GRAPH CONSTRUCTION FOR MANIFOLD DISCOVERY

A Dissertation Presented

by

CJ CAREY

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2017

College of Information and Computer Sciences

GRAPH CONSTRUCTION FOR MANIFOLD DISCOVERY

A Dissertation Presented

by

CJ CAREY

Approved as to style and content by:

DEDICATION

To Angela:

my motivator,

my sanity checker,

my unfailing supporter,

my best friend,

my wife.

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor, Sridhar Mahadevan. Sridhar has always encouraged me to follow the problems I find interesting, and his mentorship has been essential to my growth as a researcher. I'd also like to thank my thesis committee members, Don Towsley, Ben Marlin, and Darby Dyar. In particular I want to thank Darby for introducing me to spectroscopy, geology, and planetary science, and for her constant encouragement and support.

Thank you to the faculty and staff of the College of Information and Computer Sciences, who provided a welcoming and productive environment for my graduate studies. Special thanks are due to Susan Overstreet and Leeanne Leclerc, without whose support and patience I would be lost.

Thanks also to the current and former members of the Autonomous Learning Lab, an excellent group of graduate students that I'm proud to have as colleagues and friends.

Finally, I want to thank my family for their unfailing encouragement and support: my wife Angela, who makes every challenge achievable; my sister Lisa, who has always inspired me to live up to her example; and my parents Clif and Kelli, whose nurture and love have permeated my life in more ways than I can acknowledge in words.

ABSTRACT

GRAPH CONSTRUCTION FOR MANIFOLD DISCOVERY

MAY 2017

CJ CAREY

B.Sc., WASHINGTON UNIVERSITY IN ST. LOUIS

M.Sc., WASHINGTON UNIVERSITY IN ST. LOUIS

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Sridhar Mahadevan

Manifold learning is a class of machine learning methods that exploits the observation that high-dimensional data tend to lie on a smooth lower-dimensional manifold.

Manifold discovery is the essential first component of manifold learning methods, in which the manifold structure is inferred from available data. This task is typically posed as a *graph construction* problem: selecting a set of vertices and edges that most closely approximates the true underlying manifold. The quality of this learned graph is critical to the overall accuracy of the manifold learning method. Thus, it is essential to develop accurate, efficient, and reliable algorithms for constructing manifold approximation graphs.

To aid in this investigation of graph construction methods, we propose new methods for evaluating graph quality. These quality measures act as a proxy for ground-truth manifold approximation error and are applicable even when prior information about the dataset is limited. We then develop an incremental update scheme for some quality measures, demonstrating their usefulness for efficient parameter tuning.

We then propose two novel methods for graph construction, the Manifold Spanning Graph and the Mutual Neighbors Graph algorithms. Each method leverages assumptions about the structure of both the input data and the subsequent manifold learning task. The algorithms are experimentally validated against state of the art graph construction techniques on a multi-disciplinary set of application domains, including image classification, directional audio prediction, and spectroscopic analysis.

The final contribution of the thesis is a method for aligning sequential datasets while still respecting each set's internal manifold structure. The use of high quality manifold approximation graphs enables accurate alignments with few ground-truth correspondences.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

As machine learning practitioners make use of increasingly large and complex data sets, better data representations are necessary to improve task performance while reducing computational burden. Manifold learning techniques, which aim to uncover the underlying geometry of a dataset, have proven to be especially useful in combating the "curse of dimensionality" [62]. For example, manifold methods have been used to make high-dimensional problems like face recognition [40] and robot control [72] more tractable. Manifold learning is often used for visualizing complex datasets [63], and has been applied to transfer learning tasks as well [94].

All of these methods operate under the *manifold hypothesis*, which assumes that input data lie on a lower-dimensional manifold [14, 68]. In this context, a manifold is a topological space that is smooth [55], compact, and locally Euclidean [76]. These properties allow any manifold to be viewed as a set of overlapping *charts*, each of which is a linear subspace [11]. The collective dimensionality of these charts is the *intrinsic dimension* of the manifold [85], which is typically much smaller than the dimensionality of the observed data. Figure 1.1 illustrates the concept of a manifold.

Figure 1.2 shows a simple example of points sampled from a two-dimensional manifold, embedded in a three-dimensional space. The higher-dimensional space is also known as *observation space*, in contrast to the underlying manifold space that structures the data. Performing traditional machine learning methods in manifold coordinates rather than the observed space reduces computational cost and improves model generalizability.

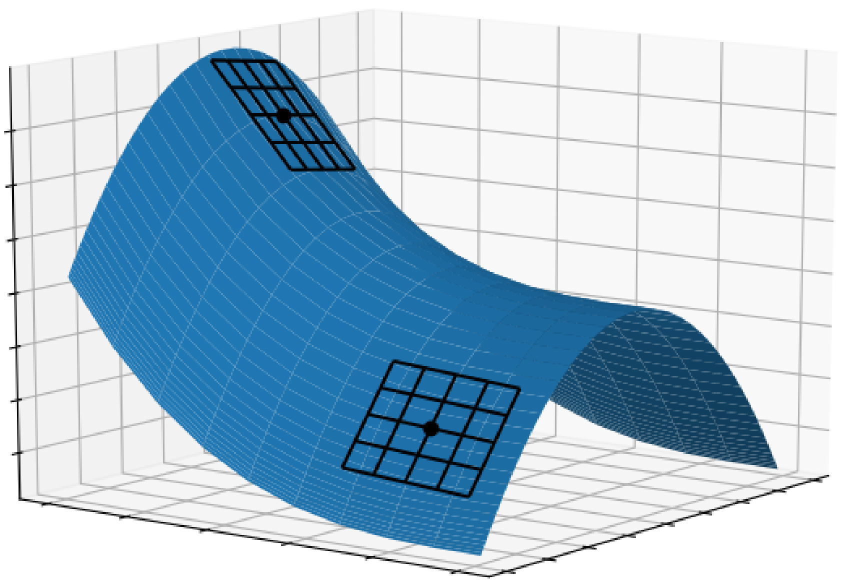
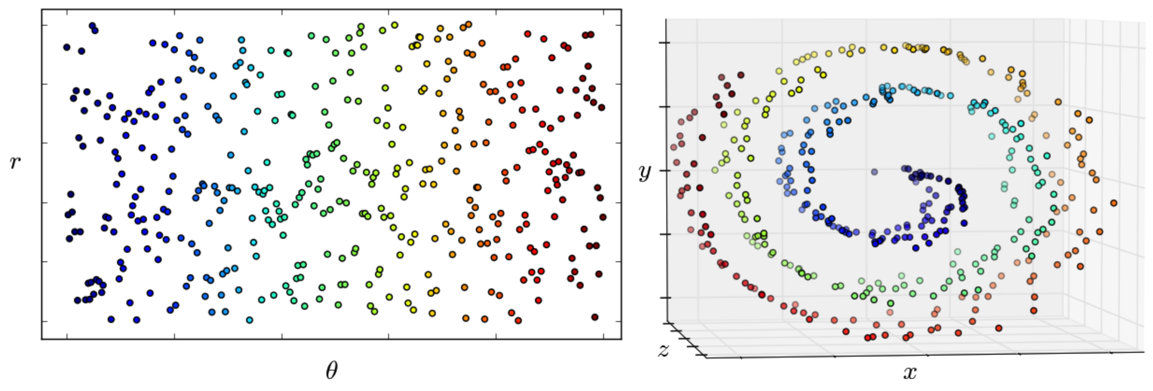


Figure 1.1: Conceptual illustration of a manifold. The global geometry (blue surface) may be arbitrarily nonlinear in the observed coordinate space, but the local vicinity of any point on the manifold (black grids) is approximately linear. The dimensionality of these locally linear subspaces is called the intrinsic dimension of the manifold.

1.2 Challenges

In many real-life applications, the underlying manifold structure of a given dataset is unknown. In these cases we must first apply *manifold discovery* to learn the structure of one or more manifolds that best explain the observed data. Manifolds are continuous objects that are often difficult to define analytically, so in practice it is common to manipulate a discrete representation in the form of a graph [66]. Vertices of the graph are sampled from the input data, and edges are chosen by connecting vertices for which the straight line between them does not leave the manifold. In this representation, each chart is defined by a vertex and the vertices with which it shares edges.

The manifold discovery problem can thus be reduced to the problem of *graph construction*: given a set of vertices, add edges such that the resulting graph best approximates the



(a) Manifold coordinate space. (b) Observed coordinate space.

Figure 1.2: Illustration of the "swiss roll" manifold. Points are generated by sampling uniformly at random from the (r, θ) manifold space, shown at left. Projecting those points to (x, y, z) coordinates as $(\theta \sin \theta, \theta \cos \theta, r)$ produces the observed data, shown at right. In both plots, points are colored by their θ value. A direct line between points in observed space risks crossing between loops of the spiral, whereas Euclidean distances in the manifold space are guaranteed to follow the structure of the roll.

true structure of the manifold. This task is made tractable by exploiting the smoothness, compactness, and local linearity properties of manifolds.

1.3 The Manifold Learning Process

Historically, the label "manifold learning" has been applied to methods that operate on known manifolds as well as to manifold discovery algorithms. In this thesis, we define the *manifold learning process* as follows:

1. Given a data set X, discover the underlying manifold M.
2. Using M and optional auxiliary information Z, perform the desired learning task.
3. Optionally, define a mapping for new data X' onto M to generalize the model to new instances.

These steps, illustrated in Figure 1.3, provide a framework for understanding both the existing work in manifold learning and the novel contributions of this thesis.

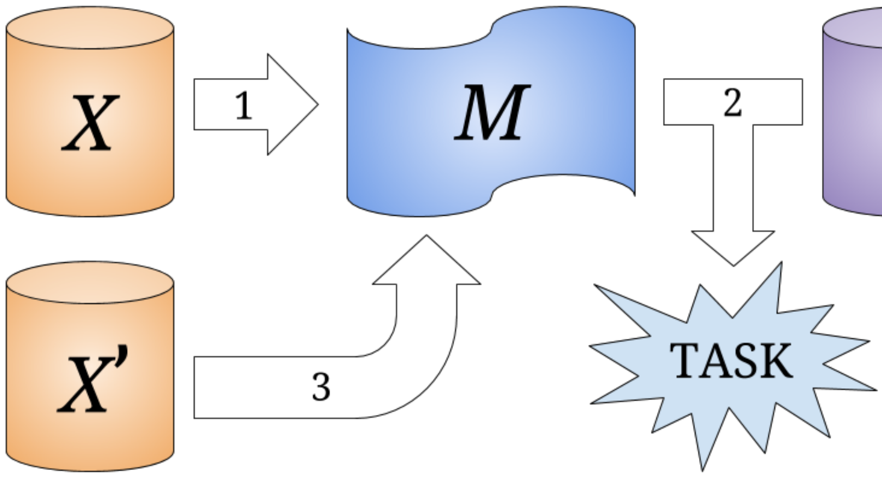


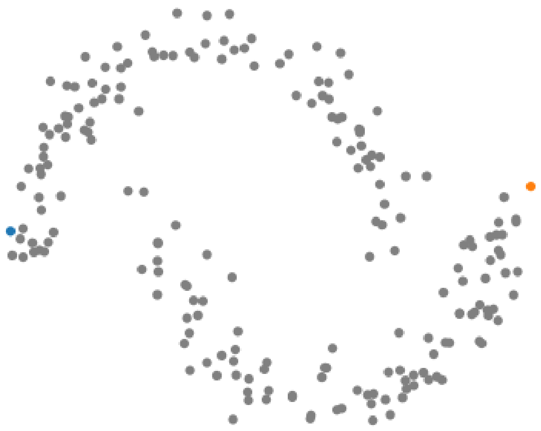
Figure 1.3: The manifold learning process. Step 1 discovers manifold M using observed data X. Step 2 uses M along with auxiliary information Z (i.e., class labels, regression targets) to perform a learning task. Step 3 learns a mapping from new samples X' (drawn from the same distribution as X) to the manifold.

Figure 1.4 provides an example of how the manifold learning process is applied on a simple classification task. In this case, two manifolds with a single intrinsic dimension each are discovered, then used to provide labels for each element of the dataset.

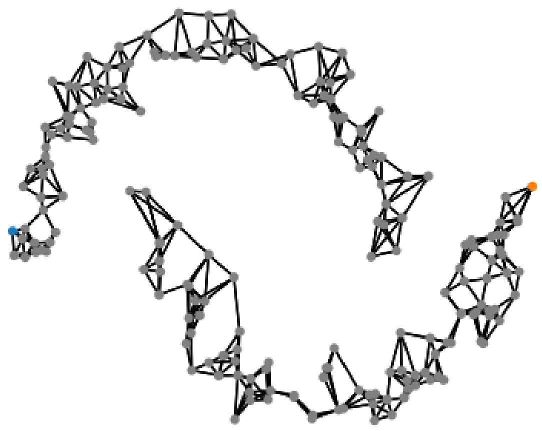
Figure 1.5 shows the same example with a different graph approximating the manifolds. This example demonstrates that the first step (manifold discovery) is critical to the success of the process as a whole, because errors in the graph estimating the manifold can propagate to errors in the second step's learning task.

1.4 Contributions

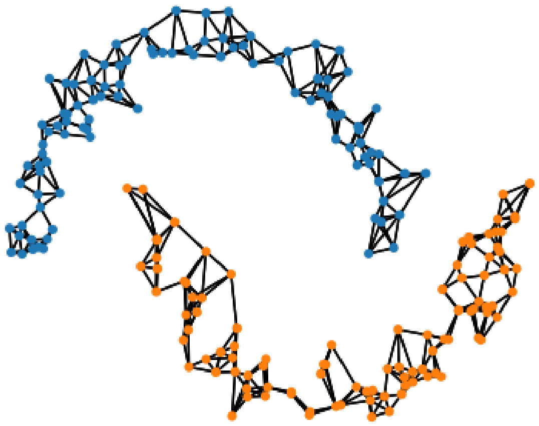
The **first contribution** of this thesis is a study of the properties of graphs that act as indicators for high quality manifold representation (Chapter 3). These quality measures allow evaluation of graphs without ground-truth information about the manifold, and do not rely on results from step two of the manifold learning process. We show that these measures are both useful and efficient, developing an incremental update algorithm for use in parameter tuning.



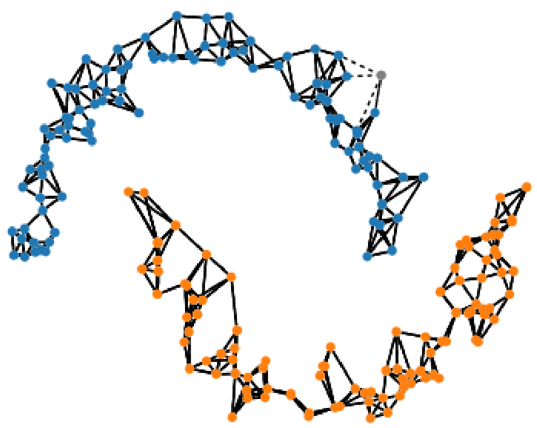
(a) The "two moons" dataset in observed coordinates, with one labeled example per moon (in orange and blue) and 198 unlabeled examples (in gray). The task is to color each gray point either orange or blue, and thus separate the two halves of the dataset. Using notation from the manifold learning process, X is the set of coordinates for all points, Z is the label information, and M is the learned graph.



(b) Step 1: Manifold discovery via graph construction. Each point is connected to its four nearest neighbors ranked by Euclidean distance, creating a graph with two disjoint components.

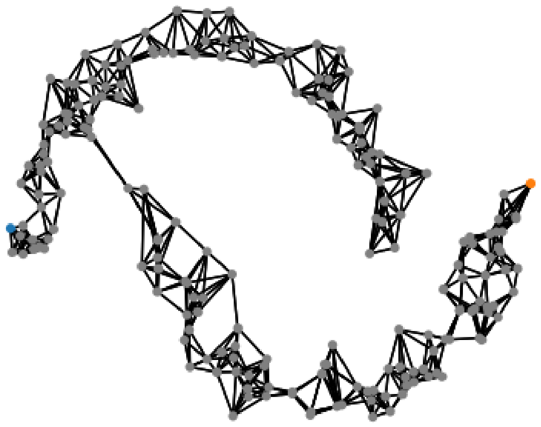


(c) Step 2: Manifold learning using M, the graph constructed in step 1, and Z, the two known labels. Using a label propagation algorithm, each labeled point assigns its color to all of its unlabeled neighboring points. This process is repeated until every point is labeled, correctly separating the two moons.



(d) Step 3: Generalizing to new data. When a new point is added to the dataset (in gray), it takes its label from the labels of its four nearest neighbors in the original dataset. This point will be colored blue, as all of its neighbors (dotted lines) are blue.

Figure 1.4: The manifold learning process applied to the task of labeling instances of the "two moons" dataset. Many options are available for each step of the process; for clarity of illustration, we have selected simple methods that visualize well.


(a) Step 1 again, with too many nearest neighbor connections. Instead of discovering two disconnected manifolds, the learned graph joins the moons together.

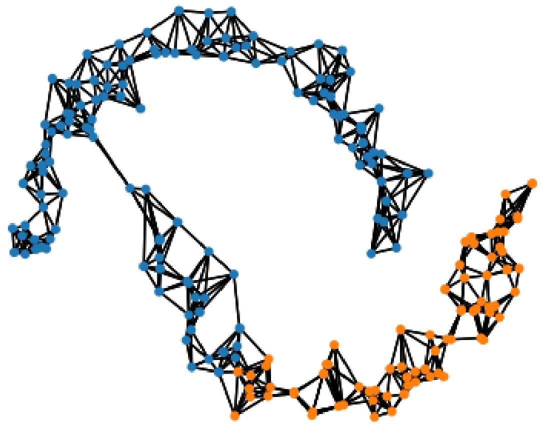

(b) Step 2 using the poor manifold approximation. The label information from the top moon leaks into the bottom moon using the erroneous connecting edge.

Figure 1.5: Failure of the manifold learning process due to poor manifold discovery. Using the same methodology as Figure 1.4, but replacing the 4-nearest neighbor graph with 6 nearest neighbors, we observe the sensitivity of the following steps in the manifold learning pipeline to the learned graph.

The **second contribution** is a novel method for graph construction, the Manifold Spanning Graph (Chapter 4). This algorithm avoids the difficult hyperparameter tuning process by iteratively joining local subgraphs in a principled way.

The **third contribution** of this thesis is the Mutual Neighbors Graph algorithm (Chapter 5), which frames graph construction in a semi-supervised setting. Given a sparsely-connected graph that lies on the manifold, this algorithm adds edges to produce a useful manifold approximation without deviating from the underlying structure.

These two algorithms are experimentally validated against state of the art graph construction techniques on a multi-disciplinary set of application domains, including image classification, directional audio prediction, and spectroscopic analysis.

The **final contribution** of the thesis is Manifold Warping (Chapter 6), a method for aligning sequential datasets while respecting each set's internal manifold structure. The use of high quality manifold approximation graphs enables accurate alignments with few ground-truth correspondences, improving alignment accuracy compared to extant methods without manifold considerations.

Thesis Statement

> Graph construction algorithms that respect structural properties of manifolds can significantly improve the performance of manifold learning applications.

CHAPTER 2

RELATED WORK

In this chapter, we review existing manifold learning and discovery methods using the notation from the manifold learning process defined in the introduction. In Section 2.1, we discuss methods for machine learning on manifolds. In Section 2.2, we discuss graph construction algorithms for several categories of manifold discovery problems.

2.1 Manifold Learning

2.1.1 Overview

Classical machine learning methods tend to scale poorly when the number of dimensions of input data increases; a phenonemon known as the "curse of dimensionality" [62]. In addition to increased time and memory requirements, many classical methods struggle to accurately capture the complexity of high-dimensional datasets with nonlinear dynamics.

Manifold learning is a class of methods that make use of the observation that these complex datasets often have well-behaved internal structure: a lower-dimensional, locally Euclidean manifold. Given a representation of this internal structure, manifold learning methods are able to efficiently and effectively operate on high-dimensional datasets. These methods are used in step two of the manifold learning process outlined in Section 1.3:

> Using M and optional auxiliary information Z, perform the desired learning task.

This section reviews five groups of existing manifold learning methods, organized by the kind of learning task they perform. In each case, we assume that the manifold M is given (typically in the form of a graph), along with various forms of additional information Z.

2.1.2 Dimensionality Reduction

Dimensionality reduction is the task of finding a mapping for each point in a given dataset to a new, lower dimensional space. For an input dataset $X \in \mathbb{R}^{n \times d_o}$, with n points in d_o dimensions, the mapping produces new coordinates $Y \in \mathbb{R}^{n \times d_e}$, where the new dimensionality $d_e < d_o$. The key constraint on this transformation is one of structure preservation; Y must preserve one or more key characteristics of X, allowing Y to be used in place of X for further analysis.

Early work in this area drew on statistical measures of information content to find linear embeddings that preserve "useful" properties of the input data. These methods include Principal Component Analysis (PCA) [42] and multidimensional scaling (MDS) [52]. PCA was designed to preserve the variance of the original data, solving for a linear transformation of X to an orthogonal basis Y in which basis vectors are ordered by the amount of variability they explain. Formally, PCA is $Y = XW$ where W a column matrix of the eigenvectors of $X^\top X$ corresponding to the d_e largest eigenvalues.

