacian eigenmaps and LPP. To extend LPP to a multiscale variant requires solving a generalized eigenvalue

problem using diffusion wavelets. This extension requires processing two matrices, and was not addressed in previous work on diffusion wavelets. (2) We also show how to apply the method to directed (non-symmetric) graphs. Previous applications of diffusion wavelets did not focus on non-symmetric weight matrices.

Similar to Laplacian eigenmaps and LPP, our approach 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 directional relationship induced by non-symmetric actions in a Markov decision process. Such pairwise similarities generate a transition probability matrix for a random walk $P = D^{-1}W$, where W is the weight matrix, and D is a diagonal "valency" matrix of the row-sums of W. In contrast to almost all previous graph-based eigenvector methods, we do not require W to be symmetric. In Laplacian eigenmaps and LPP, dimensionality reduction is achieved using eigenvectors of the graph Laplacian. In the new approach, we use diffusion scaling functions, which are defined at multiple scales. In the special case of symmetric matrices, these span the same space as selected spectral bands of eigenvectors.

The remainder of this paper is organized as follows. The next section discusses the diffusion wavelets model. Then, we explain the main multiscale manifold learning algorithms and the rationale underlying our approaches. We finish with a presentation of the experimental results and conclusions.

Diffusion Wavelets Model

The procedure for performing multiscale decompositions using diffusion wavelets and the relevant notation are explained in Figure 1. The main procedure can be explained as follows: an input matrix T is orthogonalized using an approximate QR decomposition in the first step. T's QR decomposition is written as $T = QR$, where Q is an orthogonal matrix and R is an upper triangular matrix. The orthogonal columns of Q are the scaling functions. They span the column space of matrix T. The upper triangular matrix R is the representation of T on the basis Q. In the second step, we compute T^2. Note this is not done simply by multiplying T by itself. Rather, T^2 is represented on the new basis Q: $T^2 = (RQ)^2$. This result is based on matrix invariant subspace theory (Stewart and Sun 1990). Since Q may have fewer columns than T, T^2 may be a smaller square matrix. The above process is repeated at the next level, generating compressed dyadic powers T^{2^j}, until the maximum level is reached or its effective size is a 1×1 matrix. Small powers of T correspond to short-term behavior in the diffusion process and large powers correspond to long-term behavior. Scaling functions are naturally multiscale basis functions because they account for increasing powers of T (in particular, the dyadic powers 2^j). At scale j, the representation of T^{2^j} is compressed based on the amount of remaining information and the precision we want to keep. Figure 2 illustrates this procedure.

$\{\phi_j, T_j\} = DWT(T, \phi_0, QR, J, \varepsilon)$
//**INPUT**:
$//T$: Diffusion operator. ϕ_0: Initial (unit vector) basis matrix. QR: A modified QR decomposition.
$//J$: Max step number. This is optional, since the algorithm automatically terminates.
$//\varepsilon$: Desired precision, which can be set to a small number or simply machine precision.
//**OUTPUT** : ϕ_j: Diffusion scaling functions at scale j. $T_j = [T^{2^j}]_{\phi_j}^{\phi_j}$.
$For\ j = 0\ to\ J-1\{$
$\quad ([\phi_{j+1}]_{\phi_j}, [T^{2^j}]_{\phi_j}^{\phi_{j+1}}) \leftarrow QR([T^{2^j}]_{\phi_j}^{\phi_j}, \varepsilon);$
$\quad [T^{2^{j+1}}]_{\phi_{j+1}}^{\phi_{j+1}} = ([T^{2^j}]_{\phi_j}^{\phi_{j+1}}[\phi_{j+1}]_{\phi_j})^2;$
$\}$

Figure 1: Diffusion Wavelets construct multiscale representations at different scales. 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. Typically the initial basis for the algorithm ϕ_0 is assumed to be the delta functions (represented by an identity matrix), but this is not strictly necessary.

