Manifold Alignment

Chang Wang, Peter Krafft, and Sridhar Mahadevan

April 10, 2011

Contents

Chapter 5

Manifold Alignment

Manifold alignment, the topic of this chapter, is simultaneously a solution to the problem of alignment and a framework for discovering a unifying representation of multiple datasets. The fundamental ideas of manifold alignment are to utilize the relationships of instances within each dataset to strengthen knowledge of the relationships between the datasets and ultimately to map initially disparate datasets to a joint latent space. At the algorithmic level, the approaches described in this chapter assume that the disparate datasets being aligned have the same underlying manifold structure. The underlying low-dimensional representation is extracted by modeling the local geometry using a graph Laplacian associated with each dataset. After constructing each of these Laplacians, standard manifold learning algorithms are then invoked on a joint Laplacian matrix constructed by concatenating the various Laplacians to obtain a joint latent representation of the original datasets. Manifold alignment can therefore be viewed as a form of constrained joint dimensionality reduction where the goal is to find a low-dimensional embedding of multiple datasets that preserves any known correspondences across them.

5.1 Introduction

As the availability and size of digital information repositories continues to burgeon – from bioinformatics and robotics to sensor and surveillance networks and Internet archives – the problem of extracting deep semantic structure from high-dimensional data becomes more critical. This chapter addresses the fundamental problem of *aligning* multiple data sets to extract shared latent semantic structure. Specifically, the goal of the methods described here is to create a more meaningful representation by aligning multiple data sets. Domains of applicability range across the field of engineering, humanities, and science. Examples include automatic machine translation, bioinformatics, cross-lingual information retrieval, perceptual learning, robotic control, and sensor-based activity modeling. What makes the data alignment problem challenging is that the multiple data streams that need to be coordinated are represented using disjoint features. For example, in cross-lingual information retrieval, it is often desirable to search for documents in a target language (e.g., Italian or Arabic) by typing in queries in English. In activity modeling, the motions of humans engaged in everyday indoor or outdoor activities, such as cooking or walking, is recorded using diverse sensors including audio, video, and wearable devices. Furthermore, as real-world data sets often lie in a high-dimensional space, the challenge is to construct a common semantic representation across heterogeneous data sets by automatically discovering a shared latent space. This chapter describes a *geometric* framework for data alignment, building on recent advances in manifold learning and nonlinear dimensionality reduction using spectral graph-theoretic methods.

The problem of alignment can be formalized as dimensionality reduction with constraints induced by the correspondences among data sets. In many application domains of interest, data appears high-dimensional, but often lies on low-dimensional structures, such as a *manifold*, which can be discretely approximated by a graph [1]. Nonlinear dimensionality reduction methods have recently emerged that empirically model and recover the underlying manifold, including diffusion maps [2], ISOMAP [3], Laplacian eigenmaps [4], LLE [5], Locality Preserving Projections (LPP) [6], and semi-definite embedding (SDE) [7]. When data lies on a manifold, these nonlinear techniques are much more effective than traditional linear methods, such as principal components analysis (PCA) [8]. This chapter describes a novel geometric framework for transfer using *manifold alignment* [9, 10]. Rather than constructing mappings on surface features, which may be difficult due to the high dimensionality of the data, manifold alignment constructs lower-dimensional mappings between two or more disparate data sets by aligning their underlying *learned* manifolds.

Many practical problems, ranging from bioinformatics to information retrieval and robotics, involve modeling multiple datasets that contain significant shared underlying structure. For example, in protein alignment, a set of proteins from a shared family are clustered together by finding correspondences between their three-dimensional structures. In information retrieval, it is often desirable to search documents in a target language, say Italian, given queries in a source language such as English. In robotics, activities can be modeled using parallel data streams such as visual input, audio input, and body posture. Even individual datasets can often be represented using multiple points of view. One example familiar to any calculus student is whether to represent a point in the plane with Cartesian coordinates or polar coordinates. This choice can be the difference between being able to solve a problem and not being able to solve that problem.

The opposite situation also commonly occurs, where multiple apparently different datasets have a single underlying meaning or structure. In this case, finding the "original dataset," the underlying meaning that all of the observed datasets share, may be challenging. Manifold alignment solves this problem by finding a common set of features for those disparate datasets. These features provide coordinates in a single space for the instances in all the related datasets. In other words, manifold alignment discovers a unifying representation of all the initially separate datasets that preserves the qualities of each individual dataset and highlights the similarities between the datasets. Though this new representation may not be the actual "original dataset," if such an entity exists, it should reflect the structure that the original dataset would have.

For example, the Europarl corpus [11] contains a set of documents translated into eleven languages. A researcher may be interested in finding a language-invariant representation of these parallel corpora, for instance, as preprocessing for information retrieval, where different translations of corresponding documents should be close to one another in the joint representation. Using this joint representation, the researcher could easily identify identical or

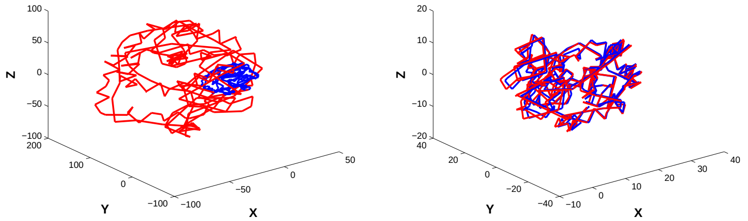


Figure 5.1: A simple example of alignment involving finding correspondences across protein tertiary structures. Here two related structures are aligned. The smaller blue structure is a scaling and rotation of the larger red structure in the original space shown on the left, but the structures are equated in the new coordinate frame shown on the right.

similar documents across languages. Section 5.4.2 describes how manifold alignment solves this problem using a small subset of the languages.

In the less extreme case, the multiple datasets do not have exactly the same underlying meaning but have related underlying structures. For example, two proteins may have related but slightly different tertiary structures, whether from measurement error or because the proteins are actually different but are evolutionarily related (see Figure 5.1). The initial representations of these two datasets, the locations of some points along each protein's structure, may be different, but comparing the local similarities within each dataset reveals that they lie on the same underlying manifold, that is, the relationships between the instances in each dataset are the same. In this case, if the proteins are actually different, there may be no "original dataset" that both observed datasets represent, there may only be a structural similarity between the two datasets which allows them to be represented in similar locations in a new coordinate frame.

Manifold alignment is useful in both of these cases. Manifold alignment preserves similarities within each dataset being aligned and correspondences between the datasets being aligned by giving each dataset a new coordinate frame that reflects that dataset's underlying manifold structure. As such, the main assumption of manifold alignment is that any datasets being aligned must lie on the same low dimensional manifold. Furthermore, the algorithm requires a similarity function that returns the similarity of any two instances within the same dataset with respect to the geodesic distance along that manifold. If these assumptions are met, the new coordinate frames for the aligned manifolds will be consistent with each other and will give a unifying representation.

In some situations, such as the Europarl example, the required similarity function may reflect semantic similarity. In this case, the unifying representation discovered by manifold alignment represents the semantic space of the input datasets. Instances that are close with respect to Euclidean distance in the latent space will be semantically similar, regardless of their original dataset. In other situations, such as the protein example, the underlying manifold is simply a common structure to the datasets, such as related covariance matrices or related local similarity graphs. In this case, the latent space simply represents a new coordinate system for all the instances that is consistent with geodesic similarity along the manifold.

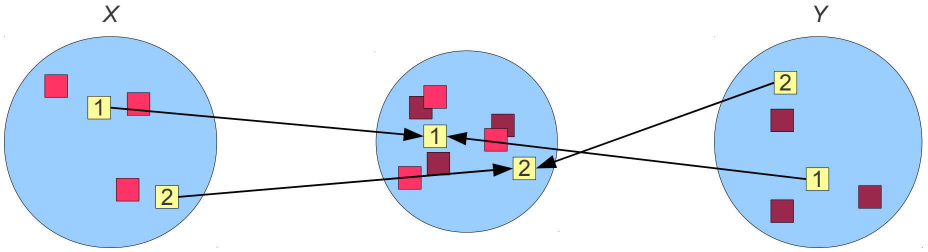


Figure 5.2: Given two datasets X and Y with two instances from both dataset that are known to be in correspondence, manifold alignment embeds all of the instances from each dataset in a new space where the corresponding instances are constrained to be equal (or at least close to each other) and the internal structures of each dataset are preserved.

From an algorithmic perspective, manifold alignment is closely related to other manifold learning techniques for dimensionality reduction such as Isomap [12], locally linear embeddings [13], and Laplacian eigenmaps [14]. Given a dataset, these algorithms attempt to identify the low dimensional manifold structure of that dataset and preserve that structure in a low dimensional embedding of the dataset. Manifold alignment follows the same paradigm but embeds multiple datasets simultaneously. Without any correspondence information (given or inferred), manifold alignment finds independent embeddings of each given dataset, but with some given or inferred correspondence information, manifold alignment includes additional constraints on these embeddings that encourage corresponding instances across datasets to have similar locations in the embedding. Figure 5.2 shows the high level idea of constrained joint embedding.

The remainder of this section provides a more detailed overview of the problem of alignment and the algorithm of manifold alignment. Following these informal descriptions, Section 5.2 develops the formal loss functions for manifold alignment and proves the optimality of the manifold alignment algorithm. Section 5.3 describes four variants of the basic manifold alignment framework. Then, Section 5.4 explores three applications of manifold alignment that illustrate how manifold alignment and its extensions are useful for identifying new correspondences between datasets, performing cross-dataset information retrieval, and performing exploratory data analysis; though, of course, manifold alignment's utility is not limited to these situations. Finally, Section 5.5 summarizes the chapter and discusses some limitations of manifold alignment and Section 5.6 reviews various approaches related to manifold alignment.