In the early 2000s, new manifold learning techniques produced state of the art results on dimensionality reduction tasks, helping to popularize the study of manifold-based methods [85, 75, 4]. These *manifold embedding* algorithms constrain Y to preserve various properties of the manifold M, enabling nonlinear transformations of the original data X. This is typically formulated as an unsupervised learning task, so no auxiliary data are required (e.g., $Z = \emptyset$).

Algorithms for manifold embedding can be clustered into one of three families:

- **Geodesic family**: The *geodesic distance* between two points on a manifold M is the length of the shortest path between them that does not leave the manifold's surface. This concept is illustrated in Figure 2.1. Geodesic embedding methods estimate pairwise geodesic distances between all points in X, then construct Y to minimize the disparity between the geodesic distances and pairwise Euclidean distances between rows of Y. This constraint ensures that the global connectivity structure of M is preserved, trading off some amount of distortion in local regions. Methods in this family include Self-Organizing Maps [49], Curvilinear Component Analysis [21], and Isomap [85].

- **Laplacian family**: The *Laplace-Beltrami operator* is defined as the divergence of the gradient of a function, and is useful in measuring the deformation induced by mapping the manifold to a coordinate space [4]. Manifolds are often approximated using a graph representation, necessitating the use of the *graph Laplacian* (L), a discrete approximation of the Laplace-Beltrami operator. Laplacian embedding methods minimize the squared gradient of the coordinate mapping function, which is equivalent to solving for the d_e eigenvectors of L corresponding to the smallest nonzero eigenvalues. Methods in this family include Laplacian Eigenmaps [4] and Diffusion Maps [16].

- **Locally Linear family**: The local linearity property of manifolds implies that any individual point in X can be reconstructed as a linear combination of its neighboring points, as given by M. Each point can then be represented in terms of its neighbors using a matrix of linear reconstruction weights, W. It is then possible to solve for Y by minimizing the reconstruction error $\|Y - WY\|_F$. Methods in this family include Locally Linear Embedding [75], Local Tangent Space Alignment [109], Hessian Locally Linear Embedding [26], Sparse Manifold Clustering and Embedding [28], and Low Rank Embedding [59].

The loss functions in these methods are commonly formulated as generalized eigendecomposition problems, which require $O\left(|X|^3\right)$ time in the general case. This is typically the dominating factor in the algorithm's running time, though extensions have been proposed to reduce this computational burden [81, 61].

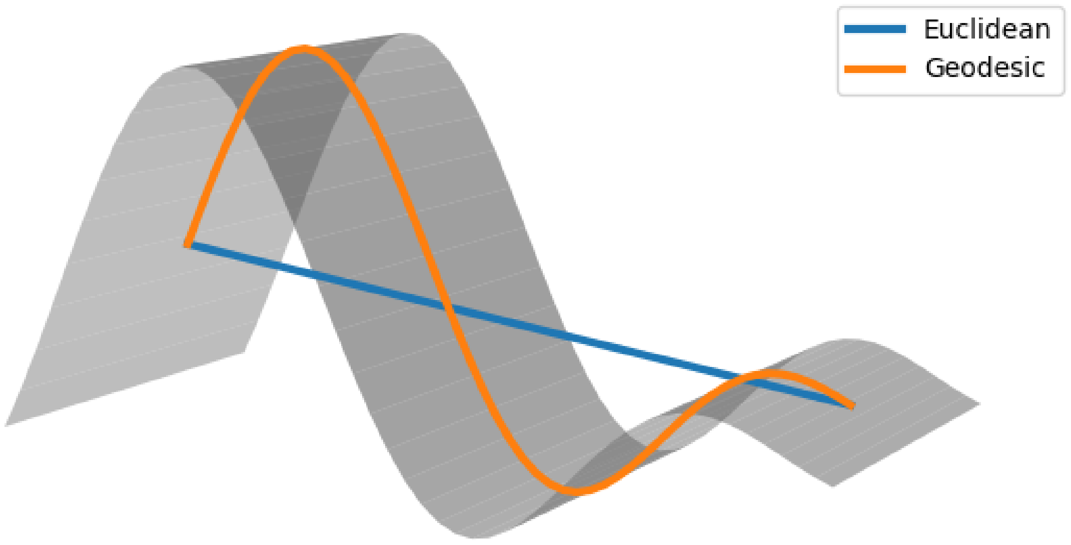


Figure 2.1: Illustration of geodesic distance. Euclidean distance in the observed coordinate space underestimates the true geodesic distance between points sampled from a manifold, representing the length of the shortest path connecting the points that does not leave the manifold. The goal of a geodesic manifold embedding algorithm is to project the manifold to a space in which Euclidean and geodesic distances are equivalent.

2.1.3 Semisupervised learning

Where supervised learning tasks have label information for each instance in X, and unsupervised tasks have no labels at all, semisupervised learning refers to learning tasks with partial label information. These tasks are often formulated as classification or clustering problems, using a set of labeled instances L_{known} to extrapolate labels for the unknown instances, $L_{unknown}$. Rather than ignoring the rows of X without label information, using the unlabeled instances allows semisupervised learning algorithms to more accurately generalize and avoid overfitting.

In the context of the manifold learning process, $Z = L_{known}$ and M is used to constrain the extrapolation of labels, forming a family of *manifold regularization* methods [5]. These methods include *label propagation* techniques, in which labeled instances influence the labels of their unlabeled neighbors in manifold space [113, 110]. More sophisticated algorithms, such as Linear Neighborhood Propagation [95] and Laplacian Assignment [13], are also built on the same core idea.

2.1.4 Value Function Approximation for Reinforcement learning

Reinforcement learning (RL) is a technique for solving Markov decision processes, in which an agent observes its environment, performs actions, and receives reward signals [83]. A key subproblem of reinforcement learning is the task of learning a good approximation of the *value function*, which maps states and actions to a real-valued estimate of expected future rewards.

The joint state-action space is typically continuous, which has led RL practitioners to use approximations consisting of a fixed set of nonlinear basis functions, such as radial or Fourier bases [50]. As the dimensionality of the state-action space increases, however, the domain of the value function increases and the curse of dimensionality makes value function estimation more difficult.

Similar to dimensionality reduction efforts, the manifold underlying the state-action space can be used to find a low-dimensional representation in which value function estimation is more tractable. Unlike the previously described settings, however, the full set of instances X is not available at the outset in reinforcement learning. Instead, X is generated by the agent performing actions in the state space directed by a sequential decision process. This setting can provide additional clues for discovering M, using Z to record information about observed reward signals and state-action pairs. Examples of manifold learning methods applied to this RL setting include Proto-Value Functions [64] and its directed variants [45].

2.1.5 Transfer learning

Transfer learning is the collective term for tasks in which information from one domain is used to bootstrap learning in another domain. This cross-domain learning strategy is attractive because it can speed up the learning process for new problems dramatically, while providing additional structure to underspecified problems [73]. For example, the *self-taught learning* framework uses a large unsupervised dataset to learn structure, then performs a specific learning task using a smaller labeled dataset [74].

One task in this category that has benefited substantially from manifold-based approaches is dataset alignment, a semisupervised learning problem in which correspondences are learned between multiple related data sets. Manifold alignment is a class of techniques that solves this alignment problem when the input data sets share a common underlying structure by finding a shared latent space in which the disparate input datasets can be compared directly.

Wang's original Manifold Alignment [94] was introduced as a semi-supervised, nonlinear extension of Canonical Correlation Analysis (CCA) [1], that aims to preserve both local geometry and inter-dataset correspondences. Many methods for manifold-based transfer learning derive from this framework, including Semisupervised Manifold Alignment [39], Manifold Alignment Preserving Global Geometry [93], Manifold Warping [91], and Low Rank Alignment [9].

2.1.6 Metric learning

The final category of machine learning tasks we consider is metric learning, which considers the problem of defining a function that compares pairs of instances in a semantically relevant way [104]. Specifically, we learn the function $d\left(x_i, x_j\right) \geq 0$ where $x \in X$, such that the distance d is smaller when two instances are more similar for some problem-specific understanding of similarity.

Much of the work on metric learning has followed the framework of generalized Mahalanobis distance, which defines a specific distance

$$d_A\left(x_i, x_j\right) = \sqrt{\left(x_i - x_j\right)^\top A \left(x_i - x_j\right)}$$

parameterized by a square matrix A. When $A = I$, this reduces to Euclidean (L_2) distance. Traditional Mahalanobis distance uses $A = \text{cov}(X)^{-1}$, and several methods have been proposed for finding an optimal A given semi-supervised label information, including Large Margin Nearest Neighbors [99] and Information Theoretic Metric Learning [17].

Manifold learning can be viewed as a kind of metric learning because M can be used to estimate geodesic distances along the manifold. Standard distance metrics accurately capture local effects due to the local linearity property, but consistently underestimate the true distance between points sampled from different regions of the manifold. Unlike generalized Mahalanobis distances, manifold distance metrics are generally nonlinear functions. The dimensionality reduction methods reviewed above may be viewed as manifold-based metric learning methods, but an explicit coordinate embedding is not required to learn a metric.

2.2 Manifold Discovery via Graph Construction

Recall step one of the manifold learning process:

Given a data set X, discover the underlying manifold M.

This step contains two challenges: inferring a manifold that best explains the observed data, and representing that manifold in a useful way for the subsequent steps of the process. Manifolds are continuous objects and the observed data are finite, so these learned representations are often discretized, approximating a smooth manifold with the local, piecewise-linear edges of a graph.

This section introduces graph basics, then discusses representations for M with a focus on manifold approximation graphs. Then we review existing methods for learning these graphs, a problem known as *graph construction*.

2.2.1 Graphs and Their Properties

A *graph* is a tuple containing a vertex set and an edge set,

$$G = (V, E),$$

where each edge $e_{ij} \in E$ connects vertices $v_i \in V$ to $v_j \in V$ [7].

A graph is *undirected* when the existence of any edge e_{ij} implies an identical edge in the reverse direction, e_{ji}. Undirected graphs can thus be treated as a special case of directed graphs.

A graph is *weighted* when every edge also has an associated real-valued, nonzero, scalar weight, w_{ij}. Edge weights are commonly used to encode the relative importance of connections between vertices. If a graph is unweighted, edge weights are implied as $w_{ij} = 1 \forall e_{ij} \in E$.

The *unweighted degree* of a vertex is defined

$$d(i) = \sum_{j \neq i \in V} [e_{ij} \in E]$$

and represents the number of outgoing edges from the vertex. The *weighted degree* of a vertex is similarly defined,

$$d_w(i) = \sum_{j \neq i \in V} w_{ij},$$

and for unweighted graphs the two are equivalent.

A graph *geodesic* is the shortest path between two vertices along edges in E. The length of this path is the *geodesic distance*, defined as the sum of the weights of the edges making up the path. An unweighted graph geodesic is equivalent to the weighted case where all edges have unit weight, and thus unweighted geodesic distance is the number of edges comprising an unweighted shortest path.

In practical applications of manifold learning, it is often useful to represent graphs using an *adjacency matrix*:

$$W_{ij} = \begin{cases} w_{ij} & \text{if } e_ij \in E \\ 0 & \text{otherwise} \end{cases}$$

This representation has the benefit of encoding edge connectivity and weighting simultaneously.

2.2.2 Manifold Approximation Without Explicit Graphs

Most of the methods presented in Section 2.1 operate on graphs in adjacency matrix representation. These graphs are assumed to be sparse such that $|E| \propto |V|$, resulting in a sparse adjacency matrix W with few nonzero entries.

Other methods use a dense pairwise similarity matrix in place of W, rather than finding an explicit set of edges. This can be viewed as the adjacency matrix of a complete graph, with $|E| \propto |V|^2$. These implicit-graph methods include Low Rank Embeddings [59], t-SNE [89], Affinity Propagation [32], and Spectral Clustering [80]. In these cases, the task of graph construction may be viewed as a complete graph weighting problem, and many of the graph construction methods presented below can be applied with minimal modification.

Some older manifold approximation methods have used non-graph representations [49, 8], but these often fail to represent complex manifold structures accurately due to their restrictive global models [86].

Several algorithms have attempted to side-step the graph construction problem by explicitly representing the manifold as a combination of piecewise linear components, solving for a set of subspaces that lie tangent to the manifold surface. This family of methods includes charting [11] and several local PCA algorithms [47, 88, 41]. This concept has also proven useful in the Local Tangent Space Alignment embedding algorithm [108], which, given an existing graph representation, computes linear subspaces from local neighborhoods to recover embeddings with less distortion. These approaches only provide coarse manifold approximations, however, and require specialized algorithms for the varied applications that graph-based representations enable naturally, such as out-of-sample extension and cross-manifold alignment.

2.2.3 Unsupervised (Classical) Graph Construction

Traditional approaches to graph construction have treated the problem as an unsupervised learning task, relying on varied assumptions about the input data and the manifold(s) they lie on. These assumptions enable enhanced performance at the cost of flexibility, and are a useful way to categorize families of existing methods.

The simplest assumption is that neighbors in original space are neighbors on the manifold. This is central to the "neighborhood graph" family of methods, which is described in more detail in Section 2.2.3.1. The Perturbed/Disjoint Minimum Spanning Tree algorithms assume that the minimum spanning tree of the input data lies on the manifold [105]. Other methods assume that edges in the original space should lie on locally-linear subspaces, including Nonlocal Manifold Tangent Learning [6]. Finally, some methods assume that the manifold is sampled non-uniformly, including Multi-Class Manifold Learning [100] and Robust Locally Linear Embedding [38].

Several unsupervised graph construction algorithms rely on the local linearity assumption, which implies that each vertex can be reconstructed as a linear combination of its neighbors. Minimizing the difference between reconstructed and actual vertex coordinates allows graph construction to be formulated as sparse optimization problem [102, 27, 58].

2.2.3.1 Neighborhood graphs

The most commonly used family of graph construction methods is based on a neighborhood criterion, $\mathcal{N}$. The neighborhood graph adjacency matrix is defined by

$$W_{ij} = \begin{cases} \delta\left(v_i, v_j\right) & \text{if } v_j \in \mathcal{N}\left(v_i\right) \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

given a distance metric δ.

The k-nearest neighbors algorithm is a popular instance of this type of graph construction, where $\mathcal{N}\left(v_i\right)$ is the set of k vertices with smallest distance from a given vertex v_i.

While computationally efficient and simple, this algorithm does not necessarily create a symmetric graph, because some of v_i's nearest neighbors are not guaranteed to include v_i among their own nearest neighbors. Symmetry is often assumed in graph-based learning algorithms, so a k-nearest adjacency matrix W is commonly symmetrized to form an undirected graph. The choice of symmetrization has been the subject of some study and has been shown to be an important factor for graph quality [19], but typical formulations include averaging,

$$W_{sym} = \frac{W + W^\top}{2} \tag{2.2}$$

and max-symmetrization,

$$W_{sym} = \max\left(W, W^\top\right). \tag{2.3}$$

Restricting the degree of every vertex to be exactly k also limits this approach, though several extensions have been proposed to allow neighborhood sizes to vary among vertices [103, 107, 67]. Even finer-tuned control is provided by *b-matching*, which uses belief propagation to find a neighbor graph that matches a given degree distribution [43].

The other classical neighbor-based graph construction method is the ϵ-close algorithm:

$$\mathcal{N}(v_i) = \{v_j \in V \mid \delta(v_i, v_j) \leq \epsilon\}$$

While this algorithm always constructs a symmetric graph, it also tends to produce too many edges, especially when the input data are poorly scaled or not uniformly distributed, because the maximum degree of vertices in the constructed graph has no upper bound.

2.2.4 Semisupervised Graph Construction

In many applications, some edges of the desired graph are already known and the remaining task is to find additional edges that complete the graph. These given edges may come from structural properties of input data (i.e., trajectories, lattices, and trees), expert-provided labels, or constraints specific to the subsequent learning task. This semi-supervised setting casts the edge assignment problem as a binary classification task in the unweighted case and a regression task in the weighted case.

Partial supervision is often derived from domain-specific constraints. For example, reinforcement learning and other sequential learning problems naturally generate data in the form of trajectories with explicit edges between time-adjacent states [45].

Side information can also be exploited by restricting the learning task to which the learned graph will be applied. For example, methods have been proposed to blend graph construction with dimensionality reduction by building a graph and computing its embedding simultaneously [103, 106]. These task-specific graph construction algorithms offer improved graph quality at the cost of limited generality.

2.2.5 Multiple Manifold Discovery

In the preceding sections of this chapter, we have assumed that only one underlying manifold exists that can explain the observed data concisely. In many practical applications, however, it can more accurate to assume that the data are generated by a *mixture* of manifolds. In this regime, each point is assigned to exactly one manifold, though the manifolds may overlap or intersect.

Multiple-manifold learning algorithms can be applied to a broader selection of settings, but the increased flexibility results in a more difficult manifold discovery process. As a result, some methods avoid explicitly building representations of the manifolds, instead relying on low-rank optimization methods to disambiguate manifold intersections [59, 9].

As with traditional single-manifold discovery, methods for multiple manifold discovery can be grouped by the assumptions they rely on, in decreasing order of required information:

- The intrinsic dimension of each manifold is known: k-Manifolds [82].
- The maximum intrinsic dimension is known: Local and Structural Consistency [97], mumCluster [96], and Spectral Multi-Manifold Clustering [98].

- Only the number of manifolds is known: k-Planes [10], Spectral Clustering [80, 71].

These methods label each input point according the manifold it lies on, but do not necessarily build representations for each manifold. This reduces the multi-manifold discovery problem to a set of traditional manifold discovery tasks, for which single-manifold graph construction methods may be applied to each manifold independently.

CHAPTER 3

GRAPH QUALITY METRICS FOR APPROXIMATING MANIFOLDS

3.1 Overview

In the manifold learning process outlined in Section 1.3, the goal is to produce the best performance on the chosen manifold learning task. This requires careful selection of algorithms and parameters for both manifold learning and manifold discovery.

Previous work has found that the methods used for manifold discovery are in many cases more critical than the manifold learning tasks used in the next step. In his 2005 PhD thesis on graph-based semi-supervised learning methods, Zhu found that starting with a good graph is often more important than the choice of learning algorithm [114]. Similarly, researchers have found that dimensionality reduction results are sensitive to the quality of the graph representation used [3, 103].

Therefore, when evaluating graph construction algorithms for manifold discovery tasks, we require some measure of how well a given graph approximates the true underlying manifold structure. This *graph quality* score can be used to set optimal hyperparameters for a graph construction algorithm, or to distinguish between competing algorithms.

Graph quality evaluations can be derived in several ways. This chapter explores three approaches:

1. Comparing against known ground truth manifold structure (Section 3.2).
2. Evaluating the performance of a fixed manifold learning application (Section 3.3).
3. Examining graph-theoretic indicators of adherence to assumptions about the manifold structure (Section 3.4).

The first contribution of this chapter is to introduce two *task-independent* measures of graph quality, based on the rate of change of graph statistics as edges are added (Section 3.5). The second contribution is a new method for incremental graph construction that updates edges, geodesics, and associated quality measures on-line (Section 3.6). When used in combination, these two contributions can accelerate and simplify the parameter tuning process for the graph construction component of the manifold learning process.

3.2 Ground Truth Comparison

When the ground truth manifold is known prior to the manifold learning process, direct methods are available for computing graph quality. These provide a convenient test bed for analyzing graph construction methods, though their applications are limited to synthetic datasets with explicitly defined structure. If the known manifold has special structure, customized measures of graph quality are simple to create. In the general case, however, we require a generic ground truth quality measure.

We propose a novel ground truth graph quality measure based on graph geodesics, which measures the agreement between the known pairwise distances on the manifold and pairwise distances of an embedding of the graph in a low-dimensional space:

$$\|D^{emb} - D^{GT}\|_F \tag{3.1}$$

where D^{emb} is the normalized pairwise distance matrix computed in the embedded space, D^{GT} is the normalized pairwise distance matrix of the ground truth manifold coordinates, and $\|\cdot\|_F$ is the Frobenius norm. We compute the embedding using the Isomap algorithm [85] with an embedding dimension equal to the dimension of the ground truth coordinates.