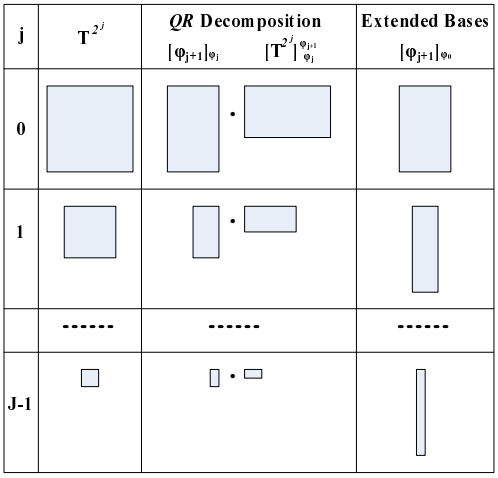

Figure 2: Multiscale diffusion analysis.

We use the "Olivetti Faces" data to illustrate the difference between eigenvector basis and diffusion wavelets basis (scaling functions). The dataset contains 200 face images represented over pixels. The well-known eigenface approach (Turk and Pentland 1991) first computes the pixel-pixel covariance matrix, and then computes the corresponding eigenvectors. Each eigenvector 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, but we use diffusion wavelets instead of eigenvectors. Each column of $[\phi_j]_{\phi_0}$ is used as a "diffusion face". Diffusion wavelets model identifies a 4 level hierar-

Figure 3: All 9 Diffusion Wavelets Basis Functions at Level 3.

Figure 4: Selected Diffusion Wavelets Basis Functions at Level 1.

Figure 5: Eigenfaces.

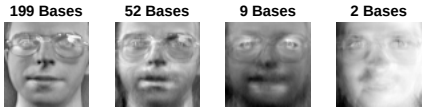

Figure 6: Image Reconstruction at Different Scales

chy of diffusionfaces, and dimensionality of each level is: 199, 52, 9, 2. We plot all 9 diffusionfaces at level 3 in Figure 3, and the top 24 diffusionfaces at level 1 in Figure 4. We also plot the top 24 eigenfaces in Figure 5. 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 finer scale (e.g. Figure 4) characterizes the details about each image, while the coarser scales (e.g. Figure 3) skip some of the details and only keep the lower frequency information. Scaling functions (especially those at low levels) are usually sparse, and most of them focus on just one particular feature on the face, like eyes and noses. Given an image written as a summation of diffusionfaces, we can estimate what the image looks like based on the coefficients (contributions) of each type of eyes, noses, etc. Figure 6 shows the face reconstruction results using diffusion faces at different scales.

Multiscale Manifold Learning

In this section, we discuss how to extend Laplacian eigenmaps and LPP to multiple scales using diffusion wavelets.

Notation: $X = [x_1, \cdots, x_n]$ be an $p \times n$ matrix representing n instances defined in a p dimensional space. W is an $n \times n$ weight matrix, where $W_{i,j}$ represents the similarity of x_i and x_j ($W_{i,j}$ can be defined by $e^{-\|x_i - x_j\|^2}$). D is a diagonal valency 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 $\mathcal{L}$ is the normalized Laplacian matrix and I is an identity matrix. $XX^T = FF^T$, where F is a $p \times r$ matrix of rank r. One way to compute F from X is singular value decomposition. $(\cdot)^+$ represents the Moore-Penrose pseudo inverse.

(1) Laplacian eigenmaps minimizes the cost function $\sum_{i,j}(y_i - y_j)^2 \mathcal{W}_{i,j}$, which encourages the neighbors in the original space to be neighbors in the new space. The c dimensional embedding is provided by eigenvectors of $\mathcal{L}x = \lambda x$ corresponding to the c smallest non-zero eigenvalues. The cost function for ***multiscale Laplacian eigenmaps*** is defined as follows: 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 $k = 1, \cdots, J$ represents each level of the underlying manifold hierarchy.

(2) LPP is a linear approximation of Laplacian eigenmaps. LPP minimizes the cost function $\sum_{i,j}(f^T x_i - f^T x_j)^2 \mathcal{W}_{i,j}$, where the mapping function f constructs a c dimensional embedding, and is defined by the eigenvectors of $X\mathcal{L}X^T x = \lambda XX^T x$ corresponding to the c smallest non-zero eigenvalues. Similar to multiscale Laplacian eigenmaps, ***multiscale LPP*** learns linear mapping functions defined at multiple scales to achieve multilevel decompositions.

