A General Framework for Manifold Alignment

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

Abstract

Manifold alignment has been found to be useful in many fields of machine learning and data mining. In this paper we summarize our work in this area and introduce a general framework for manifold alignment. This framework generates a family of approaches to align manifolds by simultaneously matching the corresponding instances and preserving the local geometry of each given manifold. Some approaches like semi-supervised alignment and manifold projections can be obtained as special cases. Our framework can also solve multiple manifold alignment problems and be adapted to handle the situation when no correspondence information is available. The approaches are described and evaluated both theoretically and experimentally, providing results showing useful knowledge transfer from one domain to another. Novel applications of our methods including identification of topics shared by multiple document collections, and biological structure alignment are discussed in the paper.

Introduction

In many areas of machine learning and data mining, one is often confronted with situations where the data is in a high dimensional space. Directly dealing with such high dimensional data is usually intractable, but in many cases, the manifold structure underlying the data may have a low intrinsic dimensionality. Manifold alignment builds connections between two or more disparate data sets by aligning their underlying manifolds and provides knowledge transfer across the data sets. More formally, given data sets $X_k = \{x_k^1, \cdots, x_k^{m_k}\}$ where $k \in \{1, \cdots, c\}$ (assuming X_k is from manifold $\mathcal{X}_k$)) together with a small fraction of samples labelled with known correspondences, we want to find a correspondence between them. Directly working with the original data instances can be quite difficult, since they are in high dimensional spaces and might be defined by totally different features. A solution is to map X_k for different ks to a new space, where instances x_a^i and x_b^j can be directly compared for any two data sets X_a and X_b. Manifold alignment can be used to transfer knowledge across domains. Real-world applications include automatic machine translation (Diaz & Metzler 2007), cross-lingual information retrieval, representation and control transfer in Markov decision processes, bioinformatics (Wang & Mahadevan 2008), image interpretation and social network analysis.

Existing manifold alignment approaches can be categorized into two types. The first type (illustrated in Figure 1(A)) includes diffusion map-based alignment (Lafon et al. 2006) and Procrustes alignment (Wang & Mahadevan 2008). These approaches first map the data sets to low dimensional spaces reflecting their intrinsic geometries using a standard (linear like LPP (He & Niyogi 2003) or nonlinear like Laplacian eigenmaps (Belkin & Niyogi 2003)) dimensionality reduction approach. Then some components (like rotational and scaling components) are removed from one set so that the alignment of two sets can be achieved. In this type of alignment, the computation of lower dimensional embedding is done in a unsupervised way (without considering the purpose of alignment), so the resulting embeddings of the two data sets might be quite different. The second type (illustrated in Figure 1(B)) includes semi-supervised alignment (Ham et al. 2005), manifold projections (Wang & Mahadevan 2009) and semi-definite alignment (Xiong et al. 2007). Semi-supervised alignment first creates a joint manifold representing the union of the given manifolds. Then it maps the joint manifold to a lower dimensional *latent* space preserving local geometry of each manifold, and matching instances in correspondence. Semi-supervised alignment is based on eigenvalue decomposition. Semi-definite alignment solves a similar problem using a semi-definite programming framework. Manifold projections is a linear approximation of semi-supervised alignment. It directly builds connections between features rather than instances and can naturally handle new test instances.

One situation that arises in many real world applications is that the given manifolds (defined by totally different features) need to be aligned with no available correspondence information. Solving this problem is rather difficult, if not impossible, since there are two unknown variables in this problem: the correspondence and the transformation. One such example is control transfer between different Markov decision processes (MDPs), where we want to align state spaces of different tasks. Here, states are usually defined by different features for different tasks and it is hard to find correspondences between them. Another example is knowledge transfer between document collections in different languages. In this task, correspondence information is usually

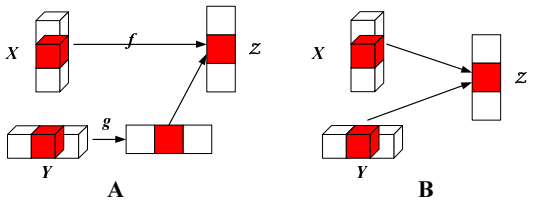

Figure 1: Two types of manifold alignment (this figure only shows two manifolds, but the same idea also applies to multiple manifold alignment). X is a data set sampled from manifold $\mathcal{X}_a$, Y is a data set sampled from manifold $\mathcal{X}_b$. $\mathcal{Z}$ is the new space. The red regions represent the subsets that are in correspondence. f and g are functions to compute lower dimensional embedding of X and Y. (A) First type: including diffusion map-based alignment and procrustes alignment; (B) Second type: including semi-supervised alignment, manifold projections, semi-definite alignment.

achieved via a manual translation, which is very expensive.

In this paper we summarize our existing work on manifold alignment, and introduce a new general framework to this problem. This framework generates a family of approaches to align manifolds by simultaneously matching the corresponding instances and preserving the local geometry of each given manifold. Some existing approaches like semi-supervised alignment and manifold projections can be obtained as special cases. Our approach can solve multiple manifold alignment problems (number of manifolds to be aligned is ≥ 3) and be adapted to handle the situation when no correspondence information is available. The framework is described and evaluated both theoretically and experimentally, providing results showing useful knowledge transfer from one domain to another. Novel applications of our algorithms including identification of topics shared by multiple document collections, and biological structure alignment are discussed in this paper.

The rest of this paper is organized as follows. In the first section, we describe the problem and the main framework. Subsequently, in the next section, we provide a theoretical analysis of our approach. Then we show the connections of our framework to some existing approaches and applications in the third section titled "Algorithms". We summarize our experimental results in the following section. Finally, we provide some concluding remarks in the final section.