This approach prioritizes the preservation of geodesic distances, measuring how well the graph represents the true manifold in a global sense. Perturbations in local neighborhoods will produce small differences, but edges that fail to follow the manifold will induce

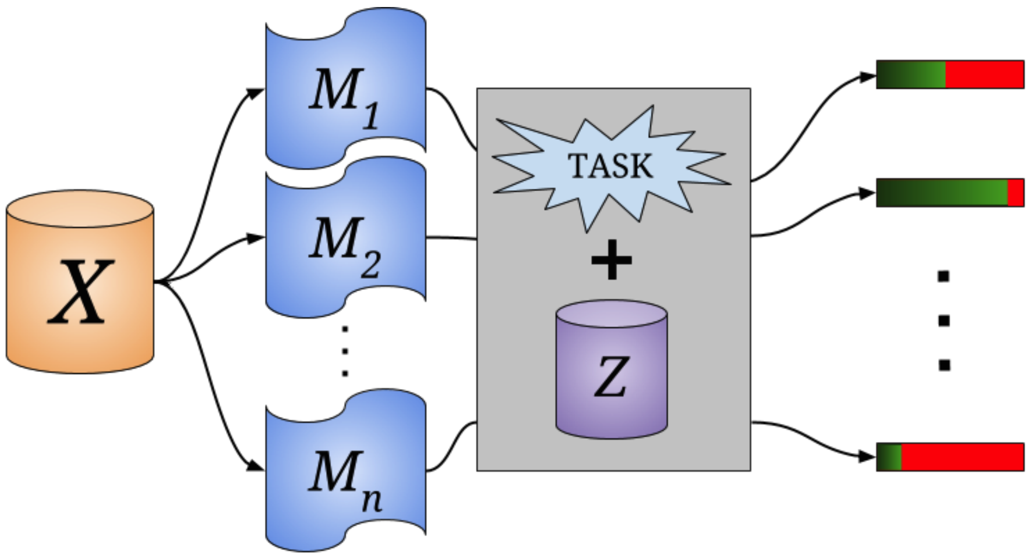


Figure 3.1: Task-specific graph quality evaluation procedure. The same dataset (X) is used to produce different manifold approximation graphs, using varying hyperparameters and/or algorithms. Each graph is used as input to a manifold learning task with all other inputs and parameters fixed, from which comparable scores can be obtained. These measures of graph quality are limited in scope, however, as they remain tied to the specific task and manifold learning algorithm chosen.

large changes to many geodesic paths and result in a large difference. Therefore, when this quality measure is minimized, we are confident that the graph edges represent the true manifold accurately.

3.3 Manifold Learning Task Evaluation

In real-world applications of manifold discovery methods, ground truth information about the underlying manifold is not available. In these settings, graph quality can be inferred by isolating the graph's contribution to the performance of a subsequent manifold learning task. The parameters of the manifold learning method are fixed, which allows different manifold approximation graphs to be compared directly. Figure 3.1 illustrates this process visually.

This technique is both straightforward and popular, with usage in embedding [3, 79], clustering [65], and semi-supervised learning [60, 19] contexts.

3.4 Task-Independent Graph Quality

The main weakness of task-specific graph quality measures is their indirectness. The quality of a graph is only inferred using task performance as a proxy, which introduces additional noise and significant computational cost to the evaluation process.

To address this issue, we propose novel *task-independent* measures of graph quality, based on graph-theoretic properties with strong correlations to the desired properties of manifolds.

3.4.1 Graph-theoretic Properties

Graph theory is a well-established field that has produced a variety of ways to formally characterize graphs, as introduced in Section 2.2.1. Several graph-theoretic properties are useful in measuring how well a graph adheres to basic tenets of manifold structure, including sparsity, connectivity, eccentricity, and centrality.

The *sparsity* of a graph is the ratio of actual edges to possible edges,

$$\frac{|E|}{|V|^2}. \tag{3.2}$$

This can also be expressed in terms of a graph's average unweighted degree, and relates to the true manifold's intrinsic dimension: as more edges connect vertices to each other, a higher-dimensional space is required to preserve local linearity [48].

A *connected component* is a set of vertices with at least one path between all pairs in the set [84]. All geodesics distances are finite within a connected component, and infinite between disconnected components. As manifolds are continuous spaces, it follows that a single manifold should be approximated by a single connected component.

The *eccentricity* of a vertex, $e(v)$, is the maximum length among all geodesics between v and any other vertex in the graph. The *diameter* of a graph is defined as

$$\max_{v \in V} e(v), \tag{3.3}$$

representing the longest geodesic between v and all other vertices. These summaries of a graph's geodesic distances are useful because they are directly analogous to distances in manifold space.

Betweenness centrality is a measure of how frequently a given vertex appears among all graph geodesics [31], defined as

$$C_B(v) = \sum_{v \in V \setminus \{s,t\}} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{3.4}$$

where σ_{st} is the number of geodesics between vertices s and t, and $\sigma_{st}(v)$ is the number of those $s - t$ shortest paths that include the vertex v [12]. Similar to how the diameter summarizes graph eccentricity, it is often useful to consider the maximum betweenness centrality,

$$\max_{v \in V} C_B(v). \tag{3.5}$$

Given vertices sampled uniformly from a smooth manifold, their betweenness centrality should also follow a uniform distribution, indicating that all vertices are equally likely to appear across all geodesics.

Each of the measures introduced above can provide insights about the degree to which a graph might correspond to underlying manifold structure, and most graph construction algorithms target at least one of these properties. For example, the k-nearest neighbors algorithm exerts direct control over the resulting graph's sparsity, but provides no guarantees about its other properties.

3.4.2 Efficient Computation of Betweenness Centrality

Brandes' 2001 algorithm [12] calculates betweenness centrality in $O\left(|V||E| + |V|^2 \log|V|\right)$ time and $O\left(|V| + |E|\right)$ space. Instead of using an all-pairs shortest-path algorithm, the following combinatorial shortest-path calculation is used,

$$\sigma_{sv} = \sum_{u \in P_{sv}} \sigma_{su}, \tag{3.6}$$

where P is a predecessor matrix where P_{sv} is the vertex preceding v on the shortest path from s to v. Dijkstra's algorithm [24] is used to count shortest paths. Vertex dependencies are accumulated recursively using the following relation:

$$\delta_{s\bullet}(v) = \sum_{w:v \in P_{sw}} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 = \delta_{s\bullet}(w)) \,. \tag{3.7}$$

Accumulating $\delta_{s\bullet}$ by traversing the points in order of non-increasing distance from s requires $O(|V||E|)$ time, so betweenness centrality becomes

$$C_B(v) = \sum_{v \in V \setminus \{s\}} \delta_{s\bullet}(v). \tag{3.8}$$

In the common case of weighted neighborhood graphs with unique pairwise distances, every pair of vertices is connected by a unique shortest path. This simplifies the betweenness centrality in Equation 3.8 to

$$C_B(v) = \sum_{v \in V \setminus \{s,t\}} \sigma_{st}(v), \tag{3.9}$$

where $\sigma_{st}(v)$ acts as an indicator function.

3.5 Differential Graph Quality

Each of the graph-theoretic properties described in Section 3.4 provides insight into the structure of a graph as it relates to manifold approximation, but none are directly usable as graph quality measures. To adapt them to this purpose, it is necessary to observe the rate of change of these properties as edges are added to a graph.

This novel approach derives from the simple observation that constructing manifold-approximation graphs can be viewed as an optimal stopping problem: Edges connecting the most similar instances are very likely to lie along the manifold, so by adding edges incrementally in the order of increasing length, it is only necessary to determine when to stop. The ideal stopping point is that where any additional edges would cause a "short circuit" by leaving the manifold to join disparate regions of manifold space, causing a disproportionate update to the graph's geodesics.

The diameter and maximum betweenness centrality are good indicators of this stopping point because they summarize a graph's geodesics between all vertices. The graph diameter will decrease slowly as new edges are added, until a short circuit edge causes a large, sudden drop. A short circuit edge will similarly cause a large increase in maximum betweenness centrality, because the new edge becomes critical to a disproportionate number of shortest paths.

Using this new concept of differential graph quality, manifold learning practitioners can more confidently select hyperparameters for graph construction by observing the effect of added edges without needing to evaluate on a specific learning task. This method assumes initialization with a connected subgraph of the desired manifold-approximation graph, an easily satisfiable precondition.

3.6 Incremental Graph Construction

The second contribution of this chapter is an algorithm for efficient incremental neighbor graph construction and geodesic path computation. Both of the task-independent measures of graph quality discussed in Section 3.5 are derived from graph geodesics, which are typically computed using an all-pairs shortest path algorithm. These algorithms are computationally expensive, with $O\left(|V|^3\right)$ cost in the general case for weighted graphs [29]. This cost can be mitigated in the incremental setting, however, using efficient update rules for added edges.

An *incremental graph construction* algorithm is defined by its ability to produce a new graph from an existing state without re-computing from scratch. Given a graph generated with hyperparameters θ, denoted $G(\theta)$, and new hyperparameters θ' sufficiently close to θ, these algorithms construct a new graph $G(\theta')$ in an efficient manner.

3.6.1 The ϵk Graph

Recall the k-nearest and ϵ-close neighbor graph construction methods introduced in Section 2.2.3.1. In addition to their simple implementation and easily understood hyperparameters, these algorithms are also easily adapted to incremental formulations.

The bulk of the computation of traditional neighbor graph construction is expended to create an all-pairs distance matrix and sort its rows, a step that does not depend on the value of either k or ϵ. Once this preliminary step is complete, constructing graphs with different hyperparameters is computationally cheap. Thus, it is quite feasible to treat these algorithms as parametrized graph generating functions, for which different hyperparameter settings can be evaluated efficiently.

In this chapter, we use a variant of the ϵ-close graph construction algorithm that we refer to as the ϵk graph: rather than accepting all edges with length less than ϵ, only the smallest k such edges per vertex are added. Formally, this graph's set of edges is the intersection of the edge sets of a k-nearest neighbor graph and an ϵ-close neighbor graph. This algorithm mitigates the common issue where ϵ-close graphs become over-dense by enforcing a hard limit on the degree of each vertex, while still affording the flexibility of the real-valued ϵ parameter. This allows the ϵk graph to handle the common problem of non-uniform sampling from the manifold.

The ϵ-close algorithm always produces an undirected graph, but the addition of the k constraint introduces some asymmetry. To simplify our analysis, we symmetrize the resulting adjacency matrix using Equation 2.3.

3.6.2 Incremental Geodesics Update

The problem of incrementally updating graph geodesics has been well-studied in the graph theory community, typically for the development of dynamic graphs that allow update, removal, and re-weighting operations [23, 22]. In this general context, geodesic path computation time is reduced to $O\left(|V|^2 \log^3|V|\right)$ for each added edge. Other efforts have focused solely on edge addition, but only in the case of integer-weighted acyclic graphs [2]. For our application, however, a simpler update is possible with improved performance compared to the fully-dynamic update.

This new incremental update appears in Algorithm 1. Given a new edge (s, t) with weight w, the algorithm updates every path that can become shorter using this new edge.

Algorithm 1 Incremental Geodesics Update (INCRUPDATE)

```
Input: Adjacency matrix, $W$
  Geodesic distance matrix, $D$
  Predecessor matrix, $P$
  A new weighted edge, $(s, t, w)$
for $1 \leq i \leq |V|$ do
  for $1 \leq j \leq |V|$ do
    $d \leftarrow D_{is} + w + D_{tj}$
    if $d < D_{ij}$ then
      $D_{ij}, D_{ji} \leftarrow d$
      $P_{ij} \leftarrow s$ if $t = j$ else $P_{tj}$
      $P_{ji} \leftarrow t$ if $i = s$ else $P_{si}$
    end if
  end for
end for
$W_{st} \leftarrow w$
```

3.6.2.1 Theoretical Analysis

Algorithm 1 runs in $O\left(|V|^2\right)$ time and requires constant memory to process each new edge, as D and P are updated in-place. In contrast, a full re-computation of the geodesics would incur a $O\left(|V|^3\right)$ cost.

In the incremental setting, it is typical that $|E_k| \ll |V|$, where $|E_k|$ is the number of edges added in update k. In cases where $|E_k| \geq |V|$, however, the full geodesic recalculation may be used for efficiency.

The correctness of this update is derived from the Bellman criterion, stated as Lemma 1 in Brandes [12], adapted for edges: if an edge (s, t) with weight w lies on a shortest path $i \to j$, then

$$D_{ij} = D_{is} + w + D_{tj} \tag{3.10}$$

Lemma 3.6.1. *The incremental update (Algorithm 1) preserves the invariant that D_{ij} is the length of the shortest path between vertices i and j.*

Proof. Assume that D_{ij} contains the length of the shortest path between i and j not including the new edge e_{st}.

Given this new edge e_{st} with length w, there is a new shortest path between i and j including e_{st} if the length of either path i-s-t-j or i-t-s-j is shorter than the current shortest i-j path length. That is, it must satisfy either

$$D_{is} + w + D_{tj} < D_{ij} \tag{3.11}$$

or

$$D_{it} + w + D_{sj} < D_{ji}. \tag{3.12}$$

The incremental update considers each undirected $i - j$ path twice, checking both conditions 3.11 and 3.12. If a new shortest path is found using e_{st} in either consideration, both entries D_{ij} and D_{ji} are updated in-place.

As the algorithm runs, some entries of D will necessarily be larger than their true shortest path distance before they are updated, which can cause one of the above conditions to fail when considering an improved path $i - j$:

1. One or both of D_{is} and D_{tj} in condition 3.11 is also improved by e_{st} and has not been updated yet. Thus, the improved path is i-t-s-j so D_{it} and D_{sj} in condition 3.12 have not improved.

2. One or both of D_{it} and D_{sj} in condition 3.12 is also improved by e_{st} and has not been updated yet. Thus, the improved path is i-s-t-j so D_{is} and D_{tj} in condition 3.12 have not improved.

Therefore, at least one of conditions 3.11 and 3.12 must always be true for an improved $i - j$ path, regardless of the order in which the entries of D are updated. □

Lemma 3.6.2. *The incremental update (Algorithm 1) correctly updates P_{ij} to be the predecessor vertex to j in the shortest path from i to j.*

Proof. Assume P is correctly constructed to contain the predecessor vertex to j in the shortest path from i to j at position P_{ij}. The algorithm will update P only if a shorter path is found by introducing a new undirected edge e_{st} with weight w.

Let d be the path length from i to j going through e_{st}, $d = D_{is} + w + D_{tj}$

Consider the following cases:

- If $d \geq D_{ij}$, the new path is not shorter and no update is performed.
- If $t = j$, e_{st} directly connects s to j, so $P_{ij} \leftarrow s$.
- If $i = s$, e_{st} directly connects i to t, so $P_{ji} \leftarrow t$.
- If $t \neq j$, the shortest path from i to j is given by the shortest path from i to s, then edge e_{st}, then the shortest path from t to j, hence $P_{ij} \leftarrow P_{tj}$.
- If $i \neq s$, the shortest path from j to i is given by the shortest path from j to t, then e_{st}, then the shortest path from s to i, hence $P_{ji} \leftarrow P_{si}$.

□

3.6.3 Betweenness Centrality Update

The incremental geodesics update provides the required information for computing the graph diameter using Equation 3.3, but an additional step is required to compute betweenness centrality (Equation 3.5).

This process is described by Algorithm 2, which computes a simplified betweenness centrality metric given by Equation 3.9.

Algorithm 2 Simplified Betweenness Centrality (SIMPLEBC)

```
Input: Graph geodesics, $(D, P)$
Output: Betweenness centrality vector, $B$

$B_i \leftarrow 0$ for $1 \leq i \leq |V|$
for $1 \leq s \leq |V|$ do
  $c_i \leftarrow 0$ for $1 \leq i \leq |V|$
  $T \leftarrow V \setminus \{s\}$, in desc. $D_s$ order
  for $t \in T$ do
    $v \leftarrow P_{st}$
    if $v \neq s$ then
      $c_v \leftarrow c_v + c_t + 1$
      $B_t \leftarrow B_t + c_t$
    end if
  end for
end for
```

3.6.3.1 Theoretical Analysis

Computing the incremental update for betweenness centrality using Algorithm 2 requires $O\left(|V|^2 \log|V|\right)$ time using $O\left(|V|\right)$ memory.

For each vertex s in the graph, we consider all other vertices t in order of decreasing geodesic distance, D_{st}. This ordering requires a sort, incurring a cost of $O\left(|V| \log|V|\right)$ per vertex. We increment an intermediate counter, c, for each vertex along the $s \rightarrow t$ path, updating the betweenness B_t after considering all paths ending at t.

The algorithm is closely derived from Brandes' general form described in Section 3.4.2, but avoids costly geodesic re-computation while exploiting the lack of multiple equal-length shortest paths.

3.6.4 Combined Incremental Update

Algorithm 3 defines the combined procedure for incremental neighbor graph construction and quality measurement. The algorithm begins with traditional neighbor graph construction (NeighborGraph) and geodesic computation (AllPairsShortestPath) steps, given an input X, a distance metric δ, and initial graph construction parameters θ_0.

Then, for each incremental update with new parameters θ_t, the graph construction algorithm provides a set of new edges (NewEdges), using an incremental formulation as introduced in Section 3.6. These new edges are used to update geodesics with Algorithm 1, after which betweenness centrality is recomputed using Algorithm 2. The algorithm then emits the updated graph as well as the maximum diameter and maximum betweenness centrality quality measures.

These quality measures are used to aid the practitioner in choosing good graph construction hyperparameters, but do not permit a fully automated hyperparameter selection algorithm as presented. Future work to produce a robust stopping criterion based on these measures will require an intelligent combination of quality signals.

Algorithm 3 Incremental Geodesics

```
  Input: Input data, $X$
    Distance metric, $\delta$
    Incremental neighbor graph algorithm with a hyperparameter schedule, $[\theta_0 \ldots \theta_n]$
  $A_{ij} \leftarrow \delta(X_i, X_j)$ for $1 \leq i, j \leq |X|$
  $W \leftarrow$ NeighborGraph $(A, \theta_0)$
  $D, P \leftarrow$ AllPairsShortestPath $(W)$
  for $\theta_t \in [\theta_1 \ldots \theta_n]$ do
    for $e \in$ NewEdges $(A, \theta_{t-1}, \theta_t)$ do
      $W, D, P \leftarrow$ IncrUpdate $(W, D, P, e)$
    end for
    $B_C \leftarrow$ SimpleBC $(D, P)$
    Emit graph $W$
    Emit quality measures $\max(D)$, $\max(B_C)$
  end for
```

3.6.4.1 Theoretical Analysis

The total running time for Algorithm 3 depends on the average number of added edges per update, m, as well as the number of total updates performed, n. The cost of computing new edges to add is typically dominated by the cost of the incremental geodesic update, so we ignore it here. If $m \leq \log|V|$, the runtime cost is dictated by betweenness computation as $O(n|V|^2 \log|V|)$. Otherwise, the cost is dominated by edge updates and becomes $O(n|V|^3)$. Thus, this procedure is most advantageous when evaluating a graph at many intermediate stages of construction, with few added edges relative to the size of the graph.

3.7 Experimental Results

To demonstrate the effectiveness of the proposed incremental geodesic algorithm and its associated quality measures, we apply it to both real and synthetic data.

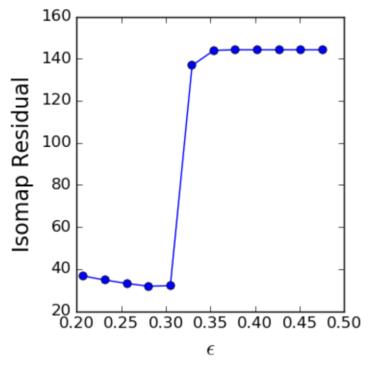
Synthetic datasets provide access to both manifold and original coordinates, which allows for a direct evaluation against the ground truth error metric described in Equation 3.1.

3.7.1 Ground Truth Evaluation: Swiss Roll

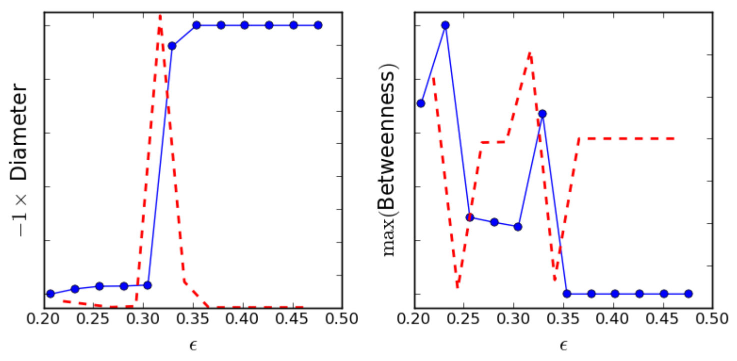
The first synthetic dataset is a Swiss roll, using original coordinates S and ground truth coordinates M:

$$\begin{aligned} S &= [\phi \sin \phi; \phi \cos \phi; w] \\ M &= [\phi; w] \\ \phi &= \text{Uniform}\,(0.5\pi, 3.75\pi) \\ w &= \text{Uniform}\,(0, 1) \end{aligned}$$

Figure 3.2 shows the relationship between the ground truth quality and the proposed quality measures on ϵk neighbor graphs constructed from a 500-point Swiss roll with $k = 14$ and increasing ϵ. A short-circuit edge introduced after $\epsilon = 0.305$ causes a large increase in ground truth error, clearly visible in the form of large spikes in the finite differences of the two proposed graph quality measures. Figure 3.3a illustrates this short-circuit edge, which leaves the surface of the manifold to connect points on opposing ends of the roll. Figure 3.3b demonstrates the benefit of using the proposed incremental graph update versus the traditional method of recomputing geodesics after each new set of edges is added for the Swiss roll experiment. As expected, computing from scratch has high fixed costs



(a) Ground truth graph quality using the formulation in Equation 3.1.



(b) Task-independent graph quality measures.