1. **Construct diffusion matrix T characterizing the given data set:**
   - $T = I - \mathcal{L}$ is an $n \times n$ diffusion matrix.
2. **Construct multiscale basis functions using diffusion wavelets:**
   - $\{\phi_j, T_j\} = DWT(T, I, QR, J, \varepsilon)$.
   - The resulting $[\phi_j]_{\phi_0}$ is an $n \times p_j$ matrix (Equation (1)).
3. **Compute lower dimensional embedding (at level j):**
   - The embedding $x_i \rightarrow y_i =$ row i of $[\phi_j]_{\phi_0}$.

1. **Construct relationship matrix T characterizing the given data set:**
   - $T = (F^+ X\mathcal{L}X^T (F^T)^+)^+$ is an $r \times r$ matrix..
2. **Apply diffusion wavelets to explore the intrinsic structure of the data:**
   - $\{\phi_j, T_j\} = DWT(T, I, QR, J, \varepsilon)$.
   - The resulting $[\phi_j]_{\phi_0}$ is an $r \times p_j$ matrix (Equation (1)).
3. **Compute lower dimensional embedding (at level j):**
   - The embedding $x_i \rightarrow y_i = ((F^T)^+ [\phi_j]_{\phi_0})^T x_i$.

Figure 7: Top: Multiscale Laplacian Eigenmaps; Bottom: Multiscale LPP.

The Multiscale Algorithms

Multiscale Laplacian eigenmaps and multiscale LPP algorithms are shown in Figure 7, where $[\phi_j]_{\phi_0}$ is used to compute a lower dimensional embedding. As shown in Figure 1, the scaling functions $[\phi_{j+1}]_{\phi_j}$ are the orthonormal bases to span the column space of T at different levels. They define a set of new coordinate systems revealing the information in the original system at different scales. The scaling functions also provide a mapping between the data at longer spatial/temporal scales and smaller scales. Using the scaling functions, the basis functions at level j can be represented in terms of the basis functions at the next lower level. In this manner, the extended basis functions can be expressed in terms of the basis functions at the finest scale using:

$$[\phi_j]_{\phi_0} = [\phi_j]_{\phi_{j-1}}[\phi_{j-1}]_{\phi_0} = [\phi_j]_{\phi_{j-1}} \cdots [\phi_1]_{\phi_0}[\phi_0]_{\phi_0}, \quad (1)$$

where each element on the right hand side of the equation is created by the procedure shown in Figure 1. In our approach, $[\phi_j]_{\phi_0}$ is used to compute lower dimensional embeddings at multiple scales. Given $[\phi_j]_{\phi_0}$, any vector/function on the compressed large scale space can be extended naturally to the finest scale space or vice versa. The connection between vector v at the finest scale space and its compressed representation at scale j is computed using the equation $[v]_{\phi_0} = ([\phi_j]_{\phi_0})[v]_{\phi_j}$. The elements in $[\phi_j]_{\phi_0}$ are usually much coarser and smoother than the initial elements in $[\phi_0]_{\phi_0}$, which is why they can be represented in a compressed form.

Theoretical Analysis

It is well-known that regular Laplacian eigenmaps and LPP both return the optimal dimensionality reduction results with respect to the cost functions described at the beginning of this section (Belkin and Niyogi 2003; He and Niyogi 2003). If the input matrix is symmetric, there is an interesting connection between our algorithms and regular approaches. Theorem 1 and 3 below prove that the dimensionality reduction results produced by the proposed approaches at level k and the results from Laplacian eigenmaps and LPP (with top p_k eigenvectors) are the same up to a rotation and a precision. So the proposed approaches are also optimal with respect to the same cost functions up to a precision. Theorems 2 proves some intermediate results, which are subsequently used in Theorem 3. One significant advantage of our approach is that it also directly generalizes to non-symmetric input matrices.

Theorem 1: Laplacian eigenmaps (with eigenvectors corresponding to p_j smallest non-zero eigenvalues) and Multiscale Laplacian eigenmaps (at level j) return the same p_j dimensional embedding up to a rotation Q and a precision.