5.1.1 Problem Statement

The problem of alignment is to identify a transformation of one dataset that "matches it up" with a transformation another dataset. That is, given two datasets[1], X and Y, whose instances lie on the same manifold, $\mathcal{Z}$, but who may be represented by different features, the problem of alignment is to find two functions f and g, such that $f(x_i)$ is close to $g(y_j)$ in

[1] There is an analogous definition for alignment of multiple datasets. This statement only considers two datasets for simplicity of notation.

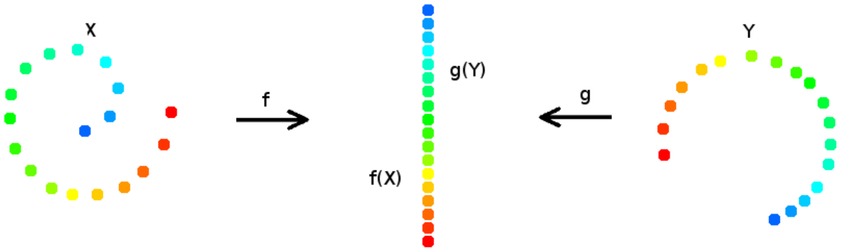


Figure 5.3: An illustration of the problem of manifold alignment. The two datasets X and Y are embedded into a single space where the corresponding instances are equal and local similarities within each dataset are preserved.

terms of Euclidean distance if x_i and y_j are close with respect to geodesic distance along $\mathcal{Z}$. Here, X is an $n \times p$ matrix containing n data instances in p-dimensional space, Y is an $m \times q$ matrix containing m data instances in q-dimensional space, $f : \mathbb{R}^p \rightarrow \mathbb{R}^k$, and $g : \mathbb{R}^q \rightarrow \mathbb{R}^k$ for some k called the latent dimensionality.

The instances x_i and y_j are in exact correspondence if and only if $f(x_i) = g(y_j)$. On the other hand, prior correspondence information includes any information about the similarity of the instances in X and Y, not just exact correspondence information. The union of the range of f and the range of g is the joint latent space, and the concatenation of the new coordinates $\begin{pmatrix} f(X) \\ g(Y) \end{pmatrix}$ is the unified representation of X and Y, where $f(X)$ is an $n \times k$ matrix containing the result f applied to each row of X, and $g(Y)$ an $m \times k$ matrix containing the result of g applied to each row of Y. $f(X)$ and $g(Y)$ are the new coordinates of X and Y in the joint latent space.

5.1.2 Overview of the Algorithm

Manifold alignment is one solution to the problem of alignment. There are two key ideas to manifold alignment: considering local geometry as as well as correspondence information and viewing multiple datasets as being samples on the same manifold. First, instead of only preserving correspondences across datasets, manifold alignment also preserves the individual structures within each dataset by mapping similar instances in each dataset to similar locations in Euclidean space. In other words, manifold alignment maps each dataset to a new joint latent space where locally similar instances within each dataset and given corresponding instances across datasets are close or identical in that space (see Figure 5.3).

This algorithm can be supervised, semi-supervised, or unsupervised. With complete correspondence information, the algorithm is supervised, and it simply finds a unifying representation of all the instances. With incomplete correspondence information, the algorithm is semi-supervised, and it relies only on the known correspondences and the datasets' intrinsic structures to form the embedding. With no correspondence information, the algorithm is unsupervised, and some correspondences must be inferred. Section 5.3.4 discusses one way to infer correspondences.

Second, manifold alignment views each individual dataset as belonging to one larger dataset. Since all the datasets have the same manifold structure, the graph Laplacians associated with each dataset are all discrete approximations of the same manifold, and thus, the diagonal concatenation of these Laplacians, along with the off-diagonal matrices filled with correspondence information, is still an approximation of that manifold. The idea is simple but elegant—by viewing two or more samples as actually being one large sample, making inferences about multiple datasets reduces to making inferences about a single dataset.

Embedding this joint Laplacian combines these ideas. By using a graph embedding algorithm, manifold alignment preserves the local similarities and correspondence information encoded by the joint Laplacian. Thus, by combining these ideas together, the problem of manifold alignment can be reduced to a variant of the standard manifold learning problem.

5.2 Formalization and Analysis

Figure 5.4 summarizes the notation used in this section.

5.2.1 Loss Functions

This section develops the intuition behind the loss function for manifold alignment in two ways, each analogous to one of the two key ideas from Section 5.1.2. The first way illustrates that the loss function captures the idea that manifold alignment should preserve local similarity and correspondence information. The second way illustrates the idea that after forming the joint Laplacian, manifold alignment is equivalent to Laplacian eigenmaps [14]. Subsequent sections use the second approach because it greatly simplifies the notation and the proofs of optimality.

The First Derivation of Manifold Alignment: Preserving Similarities

The first loss function has two parts: one part to preserve local similarity within each dataset and one part to preserve correspondence information about instances across datasets. With c datasets, $X^{(1)}, \ldots, X^{(c)}$, for each dataset the loss function includes a term of the following form:

$$C_\lambda(F^{(a)}) = \sum_{i,j} \|F^{(a)}(i,\cdot) - F^{(a)}(j,\cdot)\|^2 W^{(a)}(i,j),$$

where $F^{(a)}$ is the embedding of the ath dataset and the sum is taken over all pairs of instances in that dataset. $C_\lambda(F^{(a)})$ is the cost of preserving the local similarities within $X^{(a)}$. This equation says that if two data instances from $X^{(a)}$, $X^{(a)}(i,\cdot)$, and $X^{(a)}(j,\cdot)$ are similar, which happens when $W^{(a)}(i,j)$ is larger, their locations in the latent space, $F^{(a)}(i,\cdot)$ and $F^{(a)}(j,\cdot)$, should be closer together.

Additionally, to preserve correspondence information, for each pair of datasets the loss function includes

$$C_\kappa(F^{(a)}, F^{(b)}) = \sum_{i,j} \|F^{(a)}(i,\cdot) - F^{(b)}(j,\cdot)\|^2 W^{(a,b)}(i,j).$$

For any $n \times p$ matrix, M, $M(i,j)$ is the i,jth entry of M, $M(i,\cdot)$ is ith row and $M(\cdot,j)$ is the jth column. $(M)^+$ denotes the Moore-Penrose pseudoinverse. $\|M(i,\cdot)\|$ denotes the l_2 norm. M' denotes the transpose of M.

$X^{(a)}$ is a $n_a \times p_a$ data matrix with n_a observations and p_a features.
$W^{(a)}$ is an $n_a \times n_a$ matrix, where $W^{(a)}(i,j)$ is the similarity of $X^{(a)}(i,\cdot)$ and $X^{(a)}(j,\cdot)$ (could be defined by heat kernel).
$D^{(a)}$ is an $n_a \times n_a$ diagonal matrix: $D^{(a)}(i,i) = \sum_j W^{(a)}(i,j)$.
$L^{(a)} = D^{(a)} - W^{(a)}$ is the Laplacian associated with $X^{(a)}$.

$W_{(a,b)}$ is an $n_a \times n_b$ matrix, where $W^{(a,b)}(i,j) \neq 0$, when $X^{(a)}(i,\cdot)$ and $X^{(b)}(j,\cdot)$ are in correspondence and 0 otherwise. $W^{(a,b)}(i,j)$ is the similarity, or the strength of correspondence, of the two instances. Typically, $W^{(a,b)}(i,j) = 1$ if the instances $X^{(a)}(i,\cdot)$ and $X^{(b)}(j,\cdot)$ are in correspondence.

If c is the number of manifolds being aligned, $\mathbf{X}$ is the joint dataset, a $(\sum_i n_i) \times (\sum_i p_i)$ matrix, and $\mathbf{W}$ is the $(\sum_i n_i) \times (\sum_i n_i)$ joint adjacency matrix,

$$\mathbf{X} = \begin{pmatrix} X^{(1)} & \cdots & 0 \\ & \cdots & \\ 0 & \cdots & X^{(c)} \end{pmatrix}, \quad \mathbf{W} = \begin{pmatrix} \nu W^{(1)} & \mu W^{(1,2)} & \cdots & \mu W^{(1,c)} \\ & & \cdots & \\ \mu W^{(c,1)} & \mu W^{(c,2)} & \cdots & \nu W^{(c)} \end{pmatrix}.$$

ν and μ are scalars that control how much the alignment should try to respect local similarity versus correspondence information. Typically, $\nu = \mu = 1$. Equivalently, $\mathbf{W}$ is a $(\sum_i n_i) \times (\sum_i n_i)$ matrix with zeros on the diagonal and for all i and j,

$$\mathbf{W}(i,j) = \begin{cases} \nu W^{(a)}(i,j) & \text{if } \mathbf{X}(i,\cdot) \text{ and } \mathbf{X}(j,\cdot) \text{ are both from } X^{(a)} \\ \mu W^{(a,b)}(i,j) & \text{if } \mathbf{X}(i,\cdot) \text{ and } \mathbf{X}(j,\cdot) \text{ are corresponding instances from} \\ & X^{(a)} \text{ and } X^{(b)}\text{, respectively} \\ 0 & \text{otherwise,} \end{cases}$$

where the $W^{(a)}(i,j)$ and $W^{(a,b)}(i,j)$ here are an abuse of notation with i and j being the row and column that $\mathbf{W}(i,j)$ came from. The precise notation would be $W^{(a)}(i_a, j_a)$ and $W^{(a,b)}(i_a, j_b)$, where k_g is the index such that $\mathbf{X}(k,\cdot) = [0 \ \ldots \ 0 \ X^{(g)}(k_g,\cdot) \ 0 \ \ldots \ 0]$, $k_g = k - \sum_{l=0}^{g-1} n_l$, $n_0 = 0$.