The Algorithmic Framework

The Problem

The problem we want to solve is as follows: given multiple data sets $X_k (k \in \{1, \cdots, c\})$ where X_k comes from manifold $\mathcal{X}_k$, along with partial correspondence information in the form of paired instances $x_a^i \in X_a \longleftrightarrow x_b^j \in X_b$, map X_k to a new space preserving local geometry of each set and matching instances in correspondence. Notation used in this paper is summarized in Figure 2

High Level Explanation

Given the input manifolds $\mathcal{X}_1, \cdots, \mathcal{X}_c$, we first create a joint manifold represented by matrix L modeling the union of all manifolds. L has information coming from $L_k (k \in \{1, \cdots, c\}$. Such information models the local geometry of each given manifold. $\Omega_{a,b}^{a,b}$ and $\Omega_{a,b}^{a,a} (a, b \in \{1, \cdots, c\})$ in L play a key role in joining the these manifolds. They force the instances in correspondence (from different manifolds) to be neighbors in the joint manifold. The joint manifold is then mapped to a lower dimensional space preserving its local geometry. The idea is illustrated in Figure 3.

x_k^i is defined in a p_k dimensional space (manifold $\mathcal{X}_k$);
$X_k = \{x_k^1, \cdots, x_k^{m_k}\}$, X_k is a $p_k \times m_k$ matrix.
W_k is an $m_k \times m_k$ matrix, where $W_k^{i,j}$ is the similarity of x_k^i and x_k^j (could be defined by heat kernel).
D_k is an $m_k \times m_k$ diagonal matrix: $D_k^{i,i} = \sum_j W_k^{i,j}$.
$L_k = D_k - W_k$.

When $a = b$: $W_{a,a}$ and $\Omega_{a,a}^{a,a}$ are both $m_a \times m_a$ zero matrices.
When $a \neq b$:
$W_{a,b}$ is an $m_a \times m_b$ matrix, where $W_{a,b}^{i,j} = 1$, when x_a^i and x_b^j are in correspondence; 0, otherwise.
$\Omega_{a,b}^{a,a}$ is an $m_a \times m_a$ diagonal matrix, where $\Omega_{a,b}^{a,a}(i,i) = \sum_j W_{a,b}^{i,j}$.
$\Omega_{a,b}^{a,b}$ is an $m_a \times m_b$ matrix, where $\Omega_{a,b}^{a,b}(i,j) = W_{a,b}^{i,j}$.

c: number of manifolds that we want to align.

$$Z = \begin{pmatrix} X_1 & \cdots & 0 \\ & \cdots & \\ 0 & \cdots & X_c \end{pmatrix}, D = \begin{pmatrix} D_1 & \cdots & 0 \\ & \cdots & \\ 0 & \cdots & D_c \end{pmatrix}.$$

$$L = \begin{pmatrix} L_1 + \mu \sum_{k=1}^c \Omega_{1,k}^{1,1} & \cdots & -\mu \Omega_{1,c}^{1,c} \\ & \cdots & \\ -\mu \Omega_{c,1}^{c,1} & \cdots & L_c + \mu \sum_{k=1}^c \Omega_{c,k}^{c,c} \end{pmatrix}.$$

$\mathcal{F}_k$ is a mapping to map x_k^i to the common space (d dimensional): $\mathcal{F}_k^T x_i$ ($\mathcal{F}_k$ is a $p_k \times d$ matrix).
$(\mathcal{F}_1^T, \mathcal{F}_2^T, \cdots, \mathcal{F}_c^T)^T = \gamma_{1:d}$, where $\gamma_{1:d}$ represent eigenvectors of $ZLZ^T\gamma = \lambda ZDZ^T\gamma$ corresponding to the d smallest non-zero eigenvalues.

Figure 2: Notation used in this paper.

The Overall Algorithm

The main algorithmic framework is summarized in Figure 4. Depending on our targets – feature-level alignment or instance-level alignment – the framework has two slightly different versions.

Theoretical Results

Cost Functions

Manifold alignment can be done at two levels: ***instance-level*** and ***feature-level***. In text mining, examples of instances can be documents in English, Arabic, etc; examples of features can be English words/topics, Arabic words/topics, etc. Instance-level alignment builds connections between instances. It can compute nonlinear embeddings for alignment, but such an alignment result is defined only on known instances, and hard to be generalized to new instances. Feature-level alignment builds connections between features, and fits for knowledge

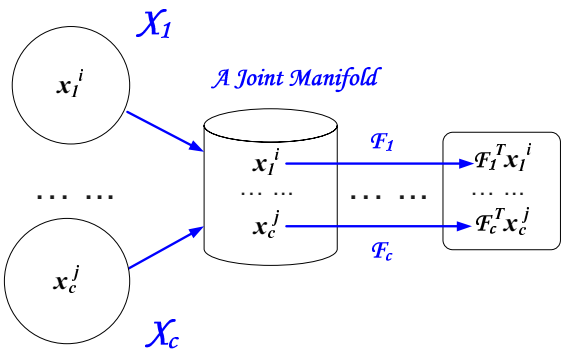

Figure 3: Illustration of the main framework.