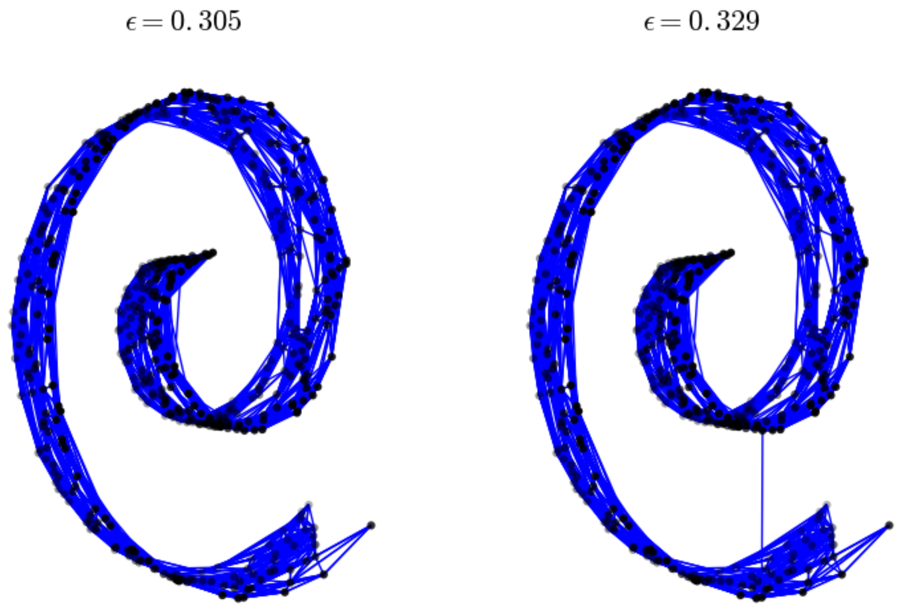
Figure 3.2: Graph quality measures over varying ϵ for a Swiss Roll. The solid blue lines show quality values changing with increasing ϵ, and the dashed red lines show the one-step change in quality between subsequent parameter settings.

per update, because no information from previous iterations is used. In contrast, the incremental approach scales with the number of added edges per update, resulting in significant time savings over the series of updates.

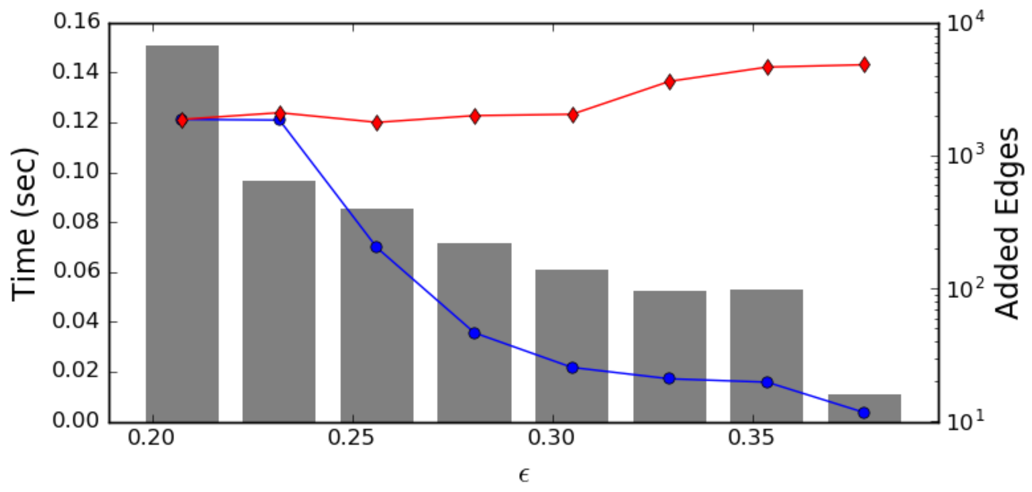
3.7.2 Ground Truth Evaluation: Isomap Faces

The second synthetic experiment involves more realistic data while maintaining access to ground truth information, using the Isomap Faces dataset [85]. The dataset contains 698 synthetically generated face images, for which the underlying three-dimensional manifold coordinates are known exactly: lighting angle, vertical face angle, and horizontal face angle. The 64×64 gray-scale images are preprocessed by flattening into 4096-dimensional vectors, then compared using the Euclidean distance metric. In this experiment, the k-nearest neighbors algorithm was used to generate edges for each incremental update.

Figure 3.4 indicates a large drop in manifold approximation fidelity between $k = 8$ and $k = 9$. The original space is difficult to visualize due to its high dimensionality, but both the ground truth error and proposed graph quality measures indicate that the new edges

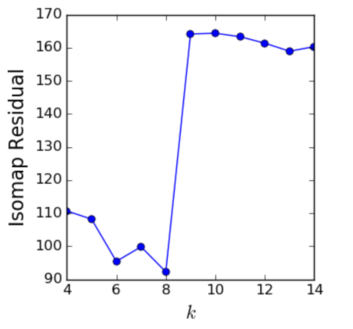


(a) Swiss roll neighbor graphs before (left) and after (right) the change in quality detected in Figure 3.2.

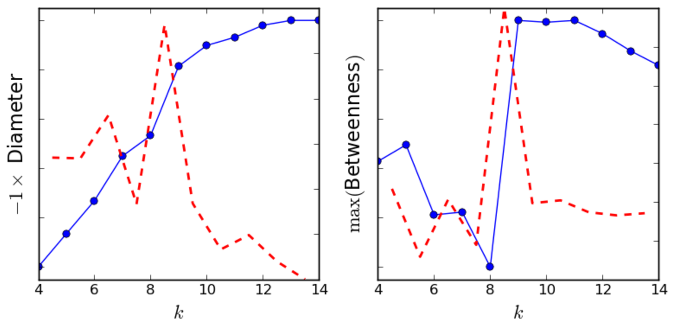


(b) Elapsed time for initialization and incremental updates during the Swiss roll experiment, shown for incremental updates (blue) vs full re-computation (red). Bars show the number of edges added per update (right axis).

Figure 3.3: Results of the synthetic Swiss roll experiment.



(a) Ground truth graph quality using the formulation in Equation 3.1.



(b) Task-independent graph quality measures.

Figure 3.4: Graph quality measures varying k on Isomap Faces. Lines as described in Figure 3.2.

introduced at $k = 9$ are responsible for change. Furthermore, our result mirrors the findings of Samko [78], while avoiding costly embedding calculations at every incremental update.

This second synthetic experiment demonstrates the utility of the proposed quality measures, as they allow the manifold learning practitioner to confidently choose a good setting for k without graph visualization or evaluation of a specific learning task.

Thus far, our experiments have been limited to synthetic datasets in which ground truth manifold coordinates are available. To demonstrate the effectiveness of our proposed method on more general problems, we apply it to two graph-based semi-supervised learning tasks using non-synthetic datasets.

3.7.3 Task-Based Evaluation: Image Classification

For the classification task, we apply the Local and Global Consistency (LGC) algorithm [111] to an image dataset. This first test used all 1440 images in 20 object classes from the COIL-20 dataset [70], representing each 128×128 gray-scale image as a 16384-dimensional vector and computing distances with the Euclidean distance metric. Graphs were generated incrementally using the ϵk algorithm with $k = 20$ and varying ϵ. For each generated graph, the LGC algorithm was run 10 times with a randomly-selected 50% of the

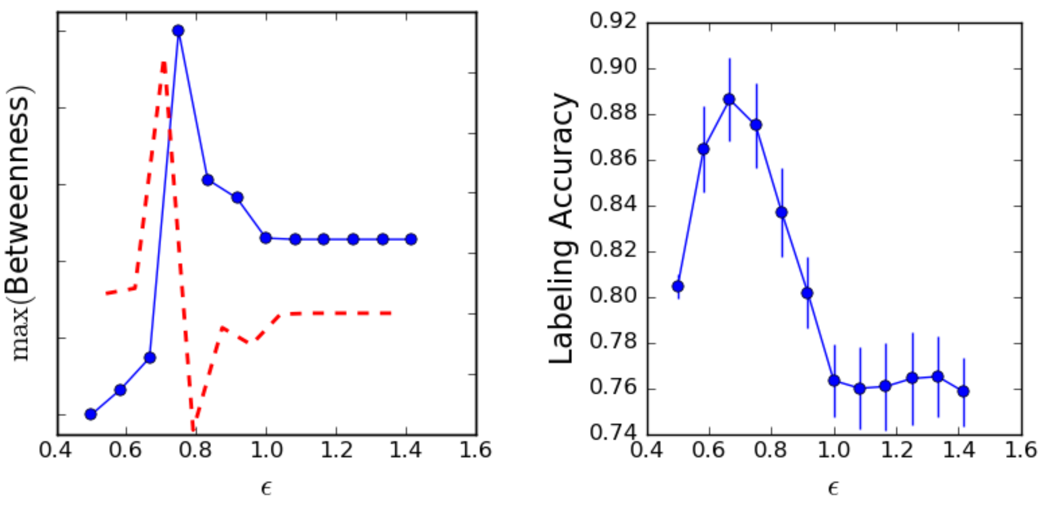


Figure 3.5: COIL-20 results over varying ϵ. At left, the change in the betweenness quality measure (red dashes) spikes after $\epsilon = 0.7$, corresponding to the highest average classification accuracy (right).

true class labels obscured each time, and its accuracy was computed as the average of each run's adjusted Rand Index [44].

The graph diameter quality measure was not used in this experiment, because the diameter of a graph with more than one connected component is infinite. For multi-class classification problems like this one, it is often desirable to produce graphs with many connected components, rendering the diameter quality measure uninformative.

The results in Figure 3.5 show that the betweenness graph quality measure can identify the optimal setting for ϵ, with a large spike in its finite difference corresponding to the point at which accuracy begins to degrade in the classification task.

3.7.4 Task-Based Evaluation: Binaural Localization

For the regression task, we used $3,600$ binaural recordings from the CAMIL dataset [20] recorded on a rotating platform with 360 degrees of motion. Our task is to predict the angle of the microphone with respect to a fixed audio source emitting white noise, given the full set of binaural audio samples and a subset of the true angles. Each one-second binaural

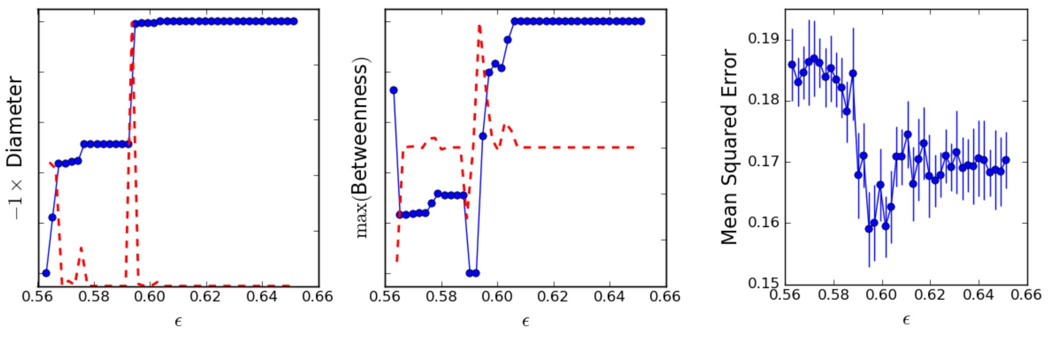


Figure 3.6: CAMIL regression results over varying ϵ. At left, the change in the both quality measures (red dashes) spike after $\epsilon = 0.59$, corresponding to the smallest regression error (right).

recording consists of two audio channels, which we preprocessed separately by converting to a standard spectrogram representation. We then concatenated each pair of spectrograms together to form 55212-dimensional vectors. The associated pan angles were preprocessed by converting each angle, θ, to a linearized two-dimensional representation, $(\sin\theta, \cos\theta)$. Graphs were generated incrementally with the ϵk algorithm with $k = 20$ and varying ϵ, using cosine distance to compare pairs of preprocessed vectors.

Each generated graph was evaluated by computing its diameter and maximum betweenness centrality, as well as with a simple semi-supervised learning task. In this task, unknown pan angles were predicted using linear regression on the graph's three-dimensional Isomap embedding.

The results in Figure 3.6 indicate that both betweenness and diameter quality measures are effective in locating the optimal setting for ϵ. Both measures show distinct spikes at the same value of ϵ, which corresponds to the graph that minimizes the error of the regression task.

3.8 Discussion

This chapter introduced the concept of differential graph quality, a task-independent method for analyzing the accuracy with which a graph conforms to the underlying manifold. While task-specific graph quality measures are directly related to the end goal of the manifold learning process, they are often expensive to compute and are rarely used for automatic parameter tuning. The new task-independent measures of graph quality are weakly coupled to the task of interest, yet still sufficient for tuning the free parameters of graph construction algorithms.

The second contribution of this chapter is a new method for efficient incremental neighbor graph construction and geodesic path computation. This algorithm mitigates the hyperparameter tuning problem by providing a low-cost mechanism for examining the proposed graph quality measures during the process of graph construction. The incremental update approach allows a manifold learning practitioner to make informed decisions about useful hyperparameter settings, offering concise information about how the graph's global geodesics change as edges are added without requiring expensive visualization or embedding steps.

In tandem, these contributions expedite the process of hyperparameter tuning, saving valuable time and producing graphs that result in desirable outcomes when used for manifold learning applications.

CHAPTER 4

MANIFOLD SPANNING GRAPHS

4.1 Motivation

Much of the manifold learning literature assumes that a simple k-nearest or ϵ-close neighbors graph will accurately model the topology of the underlying manifold of interest, and little attention is given to the process by which a manifold-approximation graph is built. In practice, these classical methods require expert tuning to produce reasonable results, and the efficacy of learning on the graph is strongly tied to the parameters of the graph construction algorithm used.

In this chapter, Section 4.2 demonstrates the hyperparameter sensitivity of traditional graph construction methods. Then, Section 4.3 introduces a new algorithm for unsupervised graph construction that reduces this parameter tuning burden.

4.2 Hyperparameter Sensitivity

To demonstrate the parameter sensitivity of classical graph construction methods, we consider the simple "swiss roll" dataset, as shown in Figure 1.2. Three-dimensional points on the roll are randomly distributed along the z-axis, while the x and y coordinates are parameterized by r, the distance from the origin, and θ, the angle created with the x-axis:

$$\begin{aligned} x &= r \sin\theta \qquad (4.1) \\ y &= r \cos\theta \\ z &= \text{Uniform}\left(0, 1\right). \end{aligned}$$

Under this parameterization, the optimal two-dimensional embedding is the θ–z plane. This information about the true underlying manifold means that we can apply a specialized ground truth graph quality measure, as introduced in Section 3.2. The swiss roll's nonlinearity is limited to the manifold coordinate θ, so we use a simple method that marks an edge as *incorrect* if it connects vertices p and q where $|\theta_p - \theta_q| > \epsilon$. For the following demonstration we used the threshold $\epsilon = 0.1$. This enables an interpretable measure of overall graph quality, the *edge error ratio*, defined as the ratio of incorrect edges to total edges in the graph.

We calculated the edge error ratio for many graphs produced by the k-nearest and ϵ-close neighbor algorithms, sampling from a set of reasonable hyperparameters for each. The results in Figure 4.1 show that, for the swiss roll dataset we considered, there is no value of k which produces a graph with both a low error ratio and a single connected component. This replicates previous results demonstrating that simple k-nearest neighbor algorithms are sensitive to noise [3]. Some values of ϵ avoid introducing incorrect edges while forming one connected component, but these parameters also tend to produce many more edges than desired, over-connecting the graph and losing the benefits of sparsity. The ϵ-close neighbors algorithm also exhibits instability by producing a sudden increase in error ratio after a certain ideal threshold is exceeded. This threshold value of ϵ is typically unknown to practicioners, and thus the algorithm requires expert understanding to tune effectively.

4.3 The Algorithm

The *Manifold Spanning Graph* is produced using a novel, bottom-up algorithm. The method is based on the parameter-free forest-joining framework introduced by Kruskal's minimum spanning tree algorithm [51], augmented with structural information captured by locally linear subspace fitting methods [41, 47, 88]. This combined approach constructs

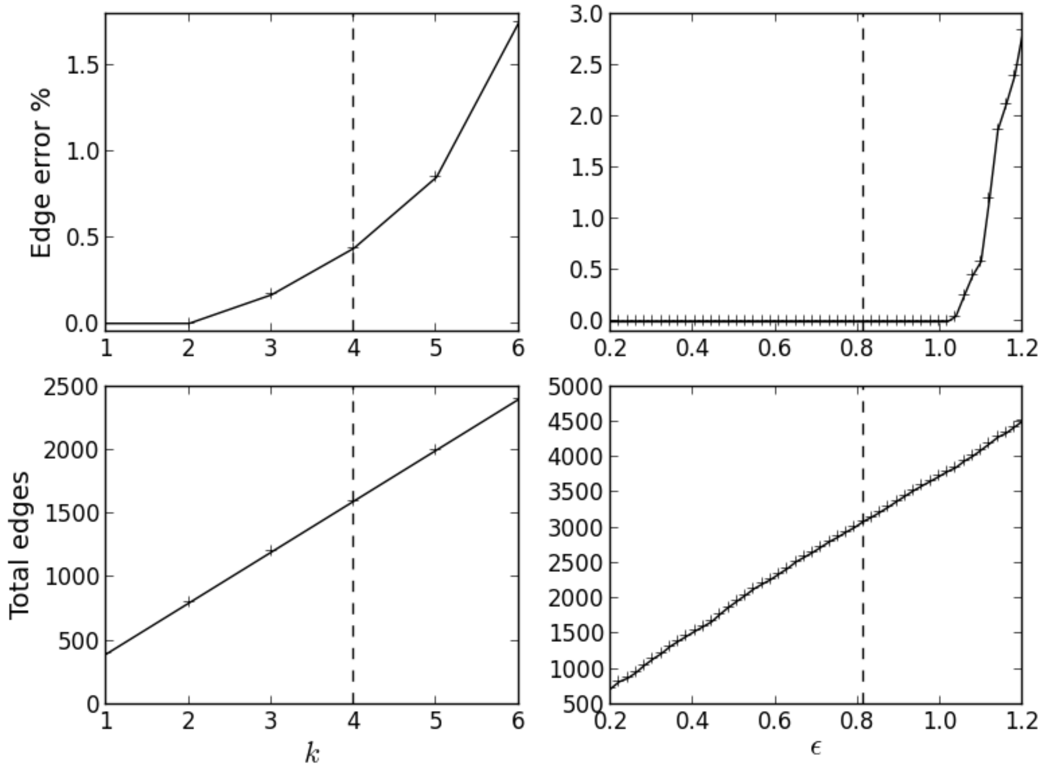


Figure 4.1: Sensitivity of k-nearest (left) and ϵ-close (right) algorithms. The dashed vertical lines represent the first hyperparameter value producing one connected component.

graphs that respect both inter-vertex distances and edge-to-manifold angles, without requiring the use of expert-tuned hyperparameters.

Inputs are X, an $N \times D$ matrix of D-dimensional points, d, an estimate of the intrinsic dimension of the underlying manifold, and m, the desired number of connected components in the final graph.

The high-level procedure can be divided into three steps:

1. (Section 4.3.1) Divide X into many small connected components of size at least $d+1$, then compute a d-dimensional linear subspace for each component via PCA.

2. (Section 4.3.2) Add edges between components that minimize both Euclidean distance and the largest angle created by the new edge and the two subspaces it joins. Continue adding edges until m connected components remain.

3. (Section 4.3.3) Provide additional structure to the graph with new edges, using the learned subspaces from step 1 as well as distance and angle thresholds learned in step 2.

The following sections examine each stage in detail, and Algorithm 4 describes the entire procedure.

4.3.1 Connectivity Forest and Subspace Construction

For a manifold with intrinsic dimension d, a locally linear patch can be accurately described with at least $d + 1$ points, assuming general position [101]. Therefore, the first task is to partition the input data into a *connectivity forest*, defined as connected groups of at least $d + 1$ points comprising many small, locally linear neighborhoods. This does not imply that each vertex in the graph will have degree d, because each neighborhood need not form a complete graph.

Instead, we begin by connecting each point with its nearest neighbor in input space. This relies on the assumption that the edge formed between a point and its first nearest neighbor in input space does not leave the underlying manifold. In practice this is a fairly weak assumption, given reasonably densely sampled input data. This assumption is also required for k-nearest neighbor graph construction, as well as ϵ-close neighbors in the case where $m = 1$.

Once the first-nearest neighbors are connected, we can compute the set of connected components $C^{(0)}$ [84]. For any components $C_i \in C^{(0)}$ where $|C_i| < d+1$, additional edges are added by merging C_i with C_j, where

$$C_j = \operatorname*{argmin}_{j \neq i} \min_{p \in C_i, q \in C_j} \|p - q\|_2 \,. \tag{4.2}$$

Again, we assume that pairs of edges close in input space are also close in the manifold's geodesic space.

When all components in $C^{(0)}$ satisfy the cardinality condition in Equation 4.2, we compute the set of linear subspaces:

$$P = \left\{ \mathrm{PCA}_d \left(C_i \right) \mid C_i \in C^{(0)} \right\} \tag{4.3}$$

Each P_i is represented by the first d principal components [46] of the vertices in C_i. The procedure at this stage is reminiscent of a Local PCA algorithm [47], albeit with a different selection criterion for the sets of points C_i. The primary difference, however, is that the subspaces P are not the final representation of the manifold, but only an intermediate set of constraints on edge connections. In this way, P may be viewed as an implicit set of charts [11, 57], with the inter-chart connections yet undefined. This use of principal angles between local subspaces has also been applied as part of the SAFFRON algorithm [34], which constructs a graph targeting a given intrinsic dimension while attempting to avoid the common failure case of "short circuit" edges.