$\mathbf{D}$ is an $\sum_i n_i \times \sum_i n_i$ diagonal matrix with $\mathbf{D}(i,i) = \sum_j \mathbf{W}(i,j)$.

$\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the joint graph Laplacian

If the dimension of the new space is d, the embedded coordinates are given by

1. in the nonlinear case, $\mathbf{F}$, a $(\sum_i n_i) \times d$ matrix representing the new coordinates.

2. in the linear case, $\mathbf{F}$, a $(\sum_i p_i) \times d$ matrix, where $\mathbf{XF}$ represents the new coordinates.

$F^{(a)}$ or $X^{(a)}F^{(a)}$ are the new coordinates of the dataset $X^{(a)}$.

Figure 5.4: Notation used in this chapter.

$C_\kappa(F^{(a)}, F^{(b)})$ is the cost of preserving correspondence information between $F^{(a)}$ and $F^{(b)}$. This equation says that if two data points, $X^{(a)}(i,\cdot)$ and $X^{(b)}(j,\cdot)$, are in stronger correspondence, which happens when $W^{(a,b)}(i,j)$ is larger, their locations in the latent space, $F^{(a)}(i,\cdot)$ and $F^{(b)}(j,\cdot)$, should be closer together.

The complete loss function is thus

$$\begin{aligned} C_1(F^{(1)}, \ldots, F^{(c)}) &= \nu \sum_a C_\lambda(F^{(a)}) + \mu \sum_{a \neq b} C_\kappa(F^{(a)}, F^{(b)}) \\ &= \nu \sum_a \sum_{i,j} \|F^{(a)}(i,\cdot) - F^{(a)}(j,\cdot)\|^2 W^{(a)}(i,j) \ + \mu \sum_{a \neq b} \sum_{i,j} \|F^{(a)}(i,\cdot) - F^{(b)}(j,\cdot)\|^2 W^{(a,b)}(i,j) \end{aligned}$$

The Second Derivation of Manifold Alignment: Embedding the Joint Laplacian

The second loss function is simply the loss function for Laplacian eigenmaps using the joint adjacency matrix:

$$C_2(\mathbf{F}) = \sum_{i,j} \|\mathbf{F}(i,\cdot) - \mathbf{F}(j,\cdot)\|^2 \mathbf{W}(i,j),$$

where the sum is taken over all pairs of instances from all datasets. Here $\mathbf{F}$ is the unified representation of all the datasets and $\mathbf{W}$ is the joint adjacency matrix. This equation says that if two data instances, $X^{(a)}(i',\cdot)$ and $X^{(b)}(j',\cdot)$, are similar, regardless of whether they are in the same dataset ($a = b$) or from different datasets ($a \neq b$), which happens when $\mathbf{W}(i,j)$ is larger in either case, their locations in the latent space, $\mathbf{F}(i,\cdot)$ and $\mathbf{F}(j,\cdot)$, should be closer together.

Equivalently, making use of the facts that $\|M(i,\cdot)\|^2 = \sum_k M(i,k)^2$ and that the Laplacian is a quadratic difference operator,

$$\begin{aligned} C_2(\mathbf{F}) &= \sum_{i,j} \sum_k [\mathbf{F}(i,k) - \mathbf{F}(j,k)]^2 \mathbf{W}(i,j) \\ &= \sum_k \sum_{i,j} [\mathbf{F}(i,k) - \mathbf{F}(j,k)]^2 \mathbf{W}(i,j) \\ &= \sum_k \operatorname{tr}(\mathbf{F}(\cdot,k)' \mathbf{L} \mathbf{F}(\cdot,k)) \\ &= \operatorname{tr}(\mathbf{F}' \mathbf{L} \mathbf{F}), \end{aligned}$$

where $\mathbf{L}$ is the joint Laplacian of all the datasets.

Overall, this formulation of the loss function says that, given the joint Laplacian, aligning all the datasets of interest is equivalent to embedding the joint dataset according to the Laplacian eigenmap loss function.

Equivalence of the Two Loss Functions

The equivalence of the two loss functions follows directly from the definition of the joint Laplacian, $\mathbf{L}$. Note that for all i and j,

$$\mathbf{W}(i,j) = \begin{cases} \nu W^{(a)}(i,j) & \text{if } \mathbf{X}(i,\cdot) \text{ and } \mathbf{X}(j,\cdot) \text{ are both from } X^{(a)} \\ \mu W^{(a,b)}(i,j) & \text{if } \mathbf{X}(i,\cdot) \text{ and } \mathbf{X}(j,\cdot) \text{ are corresponding instances from} \\ & X^{(a)} \text{ and } X^{(b)}\text{, respectively} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the terms of $C_2(\mathbf{F})$ containing instances from the same dataset are exactly the $C_\lambda(F^{(a)})$ terms, and the terms of containing instances from different datasets are exactly the $C_\kappa(F^{(a)}, F^{(b)})$ terms. Since all other terms are 0,

$$C_1(F^{(1)}, \ldots, F^{(c)}) = C_2(\mathbf{F}) = C(\mathbf{F}).$$

This equivalence means that embedding the joint Laplacian is equivalent to preserving local similarity within each dataset and correspondence information between all pairs of datasets.

Although this loss function captures the intuition of manifold alignment, for it to work mathematically it needs an additional constraint,

$$\mathbf{F}'\mathbf{D}\mathbf{F} = I,$$

where I is the $d \times d$ identity matrix. Without this constraint, the trivial solution of mapping all instances to zero would minimize the loss function.

Thus, the final optimization equation for manifold alignment is

$$\underset{\mathbf{F}:\mathbf{F}'\mathbf{D}\mathbf{F}=I}{\arg\min}\, C(\mathbf{F}) = \underset{\mathbf{F}:\mathbf{F}'\mathbf{D}\mathbf{F}=I}{\arg\min}\, \mathrm{tr}(\mathbf{F}'\mathbf{L}\mathbf{F}).$$

5.2.2 Optimal Solutions

This section derives the optimal solution to optimization problem posed in the previous section using the method of Lagrange multipliers. The technique is well-known in the literature (see for example Bishop's derivation of PCA [15]), and the solution is equivalent to that of Laplacian eigenmaps.

Consider the case when $d = 1$. Then $\mathbf{F}$ is just a vector, f, and

$$\underset{f:f'\mathbf{D}f=1}{\arg\min}\, C(f) = \underset{f}{\arg\min}\;\; f'\mathbf{L}f + \lambda(1 - f'\mathbf{D}f)$$

Differentiating with respect to f and λ and setting equal to zero gives

$$\mathbf{L}f = \lambda\mathbf{D}f$$

and

$$f'\mathbf{D}f = 1.$$

The first equation shows that the optimal f is a solution of the generalized eigenvector problem, $\mathbf{L}f = \lambda\mathbf{D}f$. Multiplying both sides of this equation by f' and using $f'\mathbf{D}f = 1$ gives $f'\mathbf{L}f = \lambda$, which means that minimizing $f'\mathbf{L}f$ requires the smallest nonzero eigenvector.

For $d > 1$, $\mathbf{F} = [f_1, f_2, \ldots, f_d]$, and the optimization problem becomes

$$\arg\min_{\mathbf{F}:\mathbf{F}'\mathbf{D}\mathbf{F}=1} C(\mathbf{F}) = \arg\min_{f_1,\ldots,f_d} \sum_i f_i'\mathbf{L}f_i + \lambda_i(1 - f_i'\mathbf{D}f_i),$$

and the solution is the d smallest nonzero eigenvectors. In this case, the total cost is $\sum_{i=1}^{d} \lambda_i$ if the eigenvalues, $\lambda_1, \ldots, \lambda_n$, are sorted in ascending order and exclude the zero eigenvalues.

5.2.3 The Joint Laplacian Manifold Alignment Algorithm

Using the optimal solution derived in the last section, this section describes the algorithm for manifold alignment using Laplacian eigenmaps on the joint Laplacian.

Given c datasets, $X^{(1)}, \ldots, X^{(c)}$, all lying on the same manifold, a similarity function (or a distance function), S, that returns the similarity of any two instances from the same dataset with respect to geodesic distance along the manifold (perhaps $S = e^{-\|x-y\|}$), and some given correspondence information in the form of pairs of similarities of instances from different datasets, the algorithm is as follows:

1. **Find the adjacency matrices, $W^{(1)}, \ldots, W^{(c)}$, of each dataset using S, possibly only including a weight between two instances if one is in the k-nearest neighbors of the other.**

2. **Construct the joint Laplacian, $\mathbf{L}$.**

3. **Compute the d smallest nonzero eigenvectors of $\mathbf{L}f = \lambda\mathbf{D}f$.**

4. **The rows $1 + \sum_{l=0}^{g-1} n_l, 2 + \sum_{l=0}^{g-1} n_l, \ldots, n_g + \sum_{l=0}^{g-1} n_l$ of $\mathbf{F}$ are the new coordinates of $X^{(g)}$.**

5.3 Variants of Manifold Alignment

There are a number of extensions to the basic joint Laplacian manifold alignment framework. This section explores four important variants: restricted the embedding functions to be linear, enforcing hard constraints on corresponding pairs of instances, finding alignments at multiple scales, and performing alignment with no given correspondence information.

5.3.1 Linear Restriction

In nonlinear manifold alignment, the eigenvectors of the Laplacian are exactly the new coordinates of the embedded instances—there is no simple closed form for the mapping function from the original data to the latent space. Linear manifold alignment, however, enforces an explicit, linear functional form for the embedding function. Besides being useful for making out-of-sample estimates, linear alignment helps diminish the problem of missing correspondence information, finds relationships between the features of multiple datasets instead of just between instances, and attempts to identify a common linear subspace of the original datasets.