1. **Construct the relationship matrices $W_k(k = 1 \cdots c)$ to model the local geometry of each manifold, and $W_{a,b}(a, b \in \{1, \cdots, c\})$ to model the correspondence relationship across manifolds.**
2. **Create the joint manifold:**
   - Compute the matrices L, Z and D. They are used to model the joint structure.
3. **Compute the optimal embedding results (or mapping functions) to reduce the dimensionality of the joint structure:**
   - ***instance-level alignment:*** The d dimensional embedding result is computed by d minimum eigenvectors $\gamma_1 \cdots \gamma_d$ of eigenvalue decomposition $L\gamma = \lambda D\gamma$.
   - ***feature-level alignment:*** The d dimensional mapping function is computed by d minimum eigenvectors $\gamma_1 \cdots \gamma_d$ of the generalized eigenvalue decomposition $ZLZ^T\gamma = \lambda ZDZ^T\gamma$.
4. **Find the correspondence between X_a and X_b:**
   - ***instance-level alignment:*** Let $\mathcal{Y}_k$ be part of $[\gamma_1 \cdots \gamma_d]$ (from row $1 + \sum_{a=1}^{k-1} m_a$ to row $\sum_{a=1}^{k} m_a$). Now $\mathcal{Y}_a^i$ and $\mathcal{Y}_b^j$ are in the same space and can be directly compared.
   - ***feature-level alignment:*** Let $\mathcal{F}_k$ be part of $[\gamma_1 \cdots \gamma_d]$ (from row $1 + \sum_{a=1}^{k-1} p_a$ to row $\sum_{a=1}^{k} p_a$). Now $\mathcal{F}_a^T x_a^i$ and $\mathcal{F}_b^T x_b^j$ are in the same space and can be directly compared.

Figure 4: The Main Framework.

transfer applications better than instance-level alignment. Feature-level alignment can only compute linear mappings for alignment, but it can be easily generalized to new instances and provides a "dictionary" representing direct connections between features in different spaces

Cost functions for instance-level alignment: First, we consider the problem of computing instance-level alignment. The cost function is as follows:

$$C(\mathcal{Y}_1, \cdots, \mathcal{Y}_c) = 0.5\mu \sum_{a=1}^{c}\sum_{b=1}^{c}\sum_{i=1}^{m_a}\sum_{j=1}^{m_b}(\mathcal{Y}_a^i - \mathcal{Y}_b^j)^2 W_{a,b}^{i,j}$$
$$+0.5\sum_{k=1}^{c}\sum_{i=1}^{m_k}\sum_{j=1}^{m_k}(\mathcal{Y}_k^i - \mathcal{Y}_k^j)^2 W_k^{i,j}, \quad (1)$$

where $\mathcal{Y}_k^i$ represents the embedding result of x_k^i (for alignment). The first term of $C(\mathcal{Y}_1, \cdots, \mathcal{Y}_c)$ penalizes the differences between the given manifolds on the mapping results of the corresponding instances. The second term guarantees that the local geometry of each given manifold will be preserved. When $C(\mathcal{Y}_1, \cdots, \mathcal{Y}_c)$ is used, alignment algorithms build connections between instances.

Cost functions for feature-level alignment: Next, we consider the problem of computing feature-level alignment. Here, we compute mapping functions for the computation of embeddings (rather than directly computing an embedding). The cost function for this case is as follows:

$$C(\mathcal{F}_1, \cdots, \mathcal{F}_c) = 0.5\mu \sum_{a=1}^{c}\sum_{b=1}^{c}\sum_{i=1}^{m_a}\sum_{j=1}^{m_b}(\mathcal{F}_a^T x_a^i - \mathcal{F}_b^T x_b^j)^2 W_{a,b}^{i,j}$$
$$+0.5\sum_{k=1}^{c}\sum_{i=1}^{m_k}\sum_{j=1}^{m_k}(\mathcal{F}_k^T x_k^i - \mathcal{F}_k^T x_k^j)^2 W_k^{i,j}, \quad (2)$$

where $\mathcal{F}_k$ is the mapping function to map instances from X_k to the new space: using $\mathcal{F}_k^T x_k^i$ to approximate $\mathcal{Y}_k^i$. In some applications, we also want to consider a possible translation following the linear transform, i.e. using $\mathcal{F}_k^T x_k^i + t_k$ to approximate $\mathcal{Y}_k^i$. In fact, such a translation can be naturally combined with $C(\mathcal{F}_1, \cdots, \mathcal{F}_c)$. Note that $\mathcal{F}_k^T x_k^i + t_k = [\mathcal{F}_k^T, t_k][x_k^{i\,T}, 1]^T$. To consider the translations, what we need to do is to add a new feature (with value 1) to each $x_k^i \in X_k$. The form of $C(\mathcal{F}_1, \cdots, \mathcal{F}_c)$ still holds.

Theorem 1: Eigenvectors corresponding to the minimum eigenvalues of $ZLZ^T\gamma = \lambda ZDZ^T\gamma$ provide optimal linear mappings to align $X_1, \cdots, X_c$ regarding the cost function $C(\mathcal{F}_1, \cdots, \mathcal{F}_c)$.

Proof:

The key part of the proof involves showing the identity $C(\mathcal{F}_1, \cdots, \mathcal{F}_c) = \gamma^T ZLZ^T\gamma$.

The procedure to get this result is lengthy and non-trivial. However, we can verify the result by expanding the right hand side of the equation. The matrix L is used to join the given manifolds such that the underlying structure in

common can be explored.

To remove an arbitrary scaling factor in the embedding, we impose an extra constraint $\sum_{k=1}^{c} \gamma^T X_k D_k X_k^T \gamma = \gamma^T Z D Z^T \gamma = 1$. The matrices D_k $(k = 1, \cdots, c)$ provide a natural measure on the vertices (instances) of the graph. If the value $D_k^{i,i}$ is large, it means x_k^i is more important. Without this constraint, all instances could be mapped to the same location in the new space. A similar constraint is also used in Laplacian eigenmaps (Belkin & Niyogi 2003).