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 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 up to a precision (Coifman and Maggioni 2006), i.e.

$$V_{1:p_j}V_{1:p_j}^T = [\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T.$$

Since the columns of both $V_{1:p_j}$ and $[\phi_j]_{\phi_0}$ are orthonormal, it is easy to verify that

$$V_{1:p_j}^T V_{1:p_j} = [\phi_j]_{\phi_0}^T[\phi_j]_{\phi_0} = I,$$

where I is a $p_j \times p_j$ identity matrix. So

$$V_{1:p_j} = V_{1:p_j}V_{1:p_j}^T V_{1:p_j} = [\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T V_{1:p_j} = [\phi_j]_{\phi_0}([\phi_j]_{\phi_0}^T V_{1:p_j}).$$

Next, we show $Q = [\phi_j]_{\phi_0}^T V_{1:p_j}$ is a rotation matrix.

$$Q^TQ = V_{1:p_j}^T[\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T V_{1:p_j} = V_{1:p_j}^T V_{1:p_j}V_{1:p_j}^T V_{1:p_j} = I.$$

$$QQ^T = [\phi_j]_{\phi_0}^T V_{1:p_j}V_{1:p_j}^T[\phi_j]_{\phi_0} = [\phi_j]_{\phi_0}^T[\phi_j]_{\phi_0}[\phi_j]_{\phi_0}^T[\phi_j]_{\phi_0} = I.$$

$$det(Q^TQ) = (det(Q))^2 = 1 \Longrightarrow det(Q) = 1$$

So Q is a rotation matrix. □

The embeddings constructed by LPP reduces to solving the generalized eigenvalue decomposition $X\mathcal{L}X^Tx = \lambda XX^Tx$, where we have two input matrices $X\mathcal{L}X^T$ and XX^T to process. However, using the DWT procedure requires converting the generalized eigenvalue decomposition to a regular eigenvalue decomposition problem (with one input 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 follows immediately that $r = p$. This implies that F is an $p \times p$ full rank matrix: $F^{-1} = F^+$.

$$X\mathcal{L}X^Tv = \lambda XX^Tv \Longrightarrow X\mathcal{L}X^Tv = \lambda FF^Tv$$
$$\Longrightarrow X\mathcal{L}X^Tv = \lambda FF^T(F^T)^{-1}F^Tv$$
$$\Longrightarrow X\mathcal{L}X^Tv = \lambda F(F^Tv) \Longrightarrow X\mathcal{L}X^T(F^T)^{-1}(F^Tv) = \lambda F(F^Tv)$$
$$\Longrightarrow F^{-1}X\mathcal{L}X^T(F^T)^{-1}(F^Tv) = \lambda(F^Tv)$$

So 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 $G = F^+X$.

$$X\mathcal{L}X^Tv = \lambda XX^Tv \Longrightarrow FG\mathcal{L}G^TF^Tv = \lambda FF^Tv$$
$$\Longrightarrow FG\mathcal{L}G^T(F^Tv) = \lambda F(F^Tv)$$
$$\Longrightarrow (F^+F)G\mathcal{L}G^T(F^Tv) = \lambda(F^Tv) \Longrightarrow G\mathcal{L}G^T(F^Tv) = \lambda(F^Tv)$$
$$\Longrightarrow F^+X\mathcal{L}X^T(F^T)^+(F^Tv) = \lambda(F^Tv)$$

So 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 the eigenvector solution to Case 2 is not unique. □

Theorem 3: For any instance u, its embedding under LPP (using the top p_j eigenvectors) is the same as its embedding under multiscale LPP (at level j) up to a rotation and a precision.

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, it follows that $V_{1:p_j} = [\phi_j]_{\phi_0}Q$ 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 multiscale LPP is

$$((F^T)^+[\phi_j]_{\phi_0})^Tu = [\phi_j]_{\phi_0}^TF^+u = QV_{1:p_j}^TF^+u.$$

So, the two embeddings are the same up to a rotation Q and a precision. □

Experimental Results

To test the effectiveness of our multiscale manifold learning approaches, we compared "flat" and "deep" multiscale approaches using dimensionality reduction tasks on both synthetic and real-wold data sets. It is useful to emphasize that the intrinsic structure of the data set does not depend on the parameters. The structure only depends on the given data. The input parameters decide the way to explore the structure. The time complexity of the proposed approaches are similar to the corresponding eigenvector based approaches.