This section develops linear alignment in steps analogous to the development of nonlinear alignment with many of the details omitted because of the similarity between the two approaches.

Problem Statement

The problem of linear alignment is slightly different from the general problem of alignment; it is to identify a linear transformation, instead of an arbitrary transformation, of one dataset that best "matches that dataset up" with a linear transformation of another dataset. That is, given two datasets[2], X and Y, whose instances lie on the same manifold, $\mathcal{Z}$, but who may be represented by different features, the problem of linear alignment is to find two matrices F and G, such that x_iF is close to y_jG in terms of Euclidean distance if x_i and y_j are close with respect to geodesic distance along $\mathcal{Z}$.

The Linear Loss Function

The motivation and intuition for linear manifold alignment are the same as for nonlinear alignment. The loss function is thus similar, only requiring an additional term for the linear constraint on the mapping function. The new loss function is

$$C(\mathbf{F}) = \sum_{i \neq j} \|\mathbf{X}(i,\cdot)\mathbf{F} - \mathbf{X}(j,\cdot)\mathbf{F}\|^2 \mathbf{W}(i,j) = \mathrm{tr}(\mathbf{F}'\mathbf{X}'\mathbf{L}\mathbf{X}\mathbf{F}),$$

where the sum is taken over all pairs of instances from all datasets. Once again, the constraint $\mathbf{F}'\mathbf{X}'\mathbf{D}\mathbf{X}\mathbf{F} = I$ allows for nontrivial solutions to the optimization problem. This equation captures the same intuitions as the nonlinear loss function, namely that if $\mathbf{X}(i,\cdot)$ is similar to $\mathbf{X}(j,\cdot)$, which occurs when $\mathbf{W}(i,j)$ is large, the embedded coordinates, $\mathbf{X}(i,\cdot)\mathbf{F}$ and $\mathbf{X}(j,\cdot)\mathbf{F}$, will be closer together, but it restricts the embedding of the $\mathbf{X}$ to being a linear embedding.

Optimal Solutions

Much like nonlinear alignment reduces to Laplacian eigenmaps on the joint Laplacian of the datasets, linear alignment reduces to locality preserving projections [6] on the joint Laplacian of the datasets. The solution to the optimization problem is the minimum eigenvectors of the generalized eigenvector problem:

$$\mathbf{X}'\mathbf{L}\mathbf{X}f = \lambda\mathbf{X}'\mathbf{D}\mathbf{X}f.$$

The proof of this fact is similar to the nonlinear case (just replace the matrix $\mathbf{L}$ in that proof with the matrix $\mathbf{X}'\mathbf{L}\mathbf{X}$, and the matrix $\mathbf{D}$ with $\mathbf{X}'\mathbf{D}\mathbf{X}$).

[2] Once again this definition could include more than two datasets.

Comparison to Nonlinear Alignment

The most immediate practical benefit of using linear alignment is that the explicit functional forms of the alignment functions allow for embedding new instances from any of the datasets into the latent space without having to use an interpolation method. This functional form is also useful for mapping instances from one dataset directly to the space of another dataset. Given some point $X^{(g)}(i,\cdot)$, the function $F^{(g)}(F^{(h)})^+$ maps that point to the coordinate system of $X^{(h)}$. This direct mapping function is useful for transfer. Given some function, f trained on $X^{(h)}$ but which is inconsistent with the coordinate system of $X^{(g)}$ (perhaps f takes input from $\mathbb{R}^3$ but the instances from $X^{(g)}$ are in $\mathbb{R}^4$), $f(X^{(g)}(i,\cdot)F^{(g)}(F^{(h)})^+)$ is an estimate of the value of what $f(X^{(g)}(i,\cdot))$ would be.

Linear alignment is also often more efficient than nonlinear alignment. If $\sum_i p_i \ll \sum_i n_i$, linear alignment will be much faster than nonlinear alignment, since the matrices $\mathbf{X}'\mathbf{L}\mathbf{X}$ and $\mathbf{X}'\mathbf{D}\mathbf{X}$ are $(\sum_i p_i) \times (\sum_i p_i)$ instead of $(\sum_i n_i) \times (\sum_i n_i)$. Of course, these benefits come at a heavy cost if the manifold structure of the original datasets cannot be expressed by a linear function of the original dataset. Linear alignment sacrifices the ability to align arbtitarily warped manifolds. However, as in any linear regression, including nonlinear transformation of the original features of the datasets is one way to circumvent this problem in some cases.

At the theoretical level, linear alignment is interesting because, letting $\mathbf{X}$ be a variable, the linear loss function is a generalization of the simpler, nonlinear loss function. Setting $\mathbf{X}$ to the identity matrix, linear alignment reduces to the nonlinear formulation. This observation highlights the fact that even in the nonlinear case the embedded coordinates, $\mathbf{F}$, are functions—they are functions of the indices of the original datasets.

Other Interpretations

Since each mapping function is linear, the features of the embedded datasets (the projected coordinate systems) are linear combinations of the features of the original datas, which means that another way to view linear alignment and its associated loss function is as a joint feature selection algorithm. Linear alignment thus tries to select the features of the original datasets that are shared across datasets; it tries select a combination of the original features that best respects similarities within and between each dataset. Because of this, examining the coefficients in $\mathbf{F}$ is informative about which features of the original datasets are most important for respecting local similarity and correspondence information. In applications where the underlying manifold of each dataset has a semantic interpretation, linear alignment attempts to filter out the features that are dataset-specific and defines a set of invariant features of the datasets.

Another related interpretation of linear alignment is as feature-level alignment. The function $F^{(g)}(F^{(h)})^+$ that maps the instances of one datasets to the coordinate frame of another dataset also represents the relationship between the features of each of those datasets. For example, if $X^{(2)}$ is a rotation of $X^{(1)}$, $F^{(2)}(F^{(1)})^+$ should ideally be that rotation matrix (it may not be if there is not enough correspondence information or if the dimensionality of the latent space is different from that of the datasets, for example). From a more abstract perspective, each column of the embedded coordinates $\mathbf{XF}$ is composed of a set linear combinations of the columns from each of the original datasets. That is, the columns $F^{(g)}(i,\cdot)$

and $F^{(h)}(i,\cdot)$, which define the ith feature of the latent space, combine some number of the columns from $X^{(g)}(i,\cdot)$ and $X^{(h)}(i,\cdot)$. They unify the features from $X^{(g)}(i,\cdot)$ and $X^{(h)}(i,\cdot)$ into a feature in the latent space. Thus linear alignment defines an alignment of the features of each dataset.

5.3.2 Hard Constraints

In nonlinear alignment, hard constraints can replace some of the soft constraints specified by the loss function. Two instances that should be exactly equal in the latent space can be constrained to be the same by merging them in the joint Laplacian. This merging action forms a new row by combining the individual edge weights in each row of the instances that are equal and removing the original rows of those instances from the joint Laplacian [9]. The eigenvectors of the joint Laplacian will then have one less entry. To recover the full embedding, the final coordinates must include two copies of the embedded merged row, each in the appropriate locations.

5.3.3 Multiscale Alignment

Many real-world data sets exhibit non-trivial regularities at *multiple* levels. For example, for the data set involving abstracts of NIPS conference papers [3], at the most abstract level, the set of all papers can be categorized into two main topics: machine learning and neuroscience. At the next level, the papers can be categorized into a number of areas, such as cognitive science, computer vision, dimensionality reduction, reinforcement learning, etc. To transfer knowledge across domains taking consideration of their intrinsic multilevel structures, we need to develop algorithms for multiscale manifold alignment. All previously studied approaches to manifold alignment are restricted to a single scale. In this section, we discuss how to extend multiscale algorithms such as diffusion wavelets [17] to yield hierarchical solutions to the alignment problem. The goal of this multiscale approach is to produce alignment results that preserve local geometry of each manifold and match instances in correspondence at every scale. Compared to "flat" methods, multiscale alignment automatically generates alignment results at different scales by exploring the intrinsic structures (in common) of the two data sets, avoiding the need to specify the dimensionality of the new space.

Finding multiscale alignments using diffusion wavelets enables a natural multiscale interpretation and give a sparse solution. Multiscale alignment offers additional advantages in transfer learning and in exploratory data analysis. Most manifold alignment methods must be modified to deal with asymmetric similarity relations, which occur when constructing graphs using k-nearest neighbor relationships, in directed citation and web graphs, in Markov decision processes, and in many other applications. In contrast to most manifold alignment methods, multiscale alignment using diffusion wavelets can be used without modification, although there is no optimality guarantee in that case. Furthermore, multiscale alignment is useful to exploratory data analysis because it generates a hierarchy of alignments that reflects a hierarchical structure common to the datasets of interest.

[3] Available at www.cs.toronto.edu/∼roweis/data.html.

Intuitively, multiscale alignment is appealing because many datasets show regularity at multiple scales. For example, in the NIPS conference paper dataset[4], there are two main topics at the most abstract level: machine learning and neuroscience. At the next level, the papers fall into a number of categories, such as dimensionality reduction or reinforcement learning. Another dataset with a similar topic structure should be able to be aligned at each of these scales. Multiscale manifold alignment simultaneously extracts this type of structure across all datasets of interest.

This section formulates the problem of multiscale alignment using the framework of multiresolution wavelet analysis [17]. In contrast to "flat" alignment methods which result in a single latent space for alignment in a pre-selected dimension, multiscale alignment using diffusion wavelets automatically generates alignment results at different levels by discovering the shared intrinsic multilevel structures of the given datasets. This multilevel approach results in multiple alignments in spaces of different dimension, where the dimensions are automatically decided according to a precision term.