Finally, the optimization problem can be written as:
$\arg\min_{\gamma:\gamma^T Z D Z^T \gamma=1} C(\mathcal{F}_1, \cdots, \mathcal{F}_c) =$
$\arg\min_{\gamma:\gamma^T Z D Z^T \gamma=1} \gamma^T Z L Z^T \gamma$

By using the Lagrange trick, it is easy to see that solution to this equation is the same as the minimum eigenvector solution to $ZLZ^T\gamma = \lambda ZDZ^T\gamma$.

The above proof only shows the 1D alignment achieved by our algorithm framework is optimal. Standard methods show that the solution to find a d dimensional alignment is provided by the eigenvectors corresponding to the d lowest eigenvalues of the same generalized eigenvalue decomposition equation. □

Theorem 2: Eigenvectors corresponding to the minimum eigenvalues of $L\gamma = \lambda D\gamma$ provide optimal embeddings to align $X_1, \cdots, X_c$ regarding the cost function $C(\mathcal{Y}_1, \cdots, \mathcal{Y}_c)$.
Proof: This instance-level alignment problem is simpler than the feature-level alignment. The proof (skipped) is similar to the proof of Theorem 1. □

Algorithms

In this section, we discuss connections between our main framework and previously studied approaches: Laplacian eigenmaps, LPP, semi-supervised manifold alignment and manifold projections. We also discuss how the main algorithm framework can be adapted to handle the situation when no correspondence is available.

1. $c = 1$**.**
Equation (1) and (2) reduce to

$$C(\mathcal{Y}_1) = 0.5 \sum_{i=1}^{m_1} \sum_{j=1}^{m_1} (\mathcal{Y}_1^i - \mathcal{Y}_1^j)^2 W_1^{i,j} \quad (3)$$

and

$$C(\mathcal{F}_1) = 0.5 \sum_{i=1}^{m_1} \sum_{j=1}^{m_1} (\mathcal{F}_1^T x_1^i - \mathcal{F}_1^T x_1^j)^2 W_1^{i,j}, \quad (4)$$

which are exactly the loss functions of Laplacian eigenmaps (Belkin & Niyogi 2003) and LPP (He & Niyogi 2003). When $c = 1$, there is only one given manifold, so the target is simplified to mapping the given data set to a lower dimensional space preserving its local geometry. This is what regular dimensionality approaches are solving. So Laplacian eigenmaps and LPP are obtained as special cases of our framework.

2. $c = 2$**.**
Equation (1) and (2) reduce to

$$C(\mathcal{Y}_1, \mathcal{Y}_2) = \mu \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} (\mathcal{Y}_1^i - \mathcal{Y}_2^j)^2 W_{1,2}^{i,j}$$
$$+0.5 \sum_{i=1}^{m_1} \sum_{j=1}^{m_1} (\mathcal{Y}_1^i - \mathcal{Y}_1^j)^2 W_1^{i,j} + 0.5 \sum_{i=1}^{m_2} \sum_{j=1}^{m_2} (\mathcal{Y}_2^i - \mathcal{Y}_2^j)^2 W_2^{i,j}. \quad (5)$$

and

$$C(\mathcal{F}_1, \mathcal{F}_2) = \mu \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} (\mathcal{F}_1^T x_1^i - \mathcal{F}_2^T x_2^j)^2 W_{1,2}^{i,j}$$
$$+0.5 \sum_{i=1}^{m_1} \sum_{j=1}^{m_1} (\mathcal{F}_1^T x_1^i - \mathcal{F}_1^T x_1^j)^2 W_1^{i,j} + 0.5 \sum_{i=1}^{m_2} \sum_{j=1}^{m_2} (\mathcal{F}_2^T x_2^i - \mathcal{F}_2^T x_2^j)^2 W_2^{i,j}, \quad (6)$$

which are loss functions of semi-supervised manifold alignment (Ham et al. 2005) and manifold projections (Wang & Mahadevan 2009), when correspondence is available. If no correspondence is given, the loss function of manifold projections can be specified in a different manner.

3. Multiple Manifold Alignment ($c \geq 3$)

When $c \geq 3$, the framework can automatically handle multiple manifold alignment problems, which are not well studied yet. Multiple manifold alignment problem arises in many applications, e.g. when we want to find common topics shared by many document collections and when we do data mining in multiple language data sets.

4. Manifold Alignment without Correspondence

A more general manifold alignment problem arises in many real world applications, where manifolds (defined by totally different features) need to be aligned with no available correspondence information. Solving this problem is rather difficult, if not impossible, since there are two unknown variables in this problem: the correspondence and the transformation. The feature-level version of this problem can be more precisely defined as follows: suppose we have c data sets $X_k = \{x_k^1, \cdots, x_k^{m_k}\} (k \in \{1, \cdots, c\})$ for which we want to find correspondence, our aim is to compute functions $\mathcal{F}_k$ to map X_k to a new space such that $\mathcal{F}_a^T x_a^i$ and $\mathcal{F}_b^T x_b^j$ can be directly compared. Its instance-level version can be defined similarly.