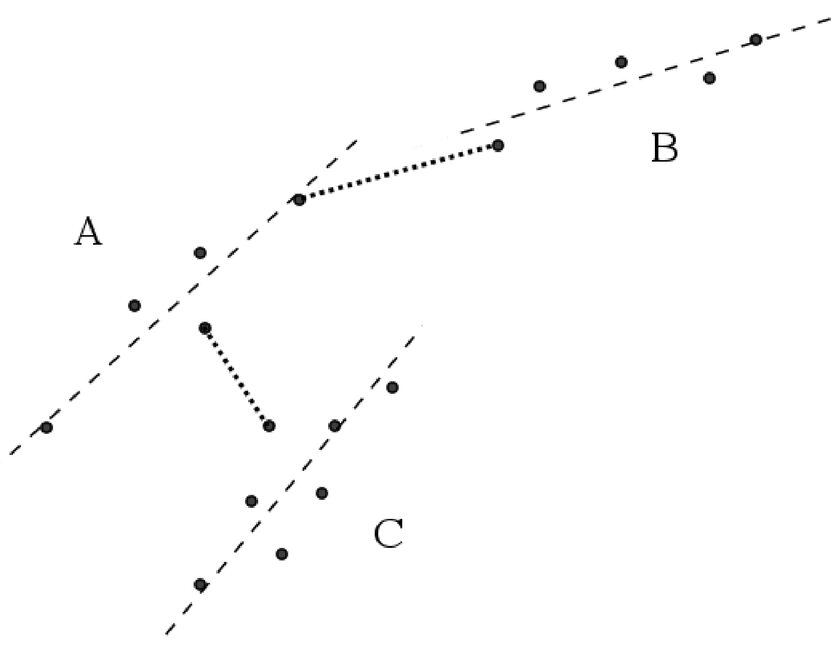


Figure 4.2: Component joining. Dashed lines indicate the P_i subspace of each component. Dotted lines show two possible edges that satisfy Equation 4.4. Only the A-B edge will be added because the A-C edge does not satisfy Equation 4.5.

4.3.2 Component Joining

Given a desired number of connected components m, we continue to add edges until $\left|C^{(t)}\right| = m$. For this task, we consider only edges between "adjacent" connected components, that is, (C_i, C_j) pairs satisfying Equation 4.2. In addition to this requirement, a candidate edge $(p \in C_i, q \in C_j)$ must satisfy:

$$\|\bar{e}\|_2 \leq \epsilon_{dist} \tag{4.4}$$

$$\max\left(d\left(P_i, \bar{e}\right), d\left(P_j, \bar{e}\right)\right) \leq \epsilon_{angle} \tag{4.5}$$

where $\bar{e} \equiv p - q$. In Equation 4.5, $d(\cdot, \cdot)$ is the projection F-norm [37], measuring the angles between the edge vector and each adjoining subspace. Figure 4.2 illustrates this process.

With appropriate threshold values of ϵ_{dist} and ϵ_{angle}, only edges that lie on the manifold will be added. Rather than assigning these thresholds as hyperparameters of the algorithm, we use an incremental step-up scheme:

1. Initialize ϵ_{dist} and ϵ_{angle} to zero.
2. Set ϵ_{dist} to the minimum inter-component distance, then set ϵ_{angle} to the minimum angle produced by the corresponding edge.
3. Add any edges that now meet the criteria in constraints 4.4 and 4.5.
4. Recalculate strongly connected components $C^{(t+1)}$, and unless $\left|C^{(t+1)}\right| = m$, return to step 2.

If desired, an additional constraint on the final degree of each vertex may be imposed to avoid edge overcrowding.

As edges are added and the cardinality of the remaining components in $C^{(t)}$ increases, each C_i is likely to lose its linearity property. For this reason, we refrain from recomputing PCA subspaces as in Equation 4.3, but instead maintain a mapping for elements from each $C^{(t)}$ to their original components in $C^{(0)}$, and thus P.

4.3.3 Edge Addition Post-processing

One limitation of the proposed algorithm is that only edges between “adjacent” connected components are added, which may result in a lack of edges that convey important structural information, even after the desired number of connected components is reached. Figure 4.3 illustrates this issue.

To mitigate these effects, we can run one iteration of post-processing, applying the same ϵ_{dist} and ϵ_{angle} constraints to filter candidate edges, with one additional constraint:

$$|\text{ShortestPath}\,(p,q)| \geq h \tag{4.6}$$

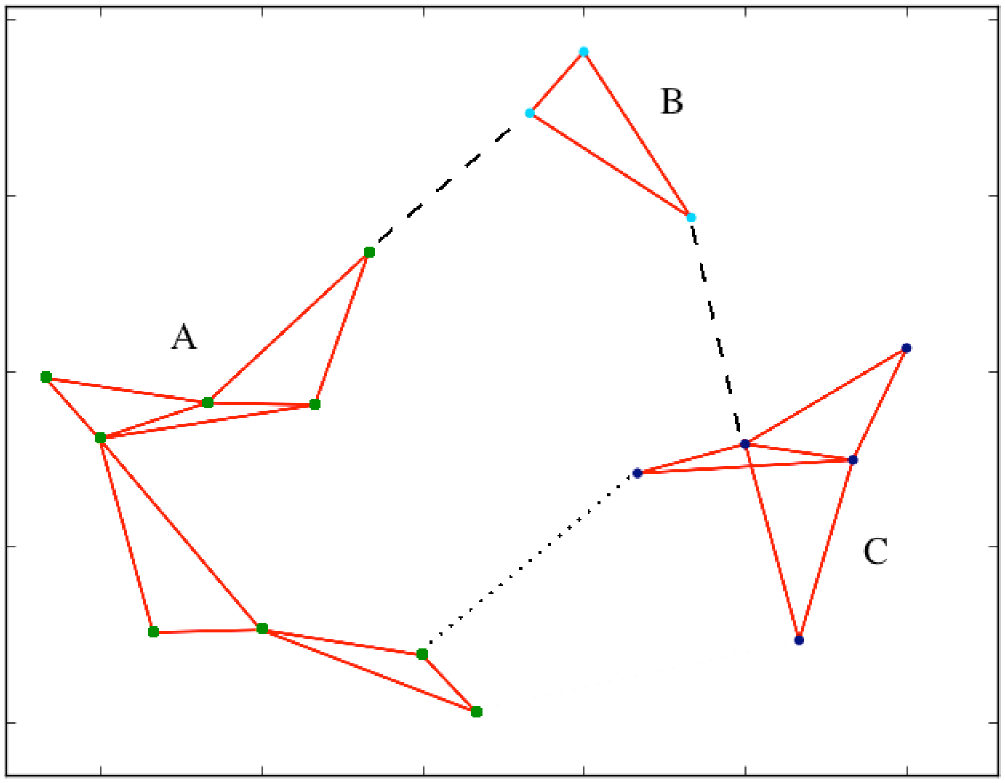


Figure 4.3: Limitations of only connecting adjacent components. Components A and C should connect along the dotted edge, but no edges will be added between them after they merge via the addition of the dashed A-B and B-C edges.

where ShortestPath (p, q) is the number of edges that a potential (p, q) edge would short-circuit. Thus, the parameter h acts as a lower bound on geodesic distance. This helps to ensure that only those edges that will add significant topological structure to the graph are added, because edges with low geodesic distance are already well-approximated by the existing graph. Setting $h \equiv d + 1$ is a reasonable default, so this extra parameter requires no expert tuning.

4.4 Theoretical Analysis

If no estimate for d is available a priori, the algorithm can infer a reasonable d from the explained variance of each PCA subspace, using Isomap's "elbow" heuristic on a local

Algorithm 4 The Manifold Spanning Graph Algorithm (MSG)

```
**Input:** Data matrix, $X \in \mathbb{R}^{N \times D}$
  Estimated intrinsic dimension, $d$
  Number of desired components, $m$
**Output:** A graph with $m$ connected components.
$\epsilon_{dist} \leftarrow 0$
$\epsilon_{angle} \leftarrow 0$
$C^{(0)} \leftarrow$ ConnectedComponents $(1\text{-NN}(X))$
Merge elements of $C^{(0)}$ using Equation 4.2 until $|C_i| \geq d + 1 \forall C_i \in C^{(0)}$.
Compute subspaces using Equation 4.3.
**while** $|C^{(t)}| \leq m$ **do**
  $\epsilon_{dist} \leftarrow \min_{p \in C_i, q \in C_j} \|p - q\|_2$
  $\epsilon_{angle} \leftarrow$ smallest principal angle made by the shortest $p - q$ edge.
  Add candidate edges satisfying Equations 4.2, 4.4, and 4.5.
  Re-compute connected components to form $C^{(t+1)}$.
**end while**
Add final edges satisfying Equations 4.4, 4.5, and 4.6.
```

scale. Alternatively, the intrinsic dimension of the underlying manifold can be estimated using any of several existing techniques [48, 56]. Thus, the d parameter can be considered optional.

The only other parameter is the number of desired connected components, m. This, too, is an optional setting, because m only determines the algorithm's stopping condition. In cases where no good value of m is known a priori, a sequence of graphs may be produced by setting $m \equiv 1$ and outputting the partial graph at each iteration.

Because the Manifold Spanning Graph algorithm is iterative, it is necessary to ensure that the convergence condition $|C^{(t+1)}| = m$ is met for some value of t. This convergence property is proved by examining the four possible cases:

1. At least one edge satisfies both constraints 4.4 and 4.5. All candidate edges connect disjoint connected components, so when the edge is added it merges at least two components and decreases $\left|C^{(t+1)}\right|$ by at least one.

2. No edge satisfies constraint 4.4, but at least one edge satisfies constraint 4.5. ϵ_{dist} is increased to the length of the minimum-angle edge satisfying constraint 4.5, and the first condition now applies.

3. No edge satisfies constraint 4.5, but at least one edge satisfies constraint 4.4. ϵ_{angle} is increased to the angle of the minimum-length edge satisfying constraint 4.4, and the first condition now applies.

4. No edge satisfies either constraint 4.4 or 4.5. ϵ_{dist} is increased to the minimum length of all candidate edges, ϵ_{angle} is increased to the chosen edge's angle, and the first condition now applies.

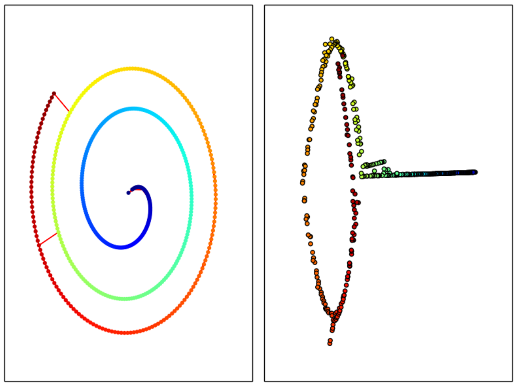
Thus, we prove that as $\lim_{t\to\infty}|C^{(t)}| \to 1$, the MSG algorithm converges in all cases where $m \leq |C^{(0)}|$.

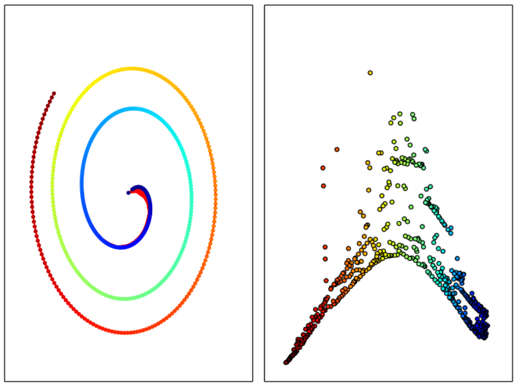
4.5 Experimental Results

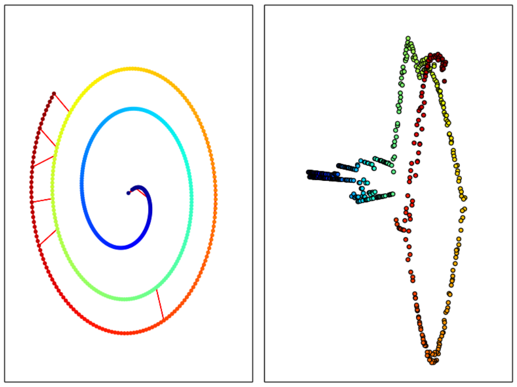
4.5.1 Parametric Swiss Roll

By contrast, the Manifold Spanning Graph algorithm presented in this chapter produces a connected graph with minimal incorrect edges, without requiring any hyperparameter tuning. Figure 4.4 illustrates the effect of incorrect edges on an Isomap embedding of the swiss roll data.

To evaluate the average performance of each algorithm, the number of incorrect edges and total edges were calculated over 200 randomly-generated swiss rolls following Equation 4.1. Hyperparameters k, ϵ, and b were optimized to produce one connected component for the first swiss roll, then the same parameters were used for all future examples. No hyperparameter tuning is required for the Manifold Spanning Graph algorithm. Figure 4.5 demonstrates that the MSG algorithm consistently produces near-zero incorrect edges, while still generating a reasonable number of total edges.


(a) k-nearest


(b) ϵ-close


(c) b-matching

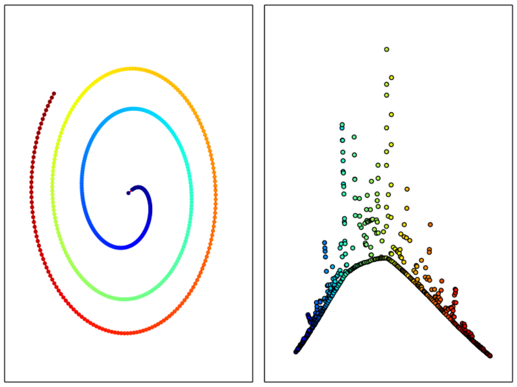
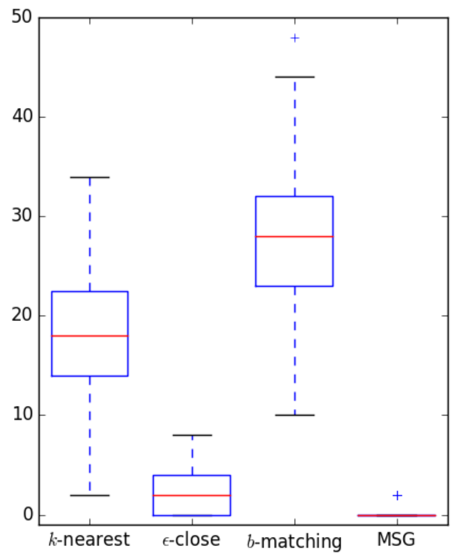
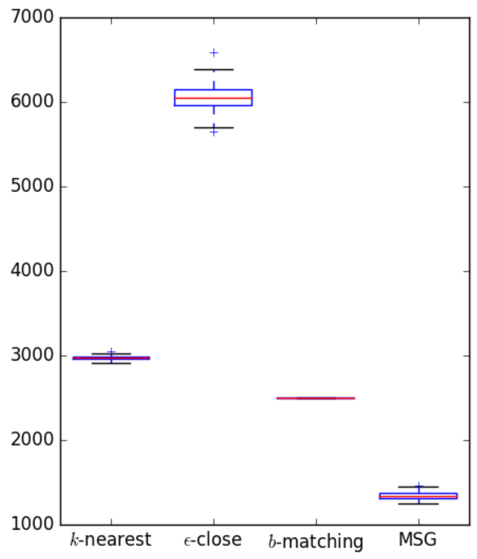

(d) Manifold Spanning Graph

Figure 4.4: Side view of Swiss Roll graphs and their resulting Isomap embeddings. For the algorithms that require hyperparameter tuning, the parameters k, ϵ, and b were set by choosing the smallest value that produced a single connected component. Sub-figures (a) and (c) show the disproportionate effect that short-circuit edges have on the learned embedding. We chose not to add edges after reaching one connected component to provide a common comparison point for each algorithm, and to underscore the problem of avoiding short-circuit edges even in the most conservative setting. As a result, the graphs producing sub-figures (b) and (d) are too sparse to produce rectangular Isomap embeddings. The Manifold Spanning Graph algorithm addresses this sparsity problem in Section 4.3.3.



(a) Incorrect edges



(b) Total edges

Figure 4.5: Summarized performance over 200 random swiss rolls, each with 500 points. The MSG algorithm produces graphs with almost zero incorrect edges, forming a single connected component with a modest number of total edges. The ϵ-close algorithm produces few bad edges, but tends to overconnect the graph. The k-nearest and b-matching algorithms produce a reasonable number of total edges, but many of these are incorrect. The high variance of these error rates also indicates sensitivity to noise.

4.5.2 MNIST digit clustering

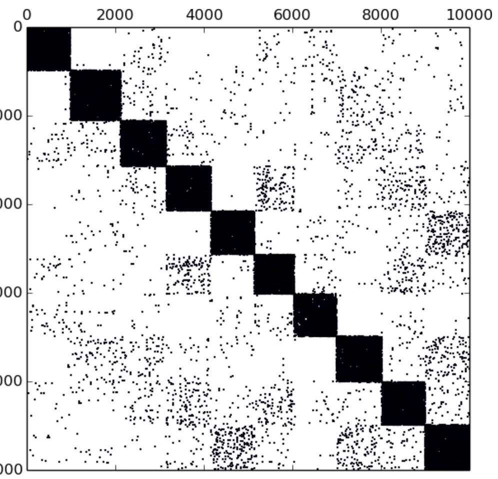
Real world data often make evaluating graph quality difficult, as the optimal low-dimensional embedding is typically unknown. However, the MNIST dataset test set of $10,000$ handwritten digits [53] allows for a simple "edge error" metric: the ratio of between-class to within-class edges. Each 28×28 gray-scale image is represented as a 784-dimensional pixel vector. This test demonstrates the Manifold Spanning Graph algorithm's ability to scale up to high-dimensional data, as well as exercising the ability to specify m, the number of desired connected components.

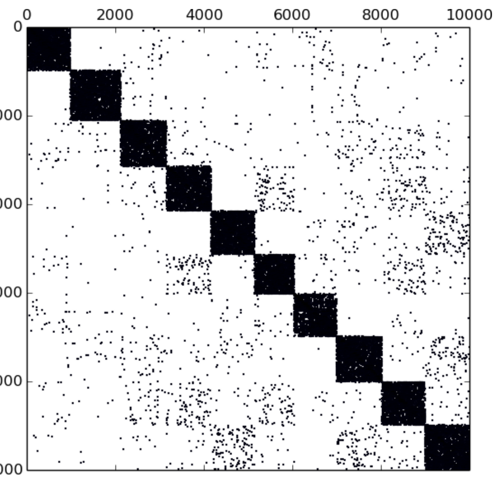
Figure 4.6 demonstrates the applicability of the MSG algorithm on high-dimensional data sets, again revealing a lower error rate without hyperparameter tuning. The k and b values were tuned by choosing the first k and b such that the number of connected components in the resulting graph was ≤ 10, the number of classes in the corpus. The same hyperparameter tuning with the ϵ-close algorithm found no value of ϵ producing ≤ 10 connected components with an edge error ratio under 50%, so we omit further results here.

As a final evaluation of the created graphs from Figure 4.6, a simple digit classification task was performed. Labeled exampled were generated by selecting 30 images at random from the $10,000$-image corpus without replacement, discarding selections that did not contain at least one example image for each digit. For each of the k-nearest, b-matching, and Manifold Spanning graphs, a simple label propagation algorithm [113] was used to classify the remaining 9970 images. The results in Table 4.1 show that the MSG algorithm produces high-quality graphs without parameter tuning.

4.6 Discussion

This chapter presented a novel algorithm for graph construction based on a bottom-up approach in the style of Kruskal's minimum spanning tree algorithm. The proposed method demonstrates improved accuracy over traditional graph construction algorithms without requiring any expert-tuned hyperparameters. The learned graphs approximate the true mani-


(a) k-nearest: $3256/45426 \approx 7.2\%$ bad edges ($k = 3$)


(b) b-matching: $1989/30000 \approx 6.6\%$ bad edges ($b = 3$)

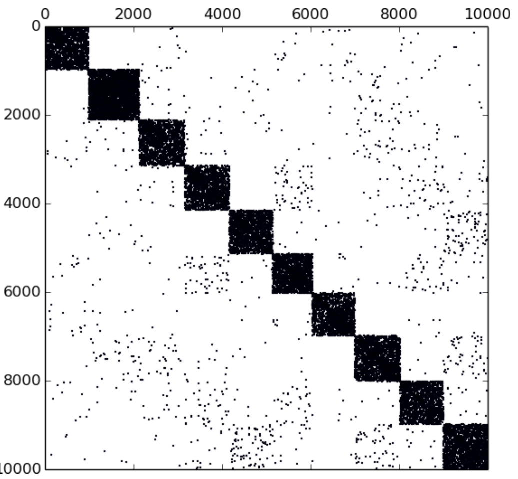

(c) MSG: $1118/23042 \approx 4.9\%$ bad edges ($m = 10, d = 2$)

Figure 4.6: MNIST neighbor graphs, each represented as a $10,000 \times 10,000$ binary adjacency matrix, sorted by digit label. Correct edges lie in the block-diagonal region, which corresponds to edges between images of the same digit. Incorrect edges are counted to compute each graph's edge error ratio.