Problem Statement

Given a fixed sequence of dimensions, $d_1 > d_2 > \ldots > d_h$, as well as two datasets, X and Y, and some partial correspondence information, $x_i \in X_l \longleftrightarrow y_i \in Y_l$, the multiscale manifold alignment problem is to compute mapping functions, $\mathcal{A}_k$ and $\mathcal{B}_k$, at each level k ($k = 1, 2, \ldots, h$) that project X and Y to a new space, preserving local geometry of each dataset and matching instances in correspondence. Furthermore, the associated sequence of mapping functions should satisfy $span(\mathcal{A}_1) \supseteq span(\mathcal{A}_2) \supseteq \ldots \supseteq span(\mathcal{A}_h)$ and $span(\mathcal{B}_1) \supseteq span(\mathcal{B}_2) \supseteq \ldots \supseteq span(\mathcal{B}_h)$, where $span(\mathcal{A}_i)$ (or $span(\mathcal{B}_i)$) represents the subspace spanned by the columns of $\mathcal{A}_i$ (or $\mathcal{B}_i$).

This view of multiscale manifold alignment consists of two parts: (1) determining a hierarchy in terms of number of levels and the dimensionality at each level and (2) finding alignments to minimize the cost function at each level. Our approach solves both of these problems simultaneously while satisfying the subspace hierarchy constraint.

Optimal Solutions

There is one key property of diffusion wavelets that needs to be emphasized. Given a diffusion operator T, such as a random walk on a graph or manifold, the diffusion wavelet (DWT) algorithm produces a subspace hierarchy associated with the eigenvectors of T (if T is symmetric). Letting λ_i be the eigenvalue associated with the ith eigenvector of T, the kth level of the DWT hiearchy is spanned by the eigenvectors of T with $\lambda_i^{2^k} \geq \epsilon$, for some precision parameter, ϵ. Although each level of the hierarchy is spanned by a certain set of eigenvectors, the DWT algorithm returns a set of scaling functions, ϕ_k, at each level, which span the same space as the eigenvectors but have some desirable properties.

To apply diffusion wavelets to multiscale alignment problem, the algorithm must address the following challenge: the regular diffusion wavelets algorithm can only handle regular eigenvalue decomposition in the form of $A\gamma = \lambda\gamma$, where A is the given matrix, γ is an eigenvector and λ is the corresponding eigenvalue. However, the problem we are interested

[4] www.cs.toronto.edu/∼roweis/data.html

in is a generalized eigenvalue decomposition, $A\gamma = \lambda B\gamma$, where we have two input matrices A and B. This multiple manifold alignment algorithm overcomes this challenge.

The Main Algorithm

Using the notation defined in Figure 5.4, the algorithm is as follows:

1. **Construct a matrix representing the joint manifold, L.**
2. **Find an $(\sum p_i) \times r$ matrix, G, such that $G'G = \mathbf{X}'\mathbf{X}$ using SVD.**
3. **Define $T = ((G')^+\mathbf{X}'\mathbf{L}\mathbf{X}G^+)^+$.**
4. **Use diffusion wavelets to explore the intrinsic structure of the joint manifold:** $[\phi_k]_{\phi_0} = \mathcal{DWT}(T^+, \epsilon)$, where $\mathcal{DWT}()$ is the diffusion wavelets implementation described in [17] with extraneous parameters omitted. $[\phi_k]_{\phi_0}$ are the scaling function bases at level k represented as an $r \times d_k$ matrix, $k = 1, \ldots, h$.
5. **Compute mapping functions for manifold alignment (at level k): $\mathbf{F}_k = (G)^+[\phi_k]_{\phi_0}$.**

Benefits

As discussed in [17], the benefits of using diffusion wavelets are:

- Wavelet analysis generalizes to asymmetric matrices.
- Diffusion wavelets result in sets of mapping functions that capture different spectral bands of the relevant operator.
- The basis vectors in ϕ_k are localized (sparse).

5.3.4 Unsupervised Alignment

Performing unsupervised alignment requires generating the portions of the joint Laplacian that represent the between-dataset similarities. One way to do this is by local pattern matching. With no given correspondence information, if the datasets $X^{(a)}$ and $X^{(b)}$ are represented by different features, there is no easy way to directly compare $X^{(a)}(i,\cdot)$ and $X^{(b)}(j,\cdot)$. One way to build connections between them is to use the relations between $X^{(a)}(i,\cdot)$ and its neighbors to characterize $X^{(a)}(i,\cdot)$'s local geometry. Using relations rather than features to represent local geometry makes the direct comparison of $X^{(a)}(i,\cdot)$ and $X^{(b)}(j,\cdot)$ possible. After generating correspondence information using this approach, any of the previous manifold alignment algorithms work. This section shows how to compute local patterns representing local geometry, and shows that these patterns are valid for comparison across datasets.

Given $X^{(a)}$, pattern matching first constructs an $n_a \times n_a$ distance matrix $Distance_a$, where $Distance_a(i,j)$ is Euclidean distance between $X^{(a)}(i,\cdot)$ and $X^{(a)}(j,\cdot)$. The algorithm then decomposes this matrix into elementary contact patterns of fixed size $k+1$. $R_{X^{(a)}(i,\cdot)}$ is a $(k+1) \times (k+1)$ matrix representing the local geometry of $X^{(a)}(i,\cdot)$.

$$R_{X^{(a)}(i,\cdot)}(u,v) = distance(z_u, z_v),$$

where $z_1 = X^{(a)}(i,\cdot)$ and $z_2, \dots, z_{k+1}$ are $X^{(a)}(i,\cdot)$'s k nearest neighbors. Similarly, $R_{X^{(b)}(j,\cdot)}$ is a $(k+1)\times(k+1)$ matrix representing the local geometry of $X^{(b)}(j,\cdot)$. The order of $X^{(b)}(j,\cdot)$'s k nearest neighbors have $k!$ permutations, so $R_{X^{(b)}(j,\cdot)}$ has $k!$ variants. Let $\{R_{X^{(b)}(j,\cdot)}\}_h$ denote its h^{th} variant.

Each local contact pattern $R_{X^{(a)}(i,\cdot)}$ is represented by a submatrix, which contains all pairwise distances between local neighbors around $X^{(a)}(i,\cdot)$. Such a submatrix is a 2D representation of a high dimensional substructure. It is independent of the coordinate frame and contains enough information to reconstruct the whole manifold. $X^{(b)}$ is processed similarly and distance between $R_{X^{(a)}(i,\cdot)}$ and $R_{X^{(b)}(j,\cdot)}$ is defined as follows:

$$dist(R_{X^{(a)}(i,\cdot)}, R_{X^{(b)}(j,\cdot)}) = \min_{1\le h\le k!} \min(dist_1(h), dist_2(h)),$$

where

$$\begin{aligned}dist_1(h) &= \|\{R_{X^{(b)}(j,\cdot)}\}_h - k_1 R_{X^{(a)}(i,\cdot)}\|_F,\\ dist_2(h) &= \|R_{X^{(a)}(i,\cdot)} - k_2 \{R_{X^{(b)}(j,\cdot)}\}_h\|_F,\\ k_1 &= \mathrm{tr}(R'_{X^{(a)}(i,\cdot)}\{R_{X^{(b)}(j,\cdot)}\}_h)/\mathrm{tr}(R'_{X^{(a)}(i,\cdot)}R_{X^{(a)}(i,\cdot)}),\\ k_2 &= \mathrm{tr}(\{R_{X^{(b)}(j,\cdot)}\}'_h R_{X^{(a)}(i,\cdot)})/\mathrm{tr}(\{R_{X^{(b)}(j,\cdot)}\}'_h\{R_{X^{(b)}(j,\cdot)}\}_h).\end{aligned}$$

Finally, $W^{(a,b)}$ is computed as follows:

$$W^{(a,b)}(i,j) = e^{-dist(R_{X^{(a)}(i,\cdot)}, R_{X^{(b)}(j,\cdot)})/\delta^2}.$$

Theorem: Given two $(k+1)\times(k+1)$ distance matrices R_1 and R_2, $k_2 = \mathbf{tr}(R_2'R_1)/\mathbf{tr}(R_2'R_2)$ minimizes $\|R_1 - k_2R_2\|_F$ and $k_1 = \mathbf{tr}(R_1'R_2)/\mathbf{tr}(R_1'R_1)$ minimizes $\|R_2 - k_1R_1\|_F$.

Proof:

Finding k_2 is formalized as

$$k_2 = \arg\min_{k_2} \|R_1 - k_2R_2\|_F,$$

where $\|\cdot\|_F$ represents Frobenius norm.

It is easy to verify that

$$\|R_1 - k_2R_2\|_F = \mathrm{tr}(R_1'R_1) - 2k_2\mathrm{tr}(R_2'R_1) + k_2^2\mathrm{tr}(R_2'R_2).$$

Since $\mathrm{tr}(R_1'R_1)$ is a constant, the minimization problem is equal to

$$k_2 = \arg\min_{k_2} k_2^2\mathrm{tr}(R_2'R_2) - 2k_2\mathrm{tr}(R_2'R_1).$$

Differentiating with respect to k_2 gives

$$2k_2\mathrm{tr}(R_2'R_2) = 2\mathrm{tr}(R_2'R_1),$$

which implies

$$k_2 = \mathrm{tr}(R_2'R_1)/\mathrm{tr}(R_2'R_2).$$

Similarly,

$$k_1 = \mathrm{tr}(R_1'R_2)/\mathrm{tr}(R_1'R_1).$$

□

To compute matrix $W^{(a,b)}$, the algorithm needs to compare all pairs of local patterns. When comparing local pattern $R_{X^{(a)}(i,\cdot)}$ and $R_{X^{(b)}(j,\cdot)}$, the algorithm assumes $X^{(a)}(i,\cdot)$ matches $X^{(b)}(j,\cdot)$. However, the algorithm does not know how $X^{(a)}(i,\cdot)$'s k neighbors match $X^{(b)}(j,\cdot)$'s k neighbors. To find the best possible match, it considers all $k!$ possible permutations, which is tractable since k is always small.