Our main framework can be adapted to solve this problem. In our framework, $W_{a,b} (a, b \in \{1, \cdots, c\}, a \neq b)$ are given by the correspondence information, i.e. $W_{a,b}^{i,j} = 1$ when x_a^i matches x_b^j, and 0 otherwise. Now we need to construct $W_{a,b}$ without using correspondence. Our approach to construct $W_{a,b}$ is discussed in (Wang & Mahadevan 2009) in detail. It constructs $W_{a,b}$ by comparing local structures of two manifolds leveraging their local geometries. We use the relation between x_a^i and its neighbors to characterize x_a^i's local geometry. Using relations rather than features to represent local geometry makes the direct comparison of x_a^i and x_b^j be possible. When we use local geometry to specify

$W_{a,b}$, x_a^i might be similar to more than one instance in X_b and it is hard to identify which one is the true match. However, an interesting fact is that solving the original coupled problem could be easier than only finding the true match. The reason is the structures of all input manifolds need to be preserved in the alignment. This helps eliminate many false positive matches. In our main framework, the local pattern generation and comparison can also be application oriented. For example, many existing kernels based on the idea of convolution kernels (Haussler 1999) can be applied here. Choosing other ways to define $W_{a,b}^{i,j}$ does not affect the other parts of the framework.

Experimental Results

In this section, we first use a bioinformatics example to illustrate how our manifold alignment algorithms work, then we apply our approaches to a real world problem in information retrieval: document corpora alignment.

1. Protein Manifold Alignment

In this example, we directly align the given manifolds and use some pictures to illustrate how the manifold alignment algorithms work. The given manifolds come from real protein tertiary structure data.

Protein 3D structure reconstruction is an important step in Nuclear Magnetic Resonance (NMR) protein structure determination. Basically, it finds a map from distances to coordinates. A protein 3D structure is a chain of amino acids. Let n be the number of amino acids in a given protein and $C_1, \cdots, C_n$ be the coordinate vectors for the amino acids, where $C_i = (C_{i,1}, C_{i,2}, C_{i,3})^T$ and $C_{i,1}, C_{i,2}$, and $C_{i,3}$ are the x, y, z coordinates of amino acid i (in biology, one usually uses atom but not amino acid as the basic element in determining protein structure. Since the number of atoms is huge, for simplicity, we use amino acid as the basic element). Then the distance $d_{i,j}$ between amino acids i and j can be defined as $d_{i,j} = \|C_i - C_j\|$. Define $A = \{d_{i,j}, i, j = 1, \cdots, n\}$, and $C = \{C_i, i = 1, \cdots, n\}$. It is easy to see that if C is given, then we can immediately compute A. However, if A is given, it is non-trivial to compute C. The latter problem is called Protein 3D structure reconstruction. In fact, the problem is even more tricky, since only the distances between neighbors are reliable, and this makes A an incomplete distance matrix. The problem has been proved to be NP-complete for general sparse distance matrices (Hogben 2006). In real world, other techniques such as angle constraints and human experience are used together with the partial distance matrix to determine protein structures.

With the information available to us, NMR techniques might find multiple estimations (models), since more than one configuration can be consistent with the distance matrix and the constraints. Thus, the result is an ensemble of models, rather than a single structure. Most usually, the ensemble of structures, with perhaps 10 - 50 members, all of which fit the NMR data and retain good stereochemistry is deposited with the Protein Data Bank (PDB) (Berman et al. 2000). Models related to the same protein should be similar and comparisons between the models in this ensemble provides some information on how well the protein conformation was determined by NMR.

In this test, we study a Glutaredoxin protein PDB-1G7O (this protein has 215 amino acids in total), whose 3D structure has 21 models. We pick up Model 1, Model 21 and Model 10 for test. These models are related to the same protein, so it makes sense to treat them as manifolds to test our techniques. We denote the i^{th} model by Manifold $\mathcal{X}_i$, which is represented by matrix X_i. Obviously, X_1, X_2 and X_3 are all 215×3 matrices. To evaluate how manifold alignment can re-scale manifolds, we manually stretch manifold $\mathcal{X}_1$ by letting $X_1 = 4X_1$, manifold $\mathcal{X}_3$ by letting $X_3 = 2X_3$. The comparison of Manifold $\mathcal{X}_1$ and $\mathcal{X}_2$ (row vectors of X_1 and X_2 represent points in the 3D space) are shown in Figure 5(A). The comparison of all three manifolds are shown in Figure 6(A). In biology, such chains are called protein backbones. It is clear that manifold $\mathcal{X}_1$ is larger than $\mathcal{X}_3$, which is larger than $\mathcal{X}_2$. The orientations of these manifolds are also quite different. To simulate pairwise correspondence information, we uniformly selected 1/4 amino acids as correspondence resulting in three 54×3 matrices.

Procrustes Manifold Alignment:
Since such models are already low dimensional (3D) embeddings of the distance matrices, we skip Step 1 and 2 in Procrustes alignment algorithm (Wang & Mahadevan 2008). We run the algorithm from Step 3 to align $\mathcal{X}_1$ and $\mathcal{X}_2$. Procrustes alignment removes the translational, rotational and scaling components so that the optimal alignment between the instances in correspondence is achieved. The algorithm identifies the re-scale factor k as 4.2971, and the rotation matrix Q as

$$\mathbf{Q} = \begin{pmatrix} 0.56151 & -0.53218 & 0.63363 \\ 0.65793 & 0.75154 & 0.048172 \\ -0.50183 & 0.38983 & 0.77214 \end{pmatrix}.$$

X_2^*, the new representation of X_2, is computed as $X_2^* = kX_2Q$. We plot X_2^* and X_1 in the same graph (Figure 5(B)). The result shows that Manifold $\mathcal{X}_2$ is rotated and enlarged to the similar size as $\mathcal{X}_1$, and now the two manifolds are aligned very well.