	0	1	2	3	4	5	6	7	8	9
0	**913**	0	1	0	0	47	16	2	1	0
1	0	367	20	1	0	3	0	**737**	7	0
2	25	1	**936**	5	3	1	3	49	9	0
3	7	5	35	324	2	**552**	9	17	41	18
4	0	9	1	0	**936**	1	6	18	1	10
5	19	14	37	0	9	**707**	6	3	95	2
6	21	2	1	0	4	96	**830**	3	1	0
7	0	7	7	0	4	0	0	**999**	1	10
8	12	5	12	4	28	135	5	22	**748**	3
9	5	12	5	1	**477**	8	5	145	5	346

(a) k-nearest: 7106/10000 correct

	0	1	2	3	4	5	6	7	8	9
0	**811**	0	23	0	11	12	48	31	8	36
1	9	**591**	12	3	86	0	0	309	118	7
2	35	0	**740**	23	31	0	8	115	63	17
3	154	3	40	**334**	3	252	17	22	48	137
4	7	0	7	0	**823**	0	23	94	1	27
5	54	0	12	57	22	**535**	19	15	145	33
6	72	1	8	6	0	123	**719**	12	14	3
7	6	0	29	25	52	1	1	**900**	12	2
8	15	1	33	22	11	63	0	26	**770**	33
9	6	1	5	6	**398**	4	3	370	13	203

(b) b-matching: 6426/10000 correct

	0	1	2	3	4	5	6	7	8	9
0	**968**	0	0	0	0	3	6	2	1	0
1	0	**737**	3	23	0	0	3	369	0	0
2	39	177	**709**	5	2	4	7	83	6	0
3	11	5	8	**752**	6	31	0	14	134	49
4	5	18	2	0	**622**	2	3	5	1	324
5	10	5	0	44	20	**744**	39	12	14	4
6	23	7	0	0	1	0	**923**	2	2	0
7	6	25	3	2	39	8	0	**935**	0	10
8	77	14	4	64	19	50	3	27	**703**	13
9	6	1	3	10	**622**	3	1	33	15	315

(c) MSG: 7408/10000 correct

Table 4.1: Confusion matrices from the MNIST digit classification task. True digit labels are on the rows, with predicted labels along the columns. For each true label, the predicted label with the largest count is bold. The classifier using the MSG-generated graph achieved the best overall accuracy.

fold structure of the data, relying only on the smoothness and local linearity properties of Riemannian manifolds.

CHAPTER 5

MUTUAL NEIGHBORS GRAPH CONSTRUCTION

This chapter presents a novel graph construction algorithm that incrementally grows a high-quality manifold approximation graph, considering mutual neighbor connectivity and local edge sparsity in addition to a pairwise similarity metric. This method leverages common properties of manifold structures to decrease hyperparameter sensitivity, while outperforming existing graph construction algorithms on a variety of datasets and manifold learning tasks.

5.1 Background

In this chapter, we only consider cases in which the data lie on a single manifold. Therefore, we require the manifold approximation graph to have exactly one *connected component*, which guarantees the existence of some geodesic path between any two vertices.

A *minimum spanning tree* (MST) of a set of points $X \in \mathbb{R}^{N \times p}$ with a given pairwise distance metric, d, is an undirected, acyclic graph with one connected component that satisfies

$$\underset{E}{\operatorname{argmin}} \sum_{e_{ij} \in E} d\left(X_i, X_j\right)$$

[51].

The MST is typically insufficient for manifold learning applications, however, because its sparse edge set fails to represent true manifold geodesic distances. Instead, the MST is better viewed as a "backbone" subgraph of the desired manifold approximation graph that requires extra densification to accurately represent the true manifold structure. To this end, ensemble methods based on combinations of MSTs have been proposed, including the disjoint MST and perturbed MST graphs [105].

5.1.1 Short Circuit Pruning with Mutual Neighbors

In contrast to the problem of densifying a sparse graph backbone, graph construction can also be approached by pruning short-circuit edges from an overly dense initial graph [54, 35].

The Isograph algorithm is an instance of this approach that removes edges from the initial graph if they connect vertices that do not share a *mutual neighbor* [36]. If some vertex k satisfies $e_{ik} \in E$ and $e_{kj} \in E$, then k is a mutual neighbor of edge e_{ij}. This algorithm relies on the assertion that a candidate edge with a mutual neighbor is likely to lie close to the true manifold surface.

Variants of this concept have been proposed to enable a fast k-nearest neighbor graph construction algorithm [25], prioritizing edges with mutual neighbors in a partial neighbor graph to minimize expensive pairwise similarity measure computations.

5.2 Mutual Neighbors Graph Construction

This chapter introduces a novel graph construction algorithm which iteratively adds edges to a sparse initial backbone graph, favoring edges with existing mutual neighbor connections. This Mutual Neighbors Graph algorithm (MNG) addresses the problem in which we are given some subgraph (S) of an unknown, ideal manifold approximation graph (G). In this setting, our graph construction task consists of finding and adding the edges from G that are missing in S.

This task can be accomplished with an incremental greedy algorithm. For each iteration t and subgraph S^t, starting with the given backbone graph S^0, rank candidate edges by their likelihood to be found in G. Then select the most likely edge(s) to add to S^t, producing the next iteration's subgraph, S_{t+1}. Continue this process until no more viable candidate edges remain.

While any appropriate graph could be used for S^0, we suggest the use of a minimum spanning tree or a small-d disjoint MST. These graphs are guaranteed to have a single connected component, with minimal-weight edges that are unlikely to form short circuit connections across the manifold.

5.2.1 Candidate Edge Ranking

As we do not have access to G directly, some measure of edge fitness is required to rank candidate edges. This fitness measure should output a score for a given edge, e_{ij}, based on information derived from the current subgraph, S^t. We derive this score from two properties:

1. **Unweighted geodesic distance**. As described in the previous section, an edge with at least one mutual neighbor in a subgraph of G is likely to also exist in G. Generalizing this property, the unweighted geodesic distance in S^t between a candidate edge's endpoint vertices is inversely proportional to the fitness of the edge.

2. **Average degree**. For an edge e_{ij}, the average unweighted degree is the arithmetic mean of the unweighted degrees of both endpoint vertices:

$$\bar{d}_{ij} = \frac{d(i) + d(j)}{2}$$

   A high average degree indicates that the neighborhoods joined by e_{ij} are already densely connected, limiting the potential influence of the new edge on graph geodesics. Penalizing average degree also counteracts the tendency of geodesic distances to decrease as S^t becomes more densely connected.

Combining these elements with existing edge weights yields a cost function:

$$\text{cost}\left(e_{ij}|S^t\right) = \begin{cases} w_{ij}h_{ij}\bar{d}_{ij} & \text{if } e_{ij} \notin S^t \\ \infty & \text{otherwise} \end{cases},$$

where h_{ij} is the unweighted geodesic distance from vertices i to j along edges in S^t. To prevent any one factor from dominating the others, w is scaled to make the largest finite entry equal to one, while h and $\bar{d}$ are scaled to the range $[1, 2]$. This converts the unweighted geodesic distance and average degree components of the cost function into interchangeable factors for penalizing edges, without requiring hyperparameters for balance.

5.2.2 Candidate Edge Generation

Applying the cost function to all edges in the complete graph produces a dense cost matrix, C^t, which can be viewed as analogous to a pairwise distance matrix. This suggests the use of iterative variants of classical graph construction techniques to generate candidate edges.

One approach derives from the ϵ-close neighbor graph method, in which edges are added if their weight is below some threshold, ϵ. In the incremental formulation, we select the edge corresponding to the smallest entry in C^t, add it to the current subgraph S^t to produce S^{t+1}, then compute a new cost matrix C^{t+1} and repeat until the stopping condition is met. This approach is simple but requires the computation of a new cost matrix for each added edge, which limits its practicality for larger problems.

A more efficient approach is based on the 1-nearest neighbor method, in which the smallest edge for each vertex is chosen. Rather than adding one edge at a time, we add all minimum cost edges for each vertex before computing C^{t+1}. This formulation adds $O\left(|V|\right)$ edges per iteration, which significantly reduces the number of iterations and thus time spent updating costs. Adding edges for each vertex also promotes a more balanced degree distribution in the constructed graph.

Both proposed candidate generation methods require a stopping condition, preventing over-densification and losing the desired manifold structure. We introduce a cost threshold, γ^t, which serves to exclude candidate edges with greater costs. This threshold is initialized as $\gamma^0 = \infty$ and monotonically decreases each iteration according to the rule

$$\gamma^{t+1} = \min\left(\gamma^t, \text{confidence}\left(\mathcal{A}, C^t\right)\right),$$

where $\mathcal{A}$ is the set of edges added in iteration t and *confidence* is some statistic of the costs of these edges. A reasonable confidence measure should exclude edges with disproportionately high cost, preventing the addition of extra edges in regions of the graph that already have the correct structure.

In our experiments, we used the 99^{th} percentile of new edge costs with the 1-nearest neighbor candidate generation method. Monotonically decreasing γ^t ensures that, eventually, all candidate edges will have costs above the threshold, so no new edges will be added and the iteration can terminate.

5.2.3 Algorithm Analysis

The full Mutual Neighbors Graph algorithm is given in Algorithm 5.

Starting from an initial graph with $|V| = n$ vertices, each iteration of MNG requires the computation of an $n \times n$ cost matrix. Selecting and pruning candidates requires $O(n^2)$ time, and produces at most n added edges per iteration. The dominating cost of MNG, therefore, is the cost matrix calculation.

The average degree matrix $\bar{d}$ is trivially computed with $O(n^2)$ time and memory, but the unweighted geodesic distance matrix h is more complicated, because it requires the computation of shortest path length between all pairs of vertices. Floyd gave a generic $O(n^3)$ algorithm [29], but later efforts have exploited the undirected and unweighted properties to yield theoretically (if not practically) faster algorithms [87]. In the MNG algorithm, the number of edges in S^t is no greater than tn, which makes the practical h computation time $O\left(tn^2 + n^2 \log n\right)$ [30].

Algorithm 5 Mutual Neighbors Graph Construction

```
**Input:** Distance matrix, $D$
    Backbone graph, $S^0$
**Output:** Final manifold approximation graph, $S^n$
$\gamma^0 \leftarrow \infty$
**for** $t \leftarrow [0, 1, \dots]$ **do**
    **for all** $e_{ij} \in D$ **do**
        $C_{ij}^t \leftarrow \text{cost}\left(e_{ij} | S^t\right)$
    **end for**
    $\mathcal{A} \leftarrow \emptyset$
    **for** $i \leftarrow [1, \dots, |V|]$ **do**
        $k \leftarrow \text{argmin}_j\, C_{ij}^t$
        **if** $C_{ik}^t < \gamma^t$ **then**
            $\mathcal{A} \leftarrow \mathcal{A} \cup \{e_{ik}\}$
        **end if**
    **end for**
    **if** $|\mathcal{A}| = 0$ **then**
        **return** $S^t$
    **end if**
    $S^{t+1} \leftarrow S^t \cup \mathcal{A}$
    $\gamma^{t+1} \leftarrow \min\left(\gamma^t, \text{confidence}\left(\mathcal{A}, C^t\right)\right)$
**end for**
```

5.2.4 Demonstration

To illustrate the state of the proposed algorithm during each iteration, we apply MNG to a synthetic image dataset with a known manifold structure. The dataset consists of 625 grayscale images of Gaussian noise, each overlaid with a smaller image of the Mona Lisa at a fixed coordinate [33]. By construction, the true manifold structure is a square grid in two dimensions, representing the pixel offsets of the Mona Lisa sub-image.

Figure 5.1 shows the evolution of S^t, plotted on both ground-truth and Isomap coordinates. The series of Isomap embeddings (colored by the ground truth y-coordinate) demonstrates the effect of the graph's increasing density, gradually joining the discon-

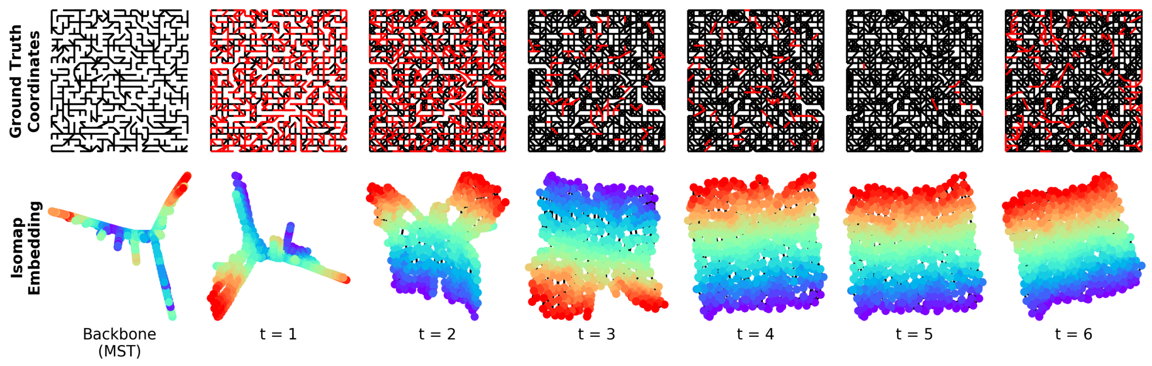


Figure 5.1: Iterations of the Mutual Neighbors Graph algorithm, demonstrated on the synthetic Mona Lisa image manifold. The top row depicts S^t on ground truth grid coordinates, with red lines indicating $\mathcal{A}$, the edges added by MNG during that iteration. The bottom row shows a 2-dimensional Isomap embedding of S^t, colored by the ground truth y coordinate.

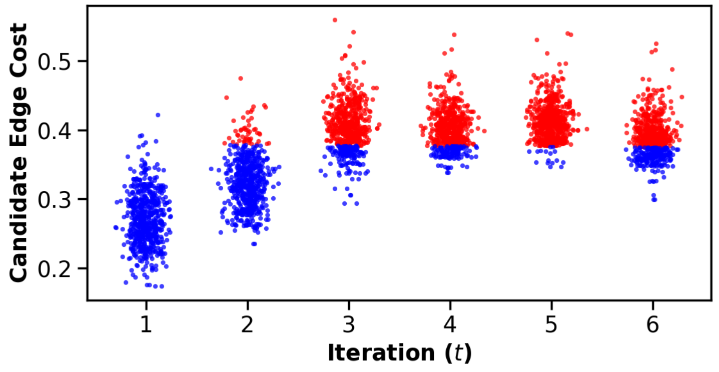


Figure 5.2: Cost distributions for candidate edges at each iteration, from the application of MNG illustrated in Figure 5.1. Edges in $\mathcal{A}$ are colored blue, while candidate edges above the γ^t threshold are colored red. Gaussian noise along the x-axis was added to spread out overlapping points.

nected segments of the initial minimum spanning tree and eventually converging on the true grid structure.

Figure 5.2 visualizes the distribution of candidate edge costs at each iteration of the algorithm. Candidates with cost greater than γ^t are red, while the final $\mathcal{A}$ set is blue. The choice of confidence function controls the progression of γ, and thus the rate of convergence of the algorithm and the total number of edges added.

5.3 Experimental Results

In this section, we investigate the performance of the proposed Mutual Neighbors Graph algorithm on a variety of datasets. Due to the unsupervised nature of the manifold approximation task, we evaluate the constructed graphs indirectly, using the performance of a fixed manifold learning algorithm as a proxy, using the formulation described in Section 3.3. When all other factors are held constant, the graph that produces the best end result can be considered a better approximation of the true manifold structure.

5.3.1 Swiss Roll

Figure 5.3 shows results from a randomized Swiss roll experiment. Using ten randomized swiss rolls generated from the distribution $[r \sin\theta; r\cos\theta; \text{Uniform}\,(-1,1)]$ with given parameter vectors r and θ, we compute two evaluations of graph quality based on a two-dimensional Isomap embedding. The first compares pairwise distances in the ground truth r-θ space with pairwise distances in the embedded space. The second evaluation computes mean squared error of a linear regression model predicting θ given the embedded coordinates.

Both tasks require that the graph captures the underlying two-dimensional structure of the Swiss roll manifold, though each penalizes distortion in the graph differently. Among all graph construction methods tested, for both tasks, the graphs produced by MNG are consistent in having the smallest error.

5.3.2 MNIST Digit Classification

To demonstrate the effectiveness of MNG on a non-synthetic dataset with a more difficult learning task, we consider the problem of classifying handwritten digit images from the MNIST database [53].

Using all 10,000 images from the MNIST test set, we generated 50 semi-supervised learning problems by partitioning the data into 10% labeled and 90% unlabeled instances. Labeled examples were chosen at random, though partitions without any labels per digit

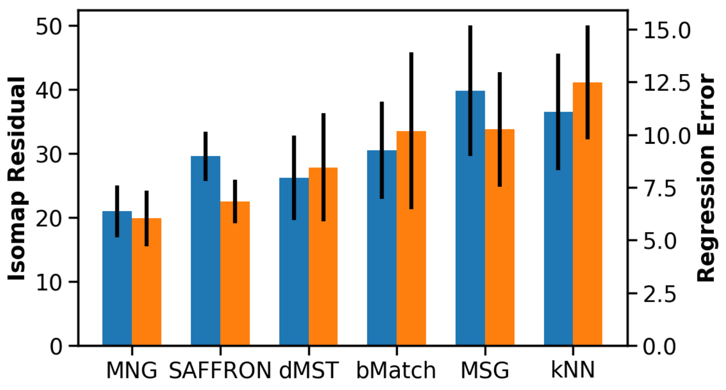


Figure 5.3: Results of randomized Swiss roll experiments. Blue bars (left axis) show the error between the graph's 2d Isomap embedding and ground truth coordinates. Orange bars (right axis) show mean squared error of regression. Each method's results use the best-performing hyperparameter, and lower error indicates better performance. Black vertical lines show one standard deviation above and below the mean.

class were rejected. Classification accuracy was computed by running the Local and Global Consistency label propagation algorithm on each of the label partitions [111].

As shown in Figure 5.4, the Mutual Neighbors Graph significantly improves classification accuracy over existing graph construction algorithms, with each algorithm using the optimal hyperparameter setting. Due to the large number of points in the dataset and limited computational resources, the SAFFRON and MSG algorithms were not evaluated in this or the following experiments.

5.3.3 Semi-Supervised Learning Benchmarks

We next evaluate on a standard set of benchmark datasets for semi-supervised learning tasks [15], which offers a larger variety of graph construction evaluation settings. These benchmarks include visual, textual, synthetic, and observed data from different sources with varying adherence to the manifold assumption. The dimensionality of the data also varies, ranging from 117 features for set 4 (BCI) to 11960 features for set 9 (Text).

We used seven of these datasets to evaluate MNG more thoroughly. We excluded set 3 (COIL2) because it is a binary version of set 6 (COIL), and set 8 (SecStr), which would

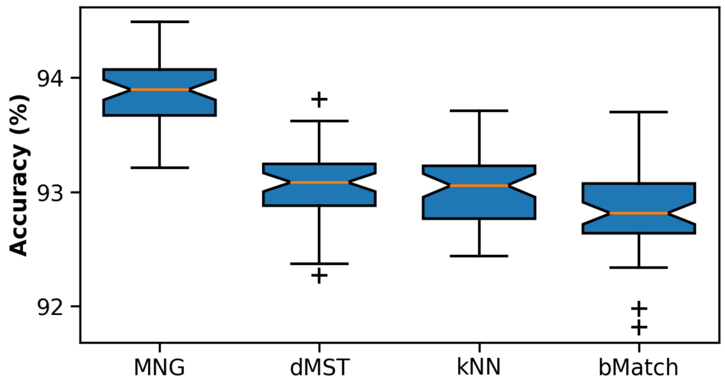


Figure 5.4: MNIST classification results. Box plots show the distribution of accuracy scores for each method's graph when applied to 50 random instances of a label propagation task, with fliers indicating outlier scores and notches showing the 95% confidence interval for the median. Higher accuracy indicates better performance.

require a prohibitive amount of memory to represent as a pairwise distance matrix. The remaining datasets contain 1500 points each, except set 4 (BCI) with 400 points.

Each dataset provides twelve random partitions of labeled and unlabeled points, with 100 labeled examples per split. Classification was performed with the same LGC label propagation classifier used in the MNIST experiment. We used seven splits per dataset to identify optimal hyperparameters for each evaluated graph construction method, and Table 5.1 reports accuracy on the remaining five splits using the selected hyperparameters.

The MNG algorithm produces graphs that achieve the best performance in five of seven benchmarks, with comparable performance on the other two. The most difficult dataset for the MNG algorithm was set 6 (COIL), for which the ground truth structure is a collection of disjoint, ring-shaped manifolds. This means that graphs derived from minimum spanning trees (which are, by construction, connected graphs) are at a particular disadvantage.