Punctured Sphere Example

Consider the punctured sphere in Figure 8(A) 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. To apply Laplacian eigenmaps and LPP, we symmetrize W: $\overline{W} = (W + W^T)/2$. Figure 8(B) shows the spectrums of $\overline{W}$ and its higher powers. The high powers have a spectrum that decays much more rapidly than the low powers. This spectral decay property is characteristic of "diffusion-like" matrices, particularly those generated by the k nearest neighbor similarity metric. The embedding results are in Figure 8(C)-(I). The results verify Theorem 1 and 3, showing multiscale approaches (using diffusion scaling functions at level j) and eigenmap approaches (using top p_j eigenvectors) result in the same embeddings up to a rotation and a precision. Furthermore, multiscale Laplacian eigenmaps can successfully identify the intrinsic structures of the data set. Dimensionality of the finest scales of multiscale Laplacian eigenmaps is 3, which is the intrinsic dimensionality of the given data. Also, among all four nonlinear dimensionality reduction approaches (Direct Laplacian (Chung 2005), Laplacian eigenmaps, Multiscale Laplacian eigenmaps with W and $\overline{W}$), only Multiscale Laplacian eigenmaps with the original weight matrix W reconstructs the original structure, while both approaches based on symmetrized W fail. 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. Finally, all three linear dimensionality reduction approaches (LPP, multiscale LPP with W and $\overline{W}$) can reconstruct the original structure. A possible reason for this is that the strong linear mapping constraint prevents overfitting from happening for this task.

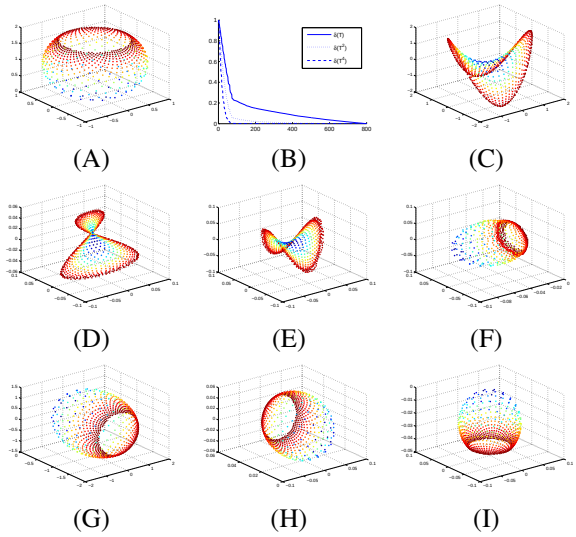

Figure 8: Punctured Sphere Example: (A) Puncture Sphere; (B) Spectrum of $\overline{W}$; (C) Directed Laplacian with W; (D) Laplacian eigenmaps with $\overline{W}$; (E) Multiscale Laplacian eigenmaps with $\overline{W}$ (finest scale); (F) Multiscale Laplacian eigenmaps with W (finest scale); (G) LPP with $\overline{W}$; (H) Multiscale LPP with $\overline{W}$; (I) Multiscale LPP with W.

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 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

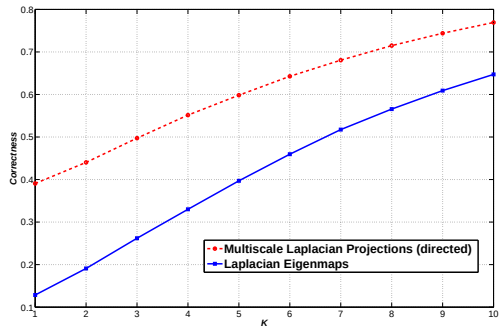

Figure 9: Comparison of citation graph embeddings.

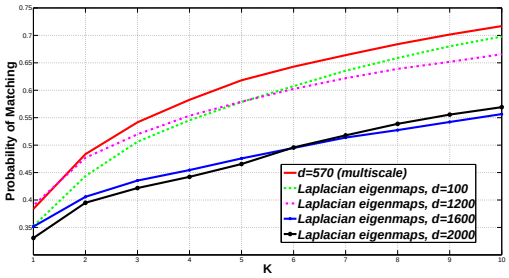

Figure 10: Comparison of NSF abstracts embeddings (using 'division' as label).