Semi-supervised Manifold Alignment:
We tried semi-supervised alignment using the same data and correspondence. The alignment result is shown in Figure 5(C). From the figure, we can see that semi-supervised alignment can map data instances in correspondence to the similar location in the new space, but the instances outside of the correspondence are not aligned well.

Manifold Projections:
We plot 3D (Figure 5(C)), 2D (Figure 5(D)) and 1D (Figure 5(E)) alignment results in Figure 5. nD alignment result is achieved by applying top n minimum eigenvectors. These figures clearly show that the alignment of two different manifolds is achieved by projecting the data (represented by the original features) onto a new space using our carefully generated mapping functions. Compared to 3D alignment result of Procrustes alignment, 3D alignment from manifold projection changes the topologies of both manifolds to make them match. Recall that Procrustes alignment does not change the shapes of the given manifolds. The real

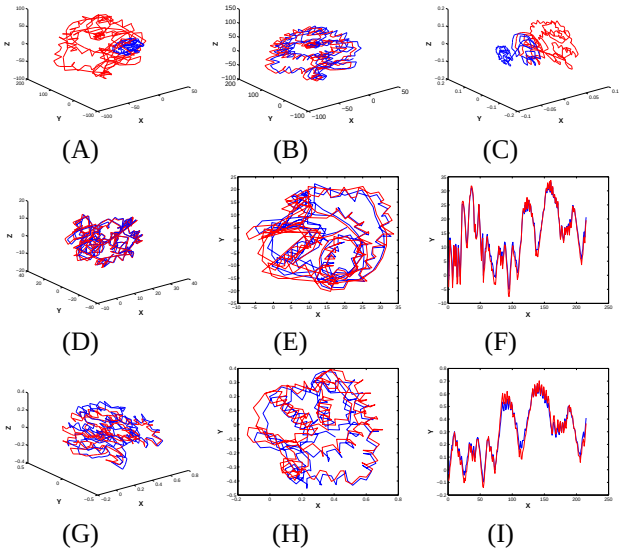

Figure 5: (A): Comparison of Manifold $\mathcal{X}_1$(red) and $\mathcal{X}_2$(blue) before alignment; (B): Procrustes manifold alignment; (C): Semi-supervised manifold alignment; (D): 3D alignment using manifold projections; (E): 2D alignment using manifold projections; (F): 1D alignment using manifold projections; (G): 3D alignment using manifold projections without correspondence; (H): 2D alignment using manifold projections without correspondence; (I): 1D alignment using manifold projections without correspondence.

mapping functions $\mathcal{F}_1$ and $\mathcal{F}_2$ to compute alignment are as follows:

$$\mathcal{F}_1 = \begin{pmatrix} -0.1589 & -0.0181 & -0.2178 \\ 0.1471 & 0.0398 & -0.1073 \\ 0.0398 & -0.2368 & -0.0126 \end{pmatrix},$$

$$\mathcal{F}_2 = \begin{pmatrix} -0.6555 & -0.7379 & -0.3007 \\ 0.0329 & 0.0011 & -0.8933 \\ 0.7216 & -0.6305 & 0.2289 \end{pmatrix}.$$

Manifold Projections without Correspondence: We tested our manifold alignment approach assuming no pairwise correspondence information is given. We plot 3D (Figure 5(G)), 2D (Figure 5H) and 1D (Figure 5(I)) alignment results in Figure 5. nD alignment result is achieved by applying top n minimum eigenvectors. A more detailed description of the setting of this experiment is in (Wang & Mahadevan 2009). These figures show that alignment can still be achieved using local geometry matching algorithm when no pairwise correspondence information is given.

Multiple Manifold Alignment: Our algorithm framework for multiple manifold alignment (using feature-level alignment, $c = 3$) is also tested using all three manifolds. The alignment results are shown in Figure 6. From these figures, we can see that all three manifolds are projected to one space, where alignment is achieved. The mapping functions $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ to compute alignment are as

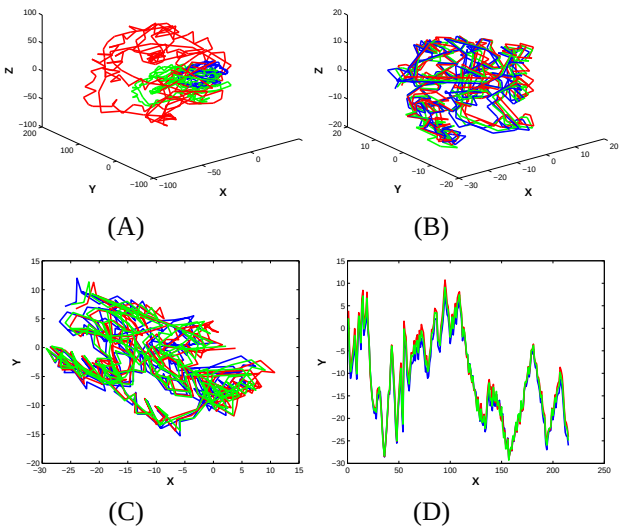

Figure 6: (A): Comparison of Manifold $\mathcal{X}_1$(red), $\mathcal{X}_2$(blue) and $\mathcal{X}_3$(green) before alignment; (B): 3D alignment using multiple manifold alignment; (C): 2D alignment using multiple manifold alignment; (D): 1D alignment using multiple manifold alignment.