$R_{X^{(a)}(i,\cdot)}$ and $R_{X^{(b)}(j,\cdot)}$ are from different manifolds, so their sizes could be quite different. The previous theorem shows how to find the best re-scaler to enlarge or shrink one of them to match the other. Showing that $dist(R_{X^{(a)}(i,\cdot)}, R_{X^{(b)}(j,\cdot)})$ considers all the possible matches between two local patterns and returns the distance computed from the best possible match is straightforward.

5.4 Application Examples

5.4.1 Protein Alignment

One simple application of alignment is aligning the three-dimensional structures of proteins. This example shows how alignment can identify the corresponding parts of datasets.

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 $C(1,\cdot),\cdots,C(n,\cdot)$ be the coordinate vectors for the amino acids, where $C(i,\cdot) = (C(i,1), C(i,2), C(i,3))$ 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 $d(i,j) = \|C(i,\cdot) - C(j,\cdot)\|$. Define $A = \{d(i,j) \mid i,j = 1,\cdots,n\}$, and $C = \{C(i,\cdot) \mid i = 1,\cdots,n\}$. 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 [18]. In the 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) [19]. 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 data matrices $X^{(1)}$, $X^{(2)}$ and $X^{(3)}$. $X^{(1)}$, which are all 215×3 matrices. To evaluate how manifold alignment can re-scale manifolds, we multiply two of the datasets by a constant, $X^{(1)} = 4X^{(1)}$ and $X^{(3)} = 2X^{(3)}$. The comparison of $X^{(1)}$ and $X^{(2)}$ (row vectors of $X^{(1)}$ and $X^{(2)}$ represent points in the 3D space) is shown in Figure 5.5(A). The comparison of all three manifolds are shown in Figure 5.6(A). In biology, such chains are called protein backbones. These pictures show that the rescaled protein represented by $X^{(1)}$ is larger than that of $X^{(3)}$, which is larger than that of $X^{(2)}$. The orientations of these proteins are also different. To simulate pairwise correspondence information, we uniformly selected a fourth of the amino acids as correspondence resulting in three 54×3 matrices. We compare the results of five alignment approaches on these datasets.

Procrustes Manifold Alignment

One of the simplest alignment algorithm is Procrustes alignment [10]. Since such models are already low dimensional (3D) embeddings of the distance matrices, we skip Step 1 and 2 in Procrustes alignment algorithm, which are normally used to get an initial low dimension embedding of the datasets. We run the algorithm from Step 3, which attempts to find a rotation matrix that best aligns two datasets $X^{(1)}$ and $X^{(2)}$. 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

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

$Y^{(2)}$, the new representation of $X^{(2)}$, is computed as $Y^{(2)} = kX^{(2)}Q$. We plot $Y^{(2)}$ and $X^{(1)}$ in the same graph (Figure 5.5(B)). The plot shows that after the second protein is rotated and rescaled to be the similar size as the first protein, the two proteins are aligned well.

Semi-supervised Manifold Alignment

Next we compare the result of nonlinear alignment, also called semi-supervised alignment [9], using the same data and correspondence. The alignment result is shown in Figure 5.5(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

Next we show the results for linear alignment, also called manifold projections. The 3D (Figure 5.5(C)), 2D (Figure 5.5(D)) and 1D (Figure 5.5(E)) alignment results are shown in Figure 5.5. 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 the 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 mapping functions $F^{(1)}$ and $F^{(2)}$ to compute the alignment are

$$F^{(1)} = \begin{pmatrix} -0.1589 & -0.0181 & -0.2178 \\ 0.1471 & 0.0398 & -0.1073 \\ 0.0398 & -0.2368 & -0.0126 \end{pmatrix}, \; F^{(2)} = \begin{pmatrix} -0.6555 & -0.7379 & -0.3007 \\ 0.0329 & 0.0011 & -0.8933 \\ 0.7216 & -0.6305 & 0.2289 \end{pmatrix}.$$

Manifold Alignment without Correspondence

We also tested the unsupervised manifold alignment approach assuming no given pairwise correspondence information. We plot 3D (Figure 5.5(G)), 2D (Figure 5.5H) and 1D (Figure 5.5(I)) alignment results in Figure 5.5. These figures show that alignment can still be achieved using local geometry matching algorithm when no pairwise correspondence information is given.

Multiple Manifold Alignment

Finally, we show the algorithm with all three datasets (using feature-level alignment, $c = 3$) is also tested using all three manifolds. The alignment results are shown in Figure 5.6. From these figures, we can see that all three manifolds are projected to one space, where alignment is achieved. The mapping functions $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$ to compute alignment are as follows:

$$F^{(1)} = \begin{pmatrix} -0.0518 & 0.2133 & 0.0810 \\ -0.2098 & 0.0816 & 0.0046 \\ -0.0073 & -0.0175 & 0.2093 \end{pmatrix}, \; F^{(2)} = \begin{pmatrix} 0.3808 & 0.2649 & 0.6860 \\ -0.7349 & 0.7547 & 0.2871 \\ -0.2862 & -0.3352 & 0.4509 \end{pmatrix},$$

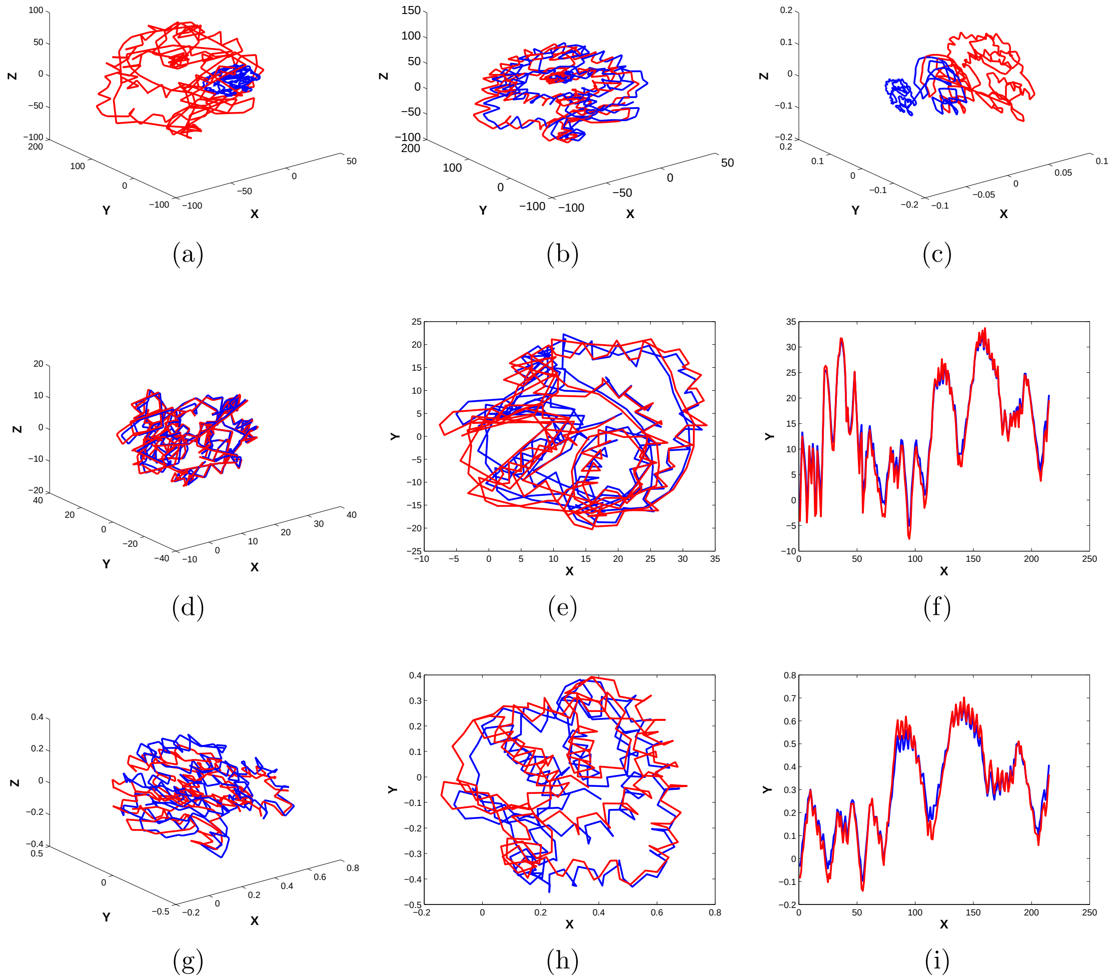
$$F^{(3)} = \begin{pmatrix} 0.1733 & 0.2354 & -0.0043 \\ -0.3785 & 0.3301 & -0.0787 \\ -0.1136 & 0.1763 & 0.4325 \end{pmatrix}.$$

5.4.2 Parallel Corpora

Assuming that similar documents have similar word usage within each language, we can generate eleven graphs, one for each language, that reflect the semantic similarity of the documents.

The data we use in this test is a collection of the proceedings of the European Parliament [11], dating from 04/1996 to 10/2006. The corpus includes versions in 11 European languages: French, Italian, Spanish, Portuguese, English, Dutch, German, Danish, Swedish, Greek and Finnish. Altogether, the corpus comprises of about 30 million words for each language.

The data for our experiments came from the English-Italian parallel corpora, each of which has more than 36,000,000 words. The data set has many files, each file contains the utterances of one speaker in turn. We treat an utterance as a document. We first extracted English-Italian document pairs where both documents have at least 100 words.