compute the similarity between any two papers. Hierarchical clustering is used to assign each document with a class. As a result, we get 7 classes. We apply both Multiscale Laplacian eigenmaps and regular Laplacian eigenmaps to the citation graph (without using document contents). Since the input is a graph, LPP and multiscale LPP can not be used. Multiscale approach results in a 8 level hierarchy. Dimensionality of each level is: 3369, 1442, 586, 251, 125, 105, 94, 7. From the result, we can see that multiscale approach successfully identifies the real intrinsic dimensionality at the highest level: 7 classes. 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 d dimensional space (we run 10 tests: $d = 10, 20, 30 \cdots 100$) using both multiscale approach (using basis functions at level 6) and regular 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. The multiscale approach using a non-symmetric graph performs much better than regular Laplacian eigenmaps with a symmetric graph in all 10 tests. We plot the average performance of these tests in Figure 9. Laplacian eigenmaps is less effective because the citation relationship is directed, and a paper that is cited by many other papers should be treated as completely different from a paper that cites many others but is not cited by others.

NSF Research Awards Abstracts Data

We also ran a test on a selected set of the NSF research awards abstracts (Frank and Asuncion 2010), which includes 5,000 abstracts describing NSF awards for basic research. The data set is represented by bag-of-words and has already been cleaned. Each abstract has a corresponding label: "division" (37 different values). Using Multiscale Laplacian eigenmaps, a 9 level manifold hierarchy was automatically constructed. Dimensionality discovered at each level was: 5000, 3069, 3052, 2524, **570**, 54, 20, 13, 9. We applied the same quantitative comparison approach as that used in the previous section to compare Multiscale Laplacian eigenmaps (level 5) and regular Laplacian eigenmaps (with varying numbers of eigenvectors: 100, 1200, 1600, 2000). The results are summarized in Figure 10. The 570 dimensional embedding returned the best results.

From the figures, we can see that choosing an appropriate scale for embedding can help improve learning performance. Using too many or too few bases may result in a redundant feature space or loss of valuable information. Finding an appropriate value for dimensionality is quite difficult. In previous approaches, the users need to specify this value. Generally speaking, even though a given problem may have tens of thousands of instances, the number of levels identified by the new approach is a much smaller number (often < 10). Also, some levels are defined by either too many or too few features. This eliminates from consideration additional levels, usually leaving a handful of levels as possible candidates. In this example, we chose the space defined by 570 features, since the levels below and above this have too few or too many features, respectively. Manifold hierarchy is task independent. For different tasks, users can select the most appropriate level by testing his/her data at different levels. For simplicity, the paper focuses on selecting scaling functions at a single level, but the methods can be extended to use multiple levels together.

Conclusions

This paper presents manifold learning techniques that yield a multiscale decomposition of high-dimensional data. The proposed approaches are based on the diffusion wavelets framework, and are data driven. In contrast to "flat" eigenvector based approaches, which can only deal with symmetric relationships, our approach is able to analyze non-symmetric relationship matrices without ad-hoc symmetrization. The superior performance of the multiscale techniques and some of their advantages are illustrated using both synthetic and real-world data sets.

Acknowledgments

This research is supported in part by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-10-1-0383, and the National Science Foundation under Grant Nos. NSF CCF-1025120, IIS-0534999, IIS-0803288, and IIS-1216467. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFOSR or the NSF.

References

Belkin, M., and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15:1373–1396.

Bengio, Y. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2(1):1–127.

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

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

Coifman, R., and Maggioni, M. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis* 21:53–94.

Frank, A., and Asuncion, A. 2010. UCI machine learning repository. [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

He, X., and Niyogi, P. 2003. Locality preserving projections. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313:504–507.

Hinton, G. E.; Osindero, S.; and Teh, Y. W. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18:1527–1554.

Lee, H.; Battle, A.; Raina, R.; and Ng, A. 2007. Efficient sparse coding algorithms. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 801–8084.

Mallat, S. 1998. *A wavelet tour in signal processing*. Academic Press.

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

Shin, H.; Hill, N.; and Raetsch, G. 2006. Graph-based semi-supervised learning with sharper edges. In *Proceedings of the European Conference on Machine Learning*, 401–412.

Stewart, G. W., and Sun, J. 1990. *Matrix perturbation theory*. Academic Press.

Turk, M., and Pentland, A. 1991. Eigenfaces for recognition. *Journal of Cognitive Neuroscience* 3:71–86.