follows:

$$\mathcal{F}_1 = \begin{pmatrix} -0.0518 & 0.2133 & 0.0810 \\ -0.2098 & 0.0816 & 0.0046 \\ -0.0073 & -0.0175 & 0.2093 \end{pmatrix},$$

$$\mathcal{F}_2 = \begin{pmatrix} 0.3808 & 0.2649 & 0.6860 \\ -0.7349 & 0.7547 & 0.2871 \\ -0.2862 & -0.3352 & 0.4509 \end{pmatrix},$$

$$\mathcal{F}_3 = \begin{pmatrix} 0.1733 & 0.2354 & -0.0043 \\ -0.3785 & 0.3301 & -0.0787 \\ -0.1136 & 0.1763 & 0.4325 \end{pmatrix}.$$

2. Alignment of Document Corpora

One application of manifold alignment in information retrieval is corpora alignment, where corpora can be aligned so that knowledge transfer between different collections is possible. In this section, we apply our manifold alignment framework (feature-level alignment, $c = 2$) to this problem.

The data set we use in this test is NIPS (1-12) full paper data set: http://www.cs.toronto.edu/∼roweis/data.html. This data set includes 1,740 papers and 2,301,375 tokens. We first represent this data set using two different topic spaces: LSI topic space (Deerwester et al. 1990) and LDA topic space (Blei, Ng, & Jordan 2003). Then the different representations of the original data set are aligned using our manifold alignment algorithm. The reasons why we align such two "simulated" data sets rather than two real data sets are as follows: (1) The two sets are defined by different features: LSI topics and LDA topics, so they are sufficient to test our approaches transferring knowledge

across domains. (2) Even though the representations of the two sets are different, they are constructed from the same data. So the resulting data sets are related and should be aligned well. (3) Both LSI and LDA topics can be mapped back to English words, so the mapping functions are semantically interpretable. This helps us understand how alignment of two collections is achieved (by aligning their underlying topics). (4) The problem (to align two topic spaces) itself is useful, since it computes the topics shared by two collections.

Topic Modeling:
Topic modeling is designed to extract succinct descriptions of the members of a collection that enable efficient generalization and further processing. Topic models are an important tool to find concepts from document corpora. They have been successfully used to analyze large amounts of textual information for many tasks. A topic could be thought as a multinomial word distribution learned from a collection of textual documents using either linear algebraic or statistical techniques. The words that contribute more to each topic provide keywords that briefly summarize the themes in the collection. Popularly used topic models include Latent Semantic Indexing (LSI) (Deerwester et al. 1990) and Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan 2003).

LSI: Latent semantic indexing (LSI) is a well-known linear algebraic method to find topics in a text corpus. The key idea is to map high-dimensional document vectors to a lower dimensional representation in a latent semantic space. Let the singular values of an $n \times m$ term-document matrix A be $\delta_1 \geq \cdots \geq \delta_r$, where r is the rank of A. The singular value decomposition of A is $A = U\Sigma V^T$, where $\Sigma = diag(\delta_1, \cdots \delta_r)$, U is an $n \times r$ matrix whose columns are orthonormal, and V is an $m \times r$ matrix whose columns are also orthonormal. LSI constructs a rank-k approximation of the matrix by keeping the k largest singular values in the above decomposition, where k is usually much smaller than r. More precisely, the best rank-k approximation is given by $A_k = U_k \Sigma_k V_k^T$, and it can be shown that this approximation has the smallest error (w.r.t. Frobenius norm). In LSI, each of the column vectors of U_k is related to a concept, and represents a topic in the given collection of documents.

LDA: Latent Dirichlet Allocation (LDA) is a widely used probabilistic topic model and the basis for many variants. LDA treats each document as a mixture of topics, where each topic is a distribution over words in a vocabulary. To generate a document, LDA first samples a per-document distribution over topics from a Dirichlet distribution, and then it samples a topic from the distribution and a word from the topic. LDA and LSI topics are quite different in many other aspects. For example, LSI topics are orthonormal to each other, while the LDA topics are not; LSI topics are supposed to provide the best low rank approximation to the original set, while LDA is not designed for this.

Represent Corpora in Topic Spaces:
If a topic space $\mathcal{S}$ is spanned by a set of r topic vectors, we write the set as $S = (t(1), \cdots, t(r))$, where topic $t(i)$ is a column vector $(t(i)_1, t(i)_2 \cdots, t(i)_n)^T$. Here n is the size of the vocabulary set, $\|t(i)\| = 1$ and the value of $t(i)_j$ represents the contribution of term j to $t(i)$. Obviously, S is an $n \times r$ matrix. We know the term-document matrix A (an $n \times m$ matrix) models the corpus, where m is the number of the documents and columns of A represent documents in the "term" space. The low dimensional embedding of A in the "topic" space $\mathcal{S}$ is then $A_{Topic} = S^T A$. A_{Topic} is a $r \times m$ matrix, whose columns are the new representations of documents in $\mathcal{S}$.

Table 1: Topic 1-5 (LDA)

Top 9 Terms
generalization function generalize shown performance theory size shepard general
hebbian hebb plasticity activity neuronal synaptic anti hippocampal modification
grid moore methods atkeson steps weighted start interpolation space
measure standard data dataset datasets results experiments measures ranking
energy minimum yuille minima shown local university physics valid

Table 2: Topic 1-5 (LSI)

Top 9 Terms
fish terminals gaps arbor magnetic die insect cone crossing
learning algorithm data model state function models distribution policy
model cells neurons cell visual figure time neuron response
data training set model recognition image models gaussian test
state neural network model time networks control system models

We extract 400 topics from the data set with both LDA and LSI models (in LSI, we simply pick up the top 400 topics; in LDA, we set number of topics = 400). The top 10 words of topic 1-5 from each model are shown in Table 1 and Table 2. It is clear that none of the corresponding topics are similar across the two sets. We represent the original data set in both topic spaces. This step eliminates a lot of information from the original set and can only provide us with an approximation of the original set. We denote the data set represented in LDA topic space manifold $\mathcal{X}_1$ (represented by a $400 \times 1,740$ matrix X_1), in LSI topic space manifold $\mathcal{X}_2$ (represented by a $400 \times 1,740$ matrix X_2).