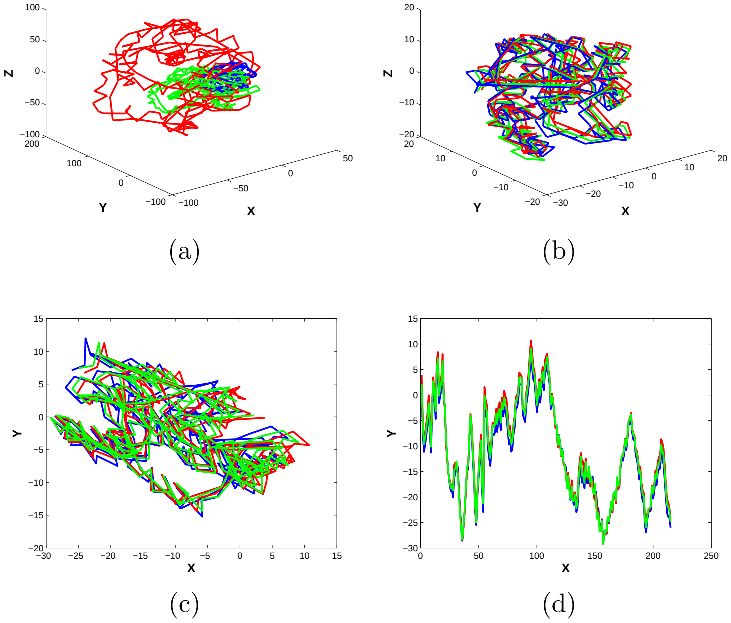


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5.5: (a): Comparison of proteins $X^{(1)}$ (red) and $X^{(2)}$ (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.



(a) (b)

(c) (d)

Figure 5.6: (a): Comparison of the proteins $X^{(1)}$ (red), $X^{(2)}$ (blue) and $X^{(3)}$ (green) before alignment; (b): 3D alignment using multiple manifold alignment; (c): 2D alignment using multiple manifold alignment; (d): 1D alignment using multiple manifold alignment.

This resulted in 59,708 document pairs. We then represented each English document with the most commonly used 4,000 English words, each Italian document with the most commonly used 4,000 Italian words. The documents are represented as bags of words, and no tag information is included. 10,000 resulting document pairs are used for training and the remaining 49,708 document pairs are held for testing. To our knowledge, no one has ever used a data set at this scale to test manifold alignment approaches.

We first tested our algorithmic framework using this data set. In this test, the only parameter we need to set is $d = 200$, i.e. we map two manifolds to the same 200 dimensional space. In The other parameters directly come with the input data sets $X^{(1)}$ (for English) and $X^{(2)}$ (for Italian): $p_1 = p_2 = 4000$; $n_1 = n_2 = 10,000$; $c = 2$; $W^{(1)}$ and $W^{(2)}$ are constructed using heat kernels, where $\delta = 1$; $W^{(1,2)}$ is given by the training correspondence information. Since the number of documents is huge, we only do feature-level alignment, which results in mapping functions $F^{(1)}$ (for English) and $F^{(2)}$ (for Italian). These two mapping functions map documents from the original English language/ Italian language spaces to the new latent 200 dimensional space. The procedure for the test is as follows: for each given English document, we retrieve its top k most similar Italian documents in the new latent space. The probability that the true match is among the top k documents is used to show the goodness of the method. The results are summarized in Table 5.7. If we retrieve the most relevant Italian document, then the true match has a 86% probability of being retrieved. If we retrieve 10, this probability jumps to 90%. Different from most approaches in cross-lingual knowledge transfer, we are not using any method from informational retrieval area to tune

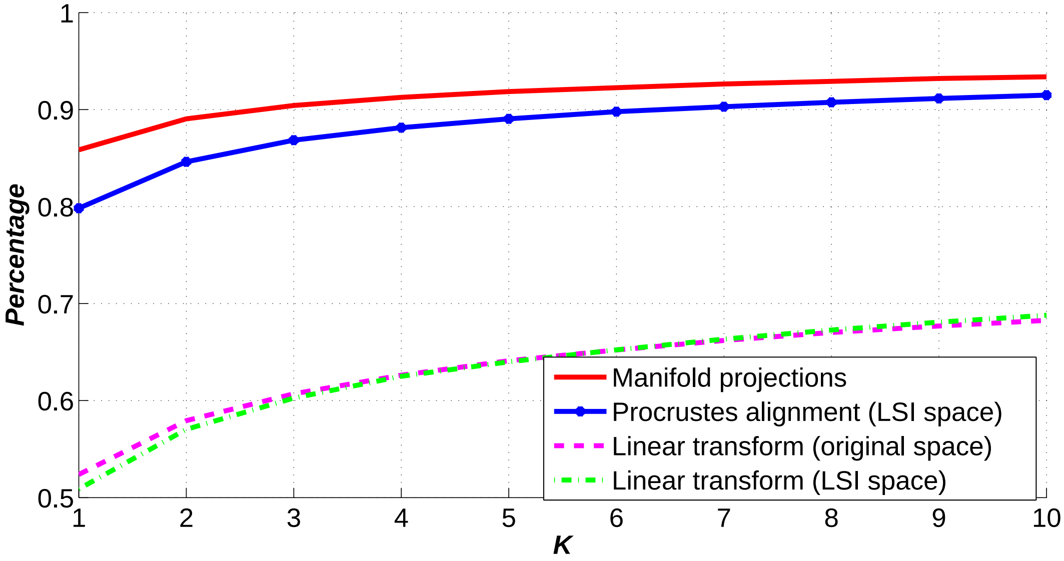


Figure 5.7: EU parallel corpus alignment test.

our framework to this task. For the purpose of comparison, we also used linear transform $\mathcal{F}$ to directly align two corpora, where $X^{(1)}\mathcal{F}$ is used to approximate $X^{(2)}$. This is a regular least square problem and the solution is given by $\mathcal{F} = (X^{(1)})^{+}X^{(2)}$, which is a $4,000 \times 4,000$ matrix for our case. The result of this approach is roughly 35% worse than manifold alignment approach. The true match has a 52% probability of being the first retrieved document. We also applied LSI [20] to preprocess the data and mapped the documents to a 200 dimensional LSI space. Procrustes alignment and Linear transform were then applied to align the corpora in the 200 dimensional spaces. The result of Procrustes alignment (Figure 5.7) is roughly 6% percent worse than manifold projections. Performance of linear transform in LSI space is almost the same as the linear transform result in the original space. There are two reasons why manifold alignment approaches perform much better than the regular linear transform approaches: (1) manifold alignment approach preserves the topologies of the given manifolds in the computation of alignment. This lowers the chance of getting into "overfitting" problems. (2) manifold alignment maps the data to a lower dimensional space, getting rid of the information that does not model the common underlying structure of the given manifolds. In manifold projections, each column of $F^{(1)}$ is a $4,000 \times 1$ vector. Each entry on this vector corresponds to a word. To illustrate how the alignment is achieved using our approach, we show 5 selected corresponding columns of $F^{(1)}$ and $F^{(2)}$ in Table 5.1 and 5.2. From these tables, we can see that our approach can automatically map the words with similar meanings from different language spaces to similar locations in the new space.

5.4.3 Aligning Topic Models

Next, we applied the diffusions wavelet-based multiscale alignment algorithm to align corpora represented in different topic spaces. We show that the alignment is useful for finding topics

	Top 10 Terms
1	ahern tuberculosis eta watts dublin wogau october september yielded structural
2	lakes vienna a4 wednesday chirac lebanon fischler ahern vaccines keys
3	scotland oostlander london tuberculosis finns chirac vaccines finland lisbon prosper
4	hiv jarzembowski tuberculosis mergers virus adjourned march chirac merger parents
5	corruption jarzembowski wednesday mayor parents thursday rio oostlander ruijten vienna

Table 5.1: 5 selected mapping functions for English Corpus

	Top 10 Terms
1	ahern tubercolosi eta watts dublino ottobre settembre wogau carbonica dicembre
2	laghi vienna mercoledi a4 chirac ahern vaccini libano fischler svedese
3	tubercolosi scozia oostlander londra finlandesi finlandia chirac lisbona vaccini svezia
4	hiv jarzembowski fusioni tubercolosi marzo chirac latina genitori vizioso venerdi
5	corruzione mercoledi jarzembowski statistici sindaco rio oostlander limitiamo concentrati vienna

Table 5.2: 5 selected mapping functions for Italian Corpus

shared by the different topic extraction methods, which suggests that alignment may be useful for integrating multiple topic spaces. Since we align the topic spaces at multiple levels, the alignment results are also useful for exploring the hierarchical topic structure of the data.

Given two collections, $X^{(1)}$ (a $n_1 \times p_1$ matrix) and $X^{(2)}$ (a $n_2 \times p_2$ matrix), where p_i is the size of the vocabulary set and n_i is the number of the documents in collection $X^{(i)}$, assume the topics learned from the two collections are given by S_1 and S_2, where S_i is a $p_i \times r_i$ matrix and r_i is the number of the topics in $X^{(i)}$. Then the representations of $X^{(i)}$ in the topic space is $X^{(i)}S_i$. Following our main algorithm, $X^{(1)}S_1$ and $X^{(2)}S_2$ can be aligned in the latent space at level k by using mapping functions $F_k^{(1)}$ and $F_k^{(2)}$. The representations of $X^{(1)}$ and $X^{(2)}$ after alignment become $X^{(1)}S_1F_k^{(1)}$ and $X^{(2)}S_2F_k^{(2)}$. The document contents ($X^{(1)}$ and $X^{(2)}$) are not changed. The only thing that has been changed is S_i, the topic matrix. Recall that the columns of S_i are topics of $X^{(i)}$. The alignment algorithm changes S_1 to $S_1F_k^{(1)}$ and S_2 to $S_2F_k^{(2)}$. The columns of $S_1F_k^{(1)}$ and $S_2F_k^{(2)}$ are still of length p_i. Such columns are in fact the new "aligned" topics.