5.4 Discussion

This chapter presented a novel graph construction method that makes use of the simple observation that vertices with mutual neighbors in a sparse subgraph are likely to be connected in the full manifold approximation graph. With this in mind, we developed an

	MNG	b-Matching	dMST	k-nearest
g241c	**64.41**	63.60	63.19	64.08
g241n	**69.20**	67.89	66.80	68.33
Digit1	**96.03**	95.79	95.92	95.84
USPS	**95.27**	94.64	94.87	94.59
COIL	81.09	**83.96**	79.43	82.20
BCI	65.65	65.20	65.40	**65.90**
Text	**75.16**	73.32	72.47	72.99

Table 5.1: Average classification accuracy on semi-supervised learning benchmark datasets. The best performing method for each dataset is highlighted in bold.

incremental graph construction algorithm around the idea of scoring candidate edges given an existing subgraph.

The Mutual Neighbors Graph algorithm is based on three concepts that promote desirable properties of a manifold approximation graphs:

- starting from a connected subgraph derived from the minimum spanning tree,
- promoting edges with mutual neighbors, and
- penalizing edges in densely connected regions of the graph.

The proposed method is simple to implement, analyze, and use, and consistently outperforms both classical and state of the art graph construction methods on a variety of domains.

CHAPTER 6

MANIFOLD WARPING: MANIFOLD ALIGNMENT OVER TIME

The advent of large, often high-dimensional, digital data sets has made automated knowledge extraction a critical research focus in machine learning. It is common to find real-world sequential data sets that encode the same information with disparate surface feature representations, such as sensor network data, activity and object recognition corpora, and audio/video streams. In these cases, an automated technique for discovering correlations between datasets will allow easy transfer of knowledge from one domain to another, avoiding costly or infeasible re-learning.

In this chapter, which is based on joint work with Hoa Vu, we present a framework that combines manifold alignment [39, 94] and dynamic time warping (DTW) [77] for aligning two such sequential data sets.

6.1 Background

6.1.1 Manifold Alignment

In manifold alignment [94], we are given two data sets $X \in \mathbb{R}^{n_X \times d_X}$ and $Y \in \mathbb{R}^{n_Y \times d_Y}$ where $\mathbb{R}^{m \times n}$ denotes a m by n real matrix. In X and Y, each row is an *instance*. $W^{(X)} \in \mathbb{R}^{n_X \times n_X}$ and $W^{(Y)} \in \mathbb{R}^{n_Y \times n_Y}$ are matrices that provide pairwise similarities between the instances in X and Y, respectively. These matrices are usually constructed as weighted adjacency matrices of neighbor graphs, optionally applying a heat kernel function to edge weights. We also have a warping matrix $W^{(X,Y)} \in \mathbb{R}^{n_X \times n_Y}$ that specifies the correspondences between instances in X and Y. Typically,

$$W_{ij}^{(X,Y)} = \begin{cases} 1 & \text{if } X_i \text{ corresponds to } Y_j \\ 0 & \text{otherwise} \end{cases}. \tag{6.1}$$

Suppose we have a mapping that maps X, Y to $F^{(X)} \in \mathbb{R}^{n_X \times d}, F^{(Y)} \in \mathbb{R}^{n_Y \times d}$ in a latent space with dimension $d \leq min(d_x, d_y)$. In the underlying matrix representation, any row i of X is mapped to row i of $F^{(X)}$, and a similar relation holds for Y and $F^{(Y)}$.

We form the following loss function for the mapping as follows. The first term indicates that corresponding points across data sets should remain close to each other in the embedding. The last two terms specify that, within an input set, points close in the original space should remain close in the embedding. The factor μ controls how much we want to preserve inter-set correspondences versus local geometry.

$$\begin{aligned} L_1\left(F^{(X)}, F^{(Y)}\right) = \mu & \sum_{i \in X, j \in Y} \|F_i^{(X)} - F_j^{(Y)}\|^2 W_{ij}^{(X,Y)} \\ +(1-\mu) & \sum_{i,j \in X} \|F_i^{(X)} - F_j^{(X)}\|^2 W_{ij}^{(X)} \\ +(1-\mu) & \sum_{i,j \in Y} \|F_i^{(Y)} - F_j^{(Y)}\|^2 W_{ij}^{(Y)} \end{aligned} \tag{6.2}$$

The notation $i \in X$ refers to the range $1 \leq i \leq n_X$. We can combine $W^{(X)}$, $W^{(Y)}$, and $W^{(X,Y)}$ into a joint similarity matrix W, defined

$$W = \begin{bmatrix} (1-\mu)W^{(X)} & \mu W^{(X,Y)} \\ \mu W^{(Y,X)} & (1-\mu)W^{(Y)} \end{bmatrix}. \tag{6.3}$$

Then, we combine $F^{(X)}, F^{(Y)}$ into F where $F = \begin{bmatrix} F^{(X)} \\ F^{(Y)} \end{bmatrix}$. Let $F_{i,k}$ denote the element (i,k) of F and $F_{\cdot,k}$ denote the kth column of F. Then the loss function can be rewritten as

$$\begin{aligned} L_1(F) = & \sum_{i,j} \|F_i - F_j\|^2 W_{ij} \\ = & \sum_k \sum_{i,j} \|F_{ik} - F_{jk}\|^2 W_{ij} \\ = & \ 2\sum_k F_{\cdot,k}^\top L F_{\cdot,k} = 2tr\left(F^\top L F\right), \end{aligned} \tag{6.4}$$

where L is the graph Laplacian of F. Let D be a diagonal matrix in which each diagonal element is the degree of the corresponding vertex. The optimization problem becomes:

$$\underset{F}{\operatorname{argmin}}\,(L_1) = \underset{F}{\operatorname{argmin}}\left(tr\left(F^\top L F\right)\right) \tag{6.5}$$

This matches the optimization problem of Laplacian Eigenmaps [4], except that in this case the similarity matrix is a joint matrix produced from two similarity matrices. As with Laplacian Eigenmaps, we add a constraint $F^\top D F = I$ to remove an arbitrary scaling factor and avoid a collapse to a subspace with dimension less than d. For example, this constraint prevents the trivial mapping to a single point. The solution $F = [f_1, f_2, \ldots, f_d]$ is given by the d smallest nonzero eigenvectors of the generalized eigenvalue problem: $Lf_i = \lambda D f_i$ for $i = 1, \ldots, d$.

We can also restrict the mapping to be linear by instead solving the optimization problem

$$\underset{\phi}{\operatorname{argmin}}\left(tr\left(\phi^\top V^\top L \phi V\right)\right) \text{ subject to } \phi^\top V^\top D \phi V = I, \tag{6.6}$$

where V is the joint data set

$$V = \begin{pmatrix} X & 0 \\ 0 & Y \end{pmatrix},$$

ϕ is the joint projection

$$\phi = \begin{bmatrix} \phi^{(X)} \\ \phi^{(Y)} \end{bmatrix},$$

L is the graph Laplacian, and D is the degree matrix. The resultant linear embedding is then $X\phi^{(X)}$ and $Y\phi^{(Y)}$, instead of $F^{(X)}$ and $F^{(Y)}$. The solution for $\phi = [\phi_1, \phi_2, \dots, \phi_d]$ is the d nonzero smallest eigenvectors of the generalized eigenvalue problem

$$V^\top L V \phi_i = \lambda V^\top D V \phi_i$$

for $i = 0, \dots, d$.

6.1.2 Dynamic Time Warping

We are given two sequential data sets

$$X = [x_1^\top, \dots, x_n^\top]^\top \in \mathbb{R}^{n\times d}$$
$$Y = [y_1^\top, \dots, y_m^\top]^\top \in \mathbb{R}^{m\times d}$$

in the same space with a distance function $X \times Y \to \mathbb{R}$. Let $P = \{p_1, \dots, p_s\}$ represent an alignment between X and Y, where each $p_k = (i, j)$ is a pair of indices such that x_i corresponds with y_j. Because the alignment is restricted to sequentially-ordered data, we impose the additional constraints:

$$p_1 = (1,1) \quad (6.7)$$
$$p_s = (n,m) \quad (6.8)$$
$$p_{k+1} - p_k = (1,0) \textit{ or } (0,1) \textit{ or } (1,1) \quad (6.9)$$

That is, an alignment must match the first and last instances and cannot skip any intermediate instance. This also yields the property that no two sub-alignments cross each

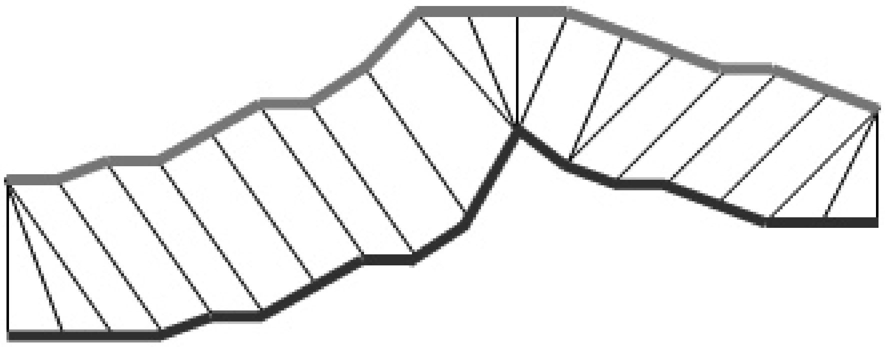


Figure 6.1: A valid time-series alignment

other. Figure 6.1 is an example of a valid alignment. We can also represent the alignment in unweighed adjacency matrix form, where

$$W_{ij} = \begin{cases} 1 & \text{if } (i,j) \in P \\ 0 & \text{otherwise} \end{cases} . \tag{6.10}$$

To ensure that W represents an alignment that satisfies the constraints in Equations 6.7, 6.8, and 6.9, W must be in the following form: $W_{1,1} = 1$, $W_{n,m} = 1$, none of the columns or rows of W is a 0 vector, and there must not be any zeros between any two ones in a row or column of W. We call a W that satifies these conditions a ***DTW matrix***. An optimal alignment is the one that minimizes the loss function with respect to the DTW matrix W:

$$L_2(W) = \sum_{i,j} dist\left(x_i, y_j\right) W_{ij} \tag{6.11}$$

A naïve search over the space of all valid alignments would take exponential time; however, dynamic programming can produce an optimal alignment in $O(nm)$.

Dynamic time warping has been used effectively for time-series alignment, but it requires an inter-set distance function, which usually implies that both input data sets must have the same dimensionality. DTW may also fail under arbitrary affine transformations of one or both inputs.

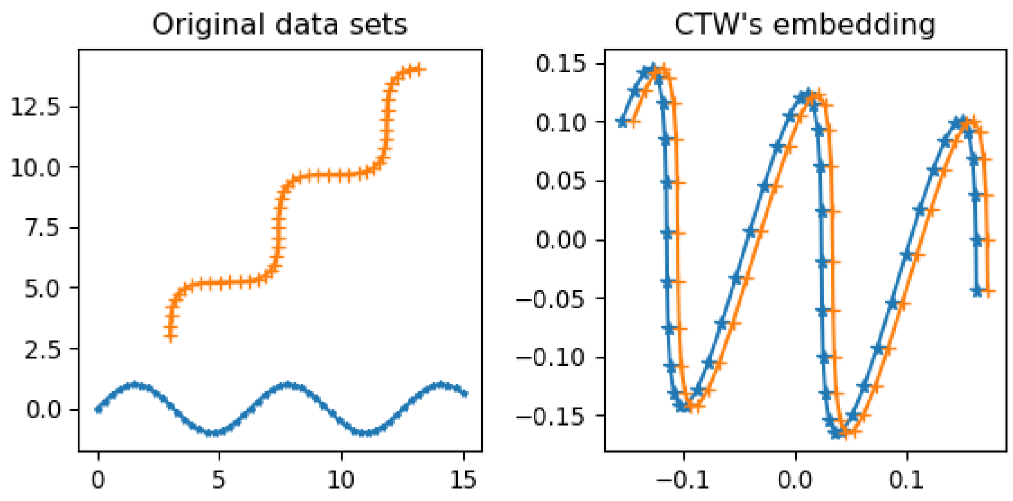


Figure 6.2: Sinusoidal curves before and after applying Canonical Time Warping.

6.1.3 Canonical Time Warping

Canonical Time Warping (CTW) [112] aims to solve these two deficiencies of DTW by alternating between Canonical Correlation Analysis (CCA) and DTW until convergence, as illustrated in Figure 6.2.

In the case where inputs are of different dimensions, CTW first projects both data sets into a shared space using Principal Components Analysis (PCA) [46]. The algorithm does not always converge to a global optimum, but CTW still improves the performance of the alignment when compared to applying DTW directly.

However, CTW fails when the two related data sets require nonlinear transformations to uncover the shared manifold space. We illustrate such a case in Figure 6.3, in which two sequential data sets are two $\sin^2(x)$ curves; one lying on a plane and the other on a Swiss roll. In this case, the CCA projection that CTW relies on will fail to unroll the second curve, making a good DTW alignment impossible.

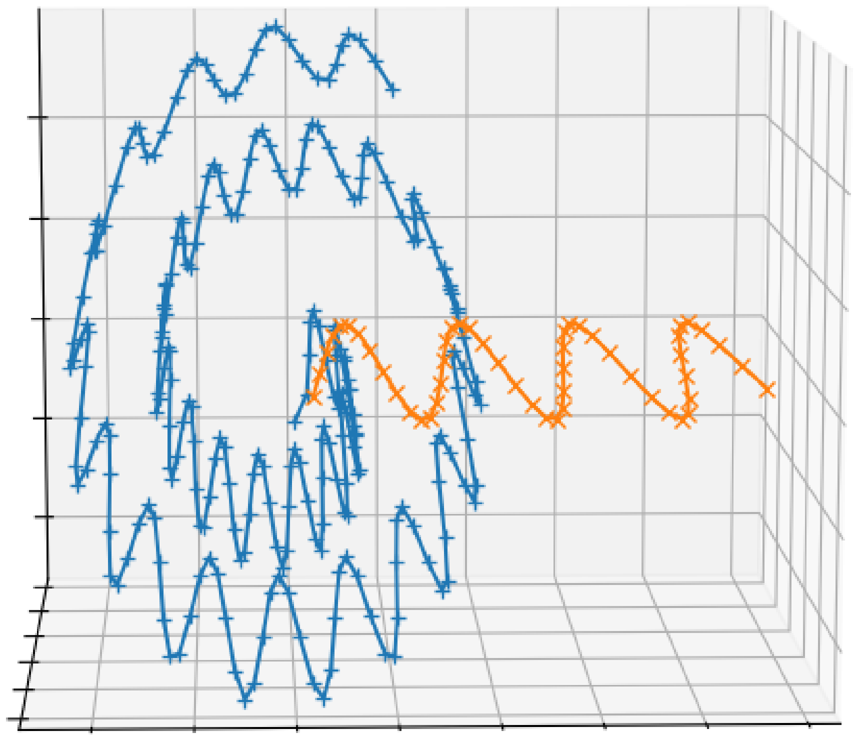


Figure 6.3: Sinusoidal curves on a plane and on a Swiss roll. Aligning these curves requires a nonlinear transformation.

6.2 Manifold Warping

6.2.1 One-step algorithm

We now present a novel framework for aligning two sequentially-ordered data sets that share a common manifold representation. In our approach, we use the warping matrix produced by DTW as a heuristic correspondence matrix for manifold alignment. The proposed algorithm uses the method of alternating projections [90], picking new correspondences with DTW and reprojecting both inputs using manifold alignment until the loss function converges. This presents an improvement over canonical time warping (CTW) in cases where nonlinear transformations are required to recover the underlying manifold structure of one or both input data sets.

We introduce the following loss function for manifold warping:

$$\begin{aligned}
&L_3\left(F^{(X)}, F^{(Y)}, W^{(X,Y)}\right) = \\
&\mu \sum_{i \in X, j \in Y} \|F_i^{(X)} - F_j^{(Y)}\|^2 W_{i,j}^{(X,Y)} \\
&+(1-\mu) \sum_{i,j \in X} \|F_i^{(X)} - F_j^{(X)}\|^2 W_{i,j}^{(X)} \\
&+(1-\mu) \sum_{i,j \in Y} \|F_i^{(Y)} - F_j^{(Y)}\|^2 W_{i,j}^{(Y)}
\end{aligned} \tag{6.12}$$

The optimization problem becomes:

$$\begin{array}{ll}
\underset{F^{(X)}, F^{(Y)}, W^{(X,Y)}}{\text{argmin}} & (L_3) \\
\text{subject to} & F^\top D F = I \\
\text{where} & F = \begin{bmatrix} F^{(X)} \\ F^{(Y)} \end{bmatrix}
\end{array}$$

Unlike manifold alignment, the correspondence matrix $W^{(X,Y)}$ is now an argument to the optimization problem. The intuition behind this loss function resembles that of manifold alignment: the last two error terms ensure that the embedding preserves the local geometry of the inputs while the first term promotes a high quality DTW alignment between two sequential data sets. Again, these goals are controlled by the parameter μ. The procedure for minimizing L_3 is given in Algorithm 6.

Algorithm 6 One-Step Manifold Warping

```
  Input: Two time-series data sets, $X, Y$
    Latent space dimension, $d$
    Preserving correspondence vs local geometry factor, $\mu$
  Output: Embeddings in the latent space, $F^{(X)}, F^{(Y)}$
    DTW matrix providing the alignment of X and Y, $W^{(X,Y)}$
  $W^{(X)} \leftarrow$ NeighborGraph$(X)$
  $W^{(Y)} \leftarrow$ NeighborGraph$(Y)$
  Set $W^{(X,Y)}_{1,1} = W^{(X,Y)}_{n_X,n_Y} = 1$, and $0$ everywhere else
  $t \leftarrow 0$
  while not converged do
    $W = \begin{bmatrix} (1-\mu)W^{(X),t} & \mu W^{(X,Y),t} \\ \mu(W^{(X,Y),t})^\top & (1-\mu)W^{(Y),t} \end{bmatrix}$
    $F^{(X),t+1}, F^{(Y),t+1} \leftarrow$ MA$(F^{(X),t}, F^{(Y),t}, W^t, d, \mu)$
    $W^{(X,Y),t+1} \leftarrow$ DTW$(F^{(X),t+1}, F^{(Y),t+1})$
    $t \leftarrow t + 1$
  end while
  $F^{(X)} \leftarrow F^{(X),t}$
  $F^{(Y)} \leftarrow F^{(Y),t}$
  $W^{(X,Y)} \leftarrow W^t$
```

In Algorithm 6, MA(X, Y, W, d, μ) is a function that returns the embedding of X, Y in a d dimensional space using manifold alignment with the joint similarity matrix W and parameter μ, as described in the manifold alignment section. The function DTW(X, Y) returns a DTW matrix after aligning two sequences X, Y using dynamic time warping. The NeighborGraph(X) function returns the adjacency matrix of a manifold approximation graph, such as a k-nearest or ϵ-close neighbor graph.

Theorem 1. *Let $L_{3,t}$ be the loss function L_3 evaluated at $F^{(X),t}, F^{(Y),t}, W^{(X,Y),t}$ of Algorithm 6. The sequence $L_{3,t}$ converges to a critical point as t $\to \infty$. Therefore, Algorithm 6 will terminate.*

Proof. In every iteration t, two steps are performed: manifold alignment to solve for new projections $F^{(X),t+1}, F^{(Y),t+1}$, and DTW to change the correspondences to $W^{(X,Y),t+1}$.

In the first step, with fixed $W^{(X,Y),t}$, Algorithm 6 solves for new projections $F^{(X),t+1}, F^{(Y),t+1}$ using manifold alignment. Recall that L_3 with a fixed $W^{(X,Y)}$ is equivalent to the loss function L_1, which we showed is minimized by manifold alignment's mappings in Section 6.1.1. Hence, when the correspondence matrix is fixed,

$$L_3(F^{(X),t+1}, F^{(Y),t+1}, W^{(X,Y),t}) \leq L_3(F^{(X),t}, F^{(Y),t}, W^{(X,Y),t}). \tag{6.13}$$

In the second step, the projections are fixed as $F^{(X),t+1}, F^{(Y),t+1}$. Algorithm 6 changes the correspondence matrix from $W^{(X,Y),t}$ to $W^{(X,Y),t+1}$, which does not affect the last two terms in L_3. If we replace $dist(F_i^{(X)}, F_j^{(Y)})$ by $\mu\|F_i^{(X),t+1} - F_j^{(Y),t+1}\|^2$ in the loss function L_2 of DTW, we recover the first term in L_3 of manifold warping. Because $W^{(X,Y),t+1}$ is produced by DTW, it will minimize the first term of L_3. Therefore, we have

$$\begin{aligned}&\mu \sum_{i\in X, j\in Y} \|F_i^{(X),t+1} - F_j^{(Y),t+1}\|^2 W_{i,j}^{(X,Y),t+1}\\ &\leq \mu \sum_{i\in X, j\in Y} \|F_i^{(X),t+1} - F_j^{(Y),t+1}\|^2 W_{i,j}^{(X,Y),t}.\end{aligned} \tag{6.14}$$

Changing the correspondence matrix does not affect the last two terms of L_3, so

$$\begin{aligned}&L_3(F^{(X),t+1}, F^{(Y),t+1}, W^{(X,Y),t+1})\\ &\leq L_3(F^{(X),t+1}, F^{(Y),t+1}, W^{(X,Y),t})\\ &\leq L_3(F^{(X),t}, F^{(Y),t}, W^{(X,Y),t}) \text{ from inequality 6.13}\\ &\Leftrightarrow L_{3,t+1} \leq L_{3,t}.\end{aligned} \tag{6.15}$$

Therefore, $L_{3,t}$ is a strictly non-increasing sequence. We also have $L_{3,t} \geq 0$, so it is convergent. Therefore, Algorithm 6 will eventually terminate. □

6.2.2 Two-step algorithm

We now propose an algorithm that exploits the observation that when the local geometries of the two data sets are roughly the same, their similarity matrices will also be very similar [92]. Thus, if we first apply a nonlinear projection to each input set independently, the embeddings are likely to be linearly alignable using either manifold warping or CTW.