Manifold Alignment:
Following our main framework (feature-level alignment, $c = 2$) using 25% uniformly selected documents as correspondences, we align these two collections in a 300 dimensional space. The mapping functions $\mathcal{A}$ and $\mathcal{B}$ are both 400×300 matrices. They change the original LDA, LSI topic vectors (defining the original spaces) to vectors spanning the new joint space (latent space). Such vectors can be treated as latent topics (spanning the latent space), which are represented over LDA/LSI topics. We know LDA/LSI topics are represented over English words, so latent topics can also be directly represented with English words. In Table 3 and 4, we show the top 5 latent topics constructed from manifold $\mathcal{X}_1$ (LDA space) and $\mathcal{X}_2$ (LSI space). From these tables, we can see that the corresponding latent topics are very similar to each other. This implies

Table 3: Top 5 latent topics constructed from LDA space

Top 9 Terms
network learning networks training error input neural recurrent output
network neural input networks figure output hierarchical xor neurons
data set training model test models error hmm missing
function figure tangent basis vector measure university theorem average
learning input training figure units visual pattern output unit

Table 4: Top 5 latent topics constructed from LSI space

Top 9 Terms
network learning networks training input error hidden units neural
network neural input output figure networks neurons processing units
data training set model mixture error test models recognition
function theorem approximation figure theory functions process dynamics basis
learning input training figure visual units pattern unit output

that the spaces spanned by two different latent topic sets are almost the same (they approximate the latent space). An interesting thing is that the latent topics constructed from LSI (or LDA) space are linear combinations of the existing LSI (or LDA) topics. So the new space is a subspace of the original LSI (or LDA) space. The alignment of two document collections is in fact done by finding a common topic subspace shared by both LSI and LDA spaces.

Knowledge Transfer Across Data Sets:
We also ran a test to directly translate test documents from LDA space to LSI space using $\mathcal{F}_1\mathcal{F}_2^+$. For each test document x, we compare its mapping result to all documents in LSI space and see if x's true match is among its j nearest neighbors. The results are summarized in Table 5. The results show that for any given a document in LDA space, we can translate it to LSI space. Its translation has a roughly 88% probability of being the nearest neighbor of its true match. This test explains how knowledge is transferred between different topic spaces. The same technique can also be applied to build connection between data sets defined by different languages. The latter is useful in machine translation and cross-lingual information retrieval.

Conclusions

In this paper, we introduce a general framework for manifold alignment. Our framework aligns manifolds by simultaneously matching the corresponding instances and preserving the local geometry of each given manifold. It can also be adapted to solve manifold alignment problems without using correspondence information. Some existing approaches like semi-supervised alignment and manifold projections can be obtained as special cases. Our approaches are described and evaluated both theoretically and experimentally, providing results showing useful knowledge transfer from one domain to another. Sample applications on information retrieval and computational biology are discussed in the paper.

Many real-world data sets exhibit non-trivial regularities at *multiple* levels. To design algorithms capturing such multi-level structure, we are currently exploring the use of diffusion wavelets (Coifman & Maggioni 2006) to explore the intrinsic structure of the joint manifold resulting in manifold alignment at different scales.

Table 5: The probability that x's true match is among $(\mathcal{A}\mathcal{B}^+)^T x$'s j nearest neighbors.

j	1	2	3	4	5
%	87.6628%	89.8084%	90.8046%	91.3410%	91.8774%
j	6	7	8	9	10
%	91.9540%	92.1839%	92.4138%	92.4904%	92.5670%

References

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

Berman, H. M., Westbrook, J., Feng, Z., Gillilandand, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., Bourne, P. E. 2000. The protein data bank. *Nucleic Acids Research*, 28:235–242.

Blei, D., Ng, A., and Jordan, M. 2003. Latent Dirichlet allocation. In *Journal of Machine Learning Research*, volume 3, 993–1022.

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

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6):391–407.

Diaz, F., Metzler, D. 2007. Pseudo-aligned multilingual corpora. *The International Joint Conference on Artificial Intelligence(IJCAI)* 2727-2732.

Ham, J., Lee, D., Saul, L. 2005. Semisupervised alignment of manifolds. 10^{th} *International Workshop on Artificial Intelligence and Statistics.* 120-127.

Haussler, D. 1999. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10.

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

Hogben, L. 2006. Handbook of linear algebra. Chapman/Hall CRC Press.

Lafon, S., Keller, Y., Coifman, R. R. 2006. Data fusion and multi-cue data matching by diffusion maps. *IEEE transactions on Pattern Analysis and Machine Intelligence.* 28(11):1784-1797.

Wang, C., Mahadevan, S. 2008. Manifold alignment using Procrustes analysis. In *Proceedings of the 25th International Conference on Machine Learning*.

Wang, C., Mahadevan, S. 2009. Manifold alignment without correspondence. In *Proceedings of the 21st International Joint Conferences on Artificial Intelligence*. 2009.

Xiong, L., Wang, F., Zhang, C. 2007. Semi-definite manifold alignment. In *Proceedings of the 18th European Conference on Machine Learning*.