In this application, we used the NIPS (1-12) full paper dataset, which includes 1,740 papers and 2,301,375 tokens in total. We first represented this dataset using two different topic spaces: LSI space [20] and LDA space [21]. In other words, $X^{(1)} = X^{(2)}$, but $S_1 \neq S_2$ for this set. The reasons for aligning these two datasets is that while they define different features, they are constructed from the same data, and hence admit a correspondence under which the resulting datasets should be aligned well. Also, LSI and LDA topics can be mapped back to the English words, so the mapping functions are semantically interpretable. This helps us understand how the alignment of two collections is achieved (by aligning their underlying topics). We extracted 400 topics from the dataset with both LDA and LSI models ($r_1 = r_2 = 400$). The top eight words of the first five topics from each model are shown in Figure 5.8a and Figure 5.8b. It is clear that none of those topics are similar across the two sets. We ran the main algorithm ($\mu = \nu = 1$) using 20% uniformly selected documents as

Top 8 Terms
generalization function generalize shown performance theory size shepard
hebbian hebb plasticity activity neuronal synaptic anti hippocampal
grid moore methods atkeson steps weighted start interpolation
measure standard data dataset datasets results experiments measures
energy minimum yuille minima shown local university physics

(a) Topic 1-5 (LDA) before alignment.

Top 8 Terms
fish terminals gaps arbor magnetic die insect cone
learning algorithm data model state function models distribution
model cells neurons cell visual figure time neuron
data training set model recognition image models gaussian
state neural network model time networks control system

(b) Topic 1-5 (LSI) before alignment.

Top 8 Terms
road car vehicle autonomous lane driving range unit
processor processors brain ring computation update parallel activation
hopfield epochs learned synapses category modulation initial pulse
brain loop constraints color scene fig conditions transfer
speech capacity peak adaptive device transition type connections

(c) 5 LDA topics at level 2 after alignment.

Top 8 Terms
road autonomous vehicle range navigation driving unit video
processors processor parallel approach connection update brain activation
hopfield pulse firing learned synapses stable states network
brain color visible maps fig loop elements constrained
speech connections capacity charge type matching depth signal

(d) 5 LSI topics at level 2 after alignment.

Top 8 Terms
recurrent direct events pages oscillator user hmm oscillators
false chain protein region mouse human proteins roc

(e) 2 LDA topics at level 3 after alignment.

Top 8 Terms
recurrent belief hmm filter user head obs routing
chain mouse region human receptor domains proteins heavy

(f) 2 LSI topics at level 3 after alignment.

Figure 5.8

correspondences. This identified a 3 level hierarchy of mapping functions. The number of basis functions spanning each level was: 800, 91, and 2. These numbers correspond to the structure of the latent space at each scale. At the finest scale, the space is spanned by 800 vectors because the joint manifold is spanned by 400 LSI topics + 400 LDA topics. At level 2, the joint manifold is spanned by 91 vectors, which we now examine more closely. Looking at how the original topics were changed can help us better understand the alignment algorithm. In Figures 5.8c and 5.8d, we show 5 corresponding topics (corresponding columns of $S_1\alpha_2$ and $S_2\beta_2$) at level 2. From these figures, we can see that the new topics in correspondence are very similar to each other across the datasets, and interestingly the new aligned topics are semantically meaningful—they represent some areas in either machine learning or neuroscience. At level 3, there are only two aligned topics (Figure 5.8e and 5.8f). Clearly, one of them is about machine learning and another is about neuroscience, which are the most abstract topics of the papers submitted to the NIPS conference. From these results, we can see that our algorithm can automatically align the given data sets at different scales following the intrinsic structure of the datasets. Also, the multiscale alignment algorithm was useful for finding the common topics shared by the given collections, and thus it is useful for finding more robust topic spaces.

5.5 Summary

Manifold alignment is useful in applications where the utility of a dataset depends only on the relative geodesic distances between its instances, which lie on some manifold. In these cases, embedding the instances in a space of the same dimensionality as the original manifold while preserving the geodesic similarity maintains the utility of the dataset. Alignment of multiple such datasets allows for simple a simple framework for transfer learning between the datasets.

The fundamental idea of manifold alignment is to view all datasets of interest as lying on the same manifold. To capture this idea mathematically, the alignment algorithm concatenates the graph Laplacians of each dataset, forming a joint Laplacian. A within-dataset similarity function gives all of the edge weights of this joint Laplacian between the instances within each dataset, and correspondence information fills in the edge weights between the instances in separate datasets. The manifold alignment algorithm then embeds this joint Laplacian in a new latent space.

A corollary of this algorithm is that any embedding technique that depends on the similarities (or distances) between data instances can also find a unifying representation of disparate datasets. To perform this extension, the embedding algorithm must use both the regular similarities within each datasets and must treat correspondence information as an additional set of similarities for instances from different datasets, thus viewing multiple datasets as all belonging to one joint dataset. Running the embedding algorithm on this joint dataset results in a unified set of features for the initially disparate datasets.

In practice, the difficulties of manifold alignment are identifying whether the datasets are actually sampled from a single underlying manifold, defining a similarity function that captures the appropriate structures of the datasets, inferring any reliable correspondence information, and finding the true dimensionality of this underlying manifold. Nevertheless,

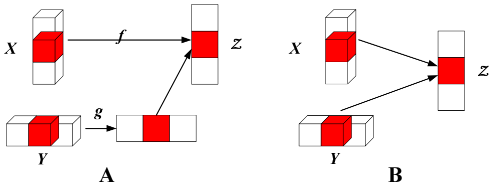


Figure 5.9: Two types of manifold alignment (this figure only shows two manifolds, but the same idea also applies to multiple manifold alignment). X and Y are both sampled from the manifold $\mathcal{Z}$, which the latent space estimates. The red regions represent the subsets that are in correspondence. f and g are functions to compute lower dimensional embedding of X and Y. Type **A** is two-step alignment, which includes diffusion map-based alignment and procrustes alignment; Type **B** is one-step alignment, which includes semi-supervised alignment, manifold projections, semi-definite alignment.

once an appropriate representation and an effective similarity metric are available, manifold alignment is optimal with respect to its loss function and efficient, requiring only the order of complexity of an eigenvalue decomposition.

5.6 Bibliographical and Historical Remarks

The problem of alignment occurs in a variety of fields. Often the alignment methods used in these fields are specialized for particular applications. Some notable field-specific problems are image alignment, protein sequence alignment, and protein structure alignment. Researchers also study the more general problem of alignment under the name information fusion or data fusion. Canonical correlation analysis [22], which has many of its own extensions, is a well-known method for alignment from the statistics community.

Manifold alignment is essentially a graph-based algorithm, but there is also a vast literature on graph-based methods for alignment that are unrelated manifold learning. The graph theoretic formulation of alignment is typically called graph matching, graph isomorphism, or approximate graph isomorphism.

This section focuses on the smaller but still substantial body of literature on methods for manifold alignment. There are two general types of manifold alignment algorithm. The first type (illustrated in Figure 5.9(A)) includes diffusion map-based alignment [23] and Procrustes alignment [10]. These approaches first map the original datasets to low dimensional spaces reflecting their intrinsic geometries using a standard manifold learning algorithm for dimensionality reduction (linear like LPP [24] or nonlinear like Laplacian eigenmaps [14]). After this initial embedding, the algorithms rotate or scale one of the embedded datatsets to achieve alignment with the other dataset. In this type of alignment, the computation of the initial embedding is unrelated to the actual alignment, so the algorithms do not guarantee that corresponding instances will be close to one another in the final alignment. Even if the second step includes some consideration of correspondence information, the embeddings are independent of this new constraint, so they may not be suited for optimal alignment of corresponding instances.

The second type of manifold alignment algorithm (illustrated in Figure 5.9(B)) includes semi-supervised alignment [9], manifold projections [25] and semi-definite alignment [26]. Semi-supervised alignment first creates a joint manifold representing the union of the given manifolds then maps that 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 that directly builds connections between features rather than instances and can naturally handle new test instances. The manifold alignment algorithm discussed in this chapter is a one-step approach.

References

[1] Sridhar Mahadevan. *Representation Discovery using Harmonic Analysis.* Morgan and Claypool Publishers, 1 edition, 2008.

[2] Ronald R Coifman, Stephane Lafon, Ann Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data. part i: Diffusion maps. *Proc. of Nat. Acad. Sci.*, 102(21):7426–7431, May 2005.

[3] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[4] M Belkin and P Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 6(15):1373–1396, June 2003.

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

[6] X. He and P. Niyogi. Locality preserving projections. In *Proceedings of the Advances in Neural Information Processing Systems*, 2003.

[7] Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:988–995, 2004.

[8] T. Jolliffe. *Principal Components Analysis.* Springer-Verlag, 1986.

[9] J. Ham, D. Lee, and L. Saul. Semisupervised alignment of manifolds. In *Proceddings of the International Workshop on Artificial Intelligence and Statistics*, pages 120–127, 2005.

[10] C. Wang and S. Mahadevan. Manifold alignment using procrustes analysis. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[11] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.

[12] J. Tenenbaum, Vin de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[13] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2000.

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

[15] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[16] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, July 2006.

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

[18] L. Hogben. *Handbook of linear algebra*. Chapman/Hall CRC Press, 2006.

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

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

[21] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[22] H. Hotelling. Relations between two sets of variates. *Biometrika*, 1936.

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

[24] X. He and P. Niyogi. Locality preserving projections. In *Proceedings of the Advances in Neural Information Processing Systems*, 2003.

[25] C. Wang and S. Mahadevan. Manifold alignment without correspondence. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

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