Algorithm 7 Two-Step Manifold Warping

```
**Input:** Two time-series data sets, $X, Y$
    Latent space dimension, $d$
    Preserving correspondence vs local geometry factor, $\mu$
**Output:** Embeddings in the latent space, $F^{(X)}, F^{(Y)}$
    DTW matrix providing the alignment of X and Y, $W^{(X,Y)}$
$W^{(X)} \leftarrow \text{NeighborGraph}(X)$
$W^{(Y)} \leftarrow \text{NeighborGraph}(Y)$
$t \leftarrow 0$
$F^{(X),t} \leftarrow \text{DimReduction}\left(F^{(X)}, W^{(X)}, d\right)$
$F^{(Y),t} \leftarrow \text{DimReduction}\left(F^{(Y)}, W^{(Y)}, d\right)$
**while** not converged **do**
    $W = \begin{bmatrix} (1-\mu)W^{(X),t} & \mu W^{(X,Y),t} \\ \mu(W^{(X,Y),t})^\top & (1-\mu)W^{(Y),t} \end{bmatrix}$
    $\phi^{(Y),t+1}, \phi^{(X),t+1} \leftarrow \text{LMA}\left(F^{(X),t}, F^{(Y),t}, W^{(X,Y),t}, d, \mu\right)$
    $F^{(X),t+1} \leftarrow F^{(X),t}\phi^{(X),t+1}$
    $F^{(Y),t+1} \leftarrow F^{(Y),t}\phi^{(Y),t+1}$
    $W^{(X,Y),t+1} \leftarrow \text{DTW}\left(F^{(X),t+1}, F^{(Y),t+1}\right)$
    $t \leftarrow t+1$
**end while**
$F^{(X)} \leftarrow F^{(X),t}$
$F^{(Y)} \leftarrow F^{(Y),t}$
$W^{(X,Y)} \leftarrow W^{(X,Y),t}$
```

In Algorithm 7, DimReduction(X, W, d) is a dimensionality reduction function that maps X with similarity matrix W to a lower dimensional space d. We use Laplacian Eigenmaps to be consistent with manifold alignment, though other methods such as LLE or Isomap could be applied.

LMA(X, Y, W, d, μ) is a function that performs linear manifold alignment described above on X and Y with a joint similarity matrix W and target dimension d, which returns the projection matrices $\phi^{(X)}$ and $\phi^{(Y)}$. We can think of DimReduction(X, W, d) as a preprocessing step, then reformulate the loss function as

$$\begin{aligned}
&L_4(\phi^{(X)}, \phi^{(Y)}, W^{(X,Y)}) \\
&= ((1-\mu)\sum_{i,j\in X} \|F_i^{(X)}\phi^{(X)} - F_j^{(X)}\phi^{(X)}\|^2 W_{ij}^{(X)} \\
&+(1-\mu)\sum_{i,j\in Y} \|F_i^{(Y)}\phi^{(Y)} - F_j^{(Y)}\phi^{(Y)}\|^2 W_{ij}^{(Y)} \\
&+\mu \sum_{i\in X, j\in Y} \|F_i^{(X)}\phi^{(X)} - F_j^{(Y)}\phi^{(Y)}\|^2 W_{ij}^{(X,Y)})
\end{aligned}, \tag{6.16}$$

which is the same loss function as in linear manifold alignment except that $W^{(X,Y)}$ is now a variable. The two constraints are the constraint in Equation 6.6 of linear manifold alignment, and $W^{(X,Y)}$ must be a DTW matrix.

Theorem 2. *Let $L_{4,t}$ be the loss function L_4 evaluated at $\prod_{i=1}^t \phi^{(X),i}, \prod_{i=1}^t \phi^{(Y),i}, W^{(X,Y),t}$ of Algorithm 7. The sequence $L_{4,t}$ converges to a critical point as $t \to \infty$. Therefore, Algorithm 7 will terminate.*

Proof. The proof is similar to that of Theorem 1. At any iteration t, Algorithm 7 first fixes the correspondence matrix at $W^{(X,Y),t}$. Now let L_4' be like L_4 except that we replace $F_i^{(X)}, F_i^{(Y)}$ by $F_i^{(X),t}, F_i^{(Y),t}$ and Algorithm 7 minimizes L_4' over $\phi^{(X),t+1}, \phi^{(Y),t+1}$ using linear manifold alignment. Thus,

$$\begin{aligned}
&L_4'(\phi^{(X),t+1}, \phi^{(Y),t+1}, W^{(X,Y),t}) \\
&\leq L_4'(I, I, W^{(X,Y),t}) \\
&= L_4(\textstyle\prod_{i=1}^t \phi^{(X),i}, \prod_{i=1}^t \phi^{(Y),i}, W^{(X,Y),t}) \\
&= L_{4,t}
\end{aligned} \tag{6.17}$$

because $F^{(X),t} = F^{(X)} \prod_{i=1}^t \phi^{(X),i}$ and $F^{(Y),t} = F^{(Y)} \prod_{i=1}^t \phi^{(X),i}$. We also have:

$$\begin{aligned}&L'_4(\phi^{(X),t+1}, \phi^{(Y),t+1}, W^{(X,Y),t})\\&= L_4(\textstyle\prod_{i=1}^{t+1}\phi^{(X),i}, \prod_{i=1}^{t+1}\phi^{(Y),i}, W^{(X,Y),t})\\&\leq L_{4,t}.\end{aligned} \tag{6.18}$$

Algorithm 7 then performs DTW to change $W^{(X,Y),t}$ to $W^{(X,Y),t+1}$. Using the same argument as in the proof of Theorem 1, we have:

$$\begin{aligned}&L_4(\textstyle\prod_{i=1}^{t+1}\phi^{(X),i}, \prod_{i=1}^{t+1}\phi^{(Y),i}, W^{(X,Y),t+1})\\&\leq L_4(\textstyle\prod_{i=1}^{t+1}\phi^{(X),i}, \prod_{i=1}^{t+1}\phi^{(Y),i}, W^{(X,Y),t})\\&\leq L_{4,t}\\&\Leftrightarrow L_{4,t+1} \leq L_{4,t}.\end{aligned} \tag{6.19}$$

The convergence follows. □

Furthermore, when we set $\mu = 1$, the loss function L_4 will resemble CTW's loss function. We can also substitute CTW in place of the loop in the algorithm.

6.3 Experimental Results

6.3.1 Synthetic data sets

We compare the performance of CTW and manifold warping by trying to align two $\sin(x^2)$ curves: one is on the flat plane, the another is projected onto the Swiss roll as illustrated in Figure 6.3. Some duplicate points are added along the curves to create many-to-one correspondences in the alignment.

As shown in Figure 6.4, manifold warping produced similar embeddings for two curves based on their local geometry while CTW linearly collapsed the Swiss roll curve onto the plane.

As a result, the warping path (that is, the alignment path) produced by manifold warping stays closer to the true warping path than the warping path produced by CTW. The error

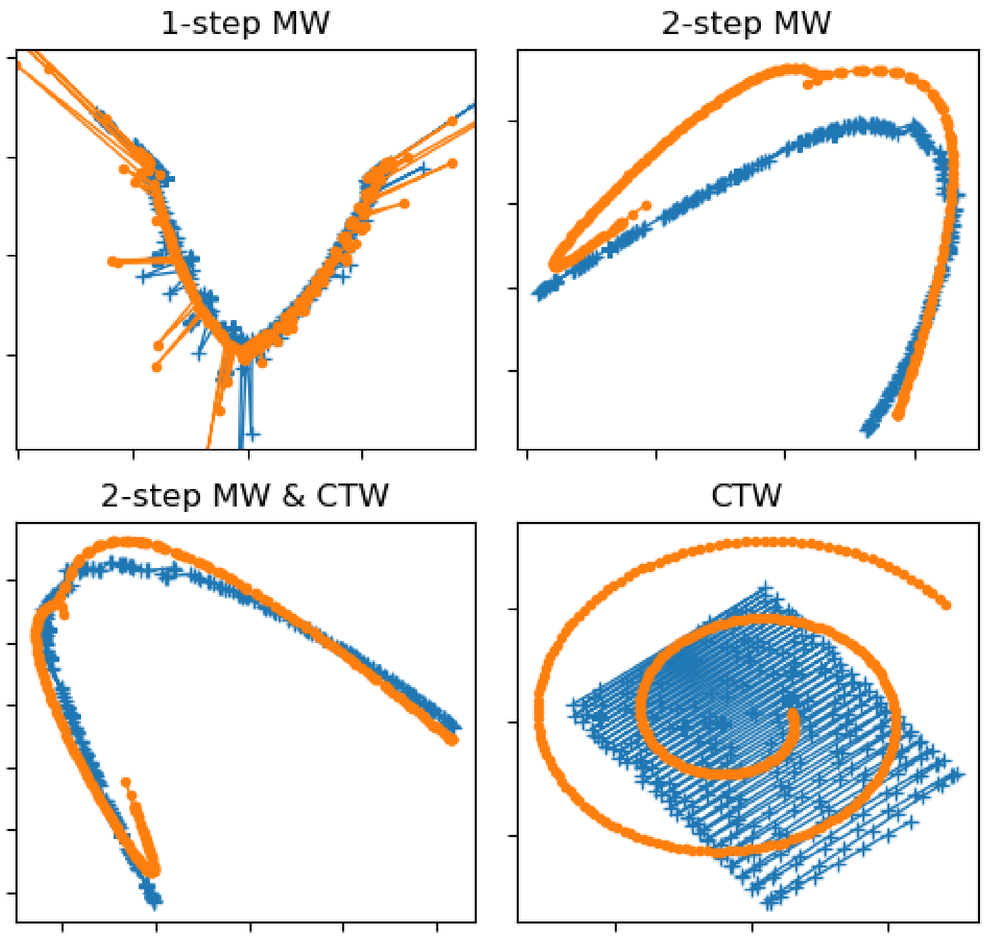


Figure 6.4: Two-dimensional embeddings of the nonlinearly warped $\sin(x^2)$ curves illustrated in Figure 6.3.

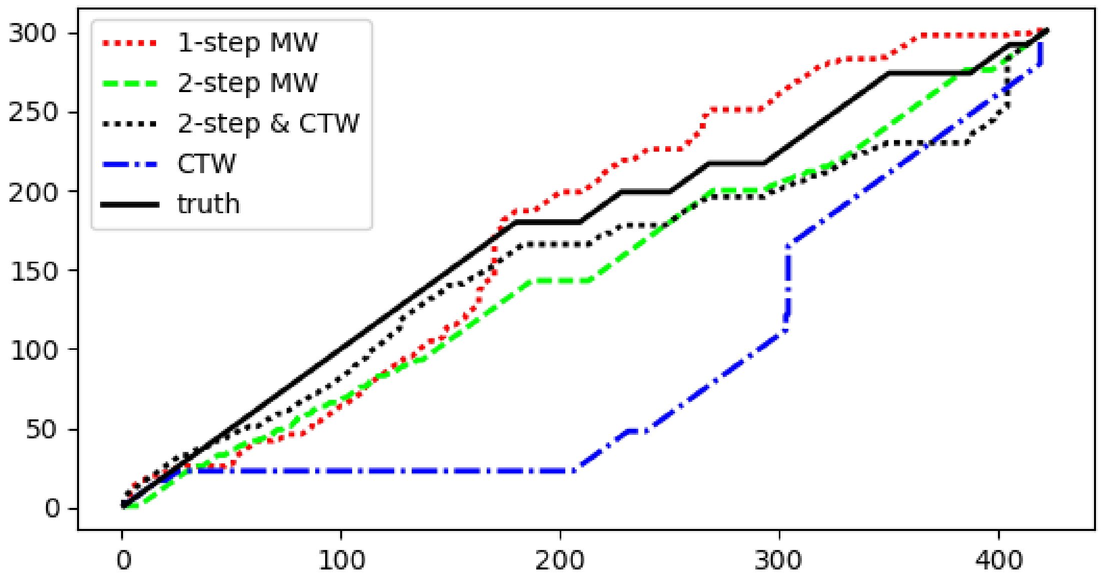


Figure 6.5: The resulting warping path of each algorithm's alignment and the ground truth warping path for the warped $\sin(x^2)$ curves in Figure 6.3.

is calculated as the area between the warping path and the ground truth path, as suggested in the original paper on CTW [112]. To enable comparison, we normalize the error by dividing by the plot area, $n_X \times n_Y$.

The warping paths and the calculated errors, shown in Figure 6.5 and Table 6.1, demonstrate that manifold warping yields a smaller error than CTW.

6.3.2 COIL-100 data set

We also test these algorithms on a real-world vision data set from the Columbia Object Image Library (COIL-100) [69]. The corpus consists of different series of images taken of different objects on a rotating platform. Each series has 72 images, each 128×128 pixels. We try to align two series of images of two different objects, with differences in shape and brightness producing very different high-dimensional representations. To demonstrate our algorithm's ability to work with data sets of different dimensionality, we compress one

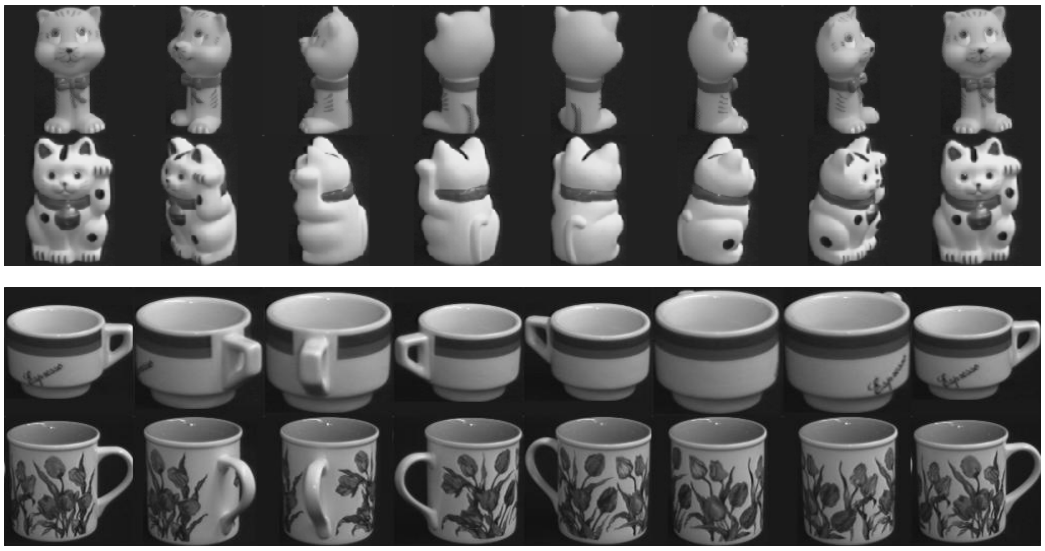


Figure 6.6: Samples from pairs of COIL-100 image series

	Synthetic	Dog+Cat	Cups	Kitchen
1-step MW	0.0768	0.0447	0.0464	**0.0257**
2-step MW	0.0817	**0.0282**	**0.0125**	0.0396
2-step + CTW	**0.0652**	0.0298	0.0143	0.0772
CTW	0.2784	0.2656	0.1668	0.0966

Table 6.1: Alignment error across algorithms and data sets.

image series to a smaller resolution (64×64 pixels). Additionally, some duplicate images are added to each series, to ensure that the correct mapping is not trivially one-to-one.

In both experiments, manifold warping methods create alignments with a much smaller error than CTW. The depiction in Figure 6.7 provides an intuitive picture of the manifold warping algorithm. In the first projection to two dimensions, both image series are mapped to circles. The next several iterations rotate these circles to match the first and last points, then the points in between. For the case of one-step Manifold Warping (where all mappings are nonlinear), we pick a small μ to prioritize preserving local geometry of each series. This avoids over-fitting the embedding to a potentially bad intermediate DTW correspondence.

We perform the experiment with two pairs of COIL image series, illustrated in Figure 6.6.

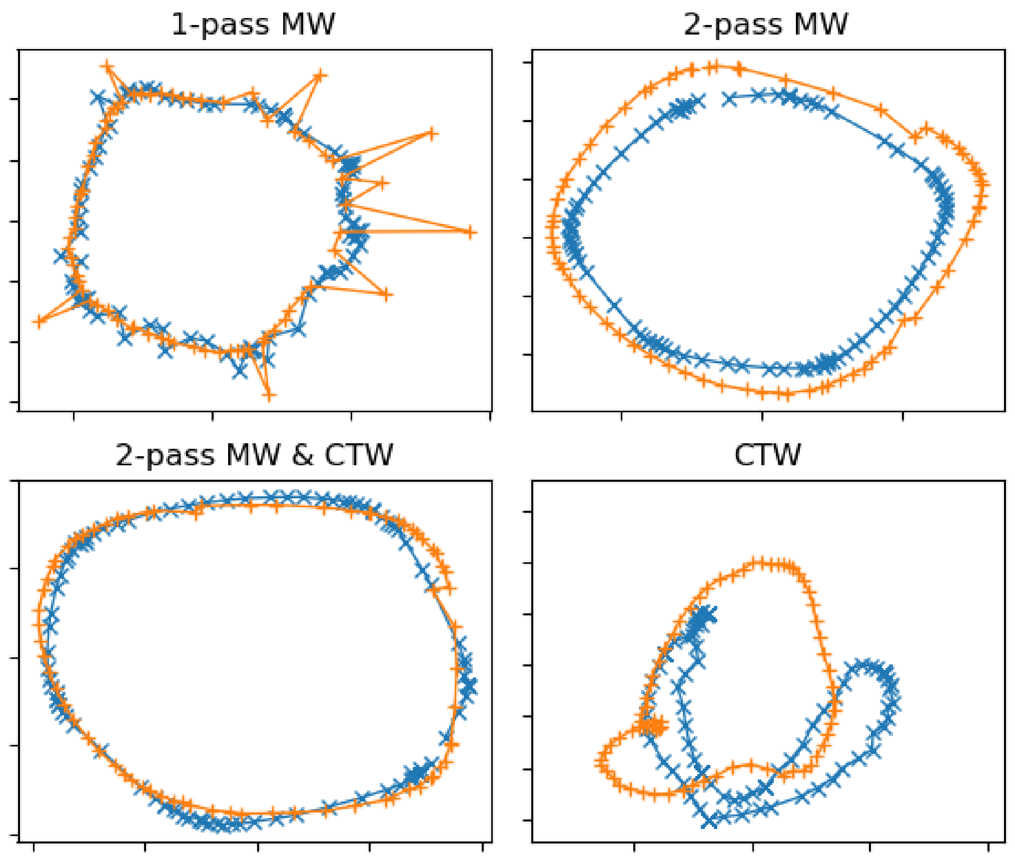


Figure 6.7: Two-dimensional embedding of the dog and cat toy image series (Figure 6.6).

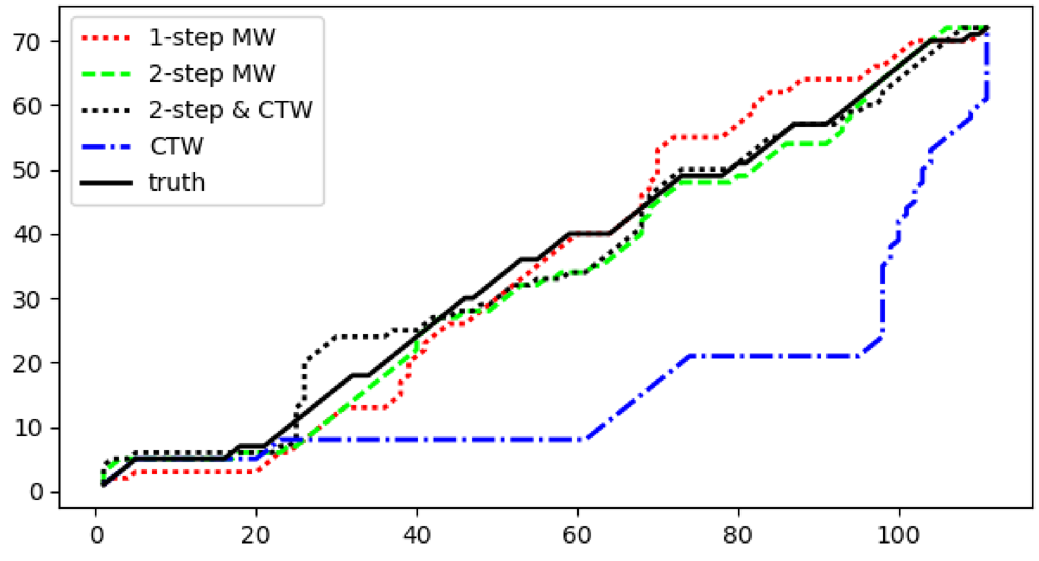


Figure 6.8: Warping paths for the dog and cat toy image series (Figure 6.6).

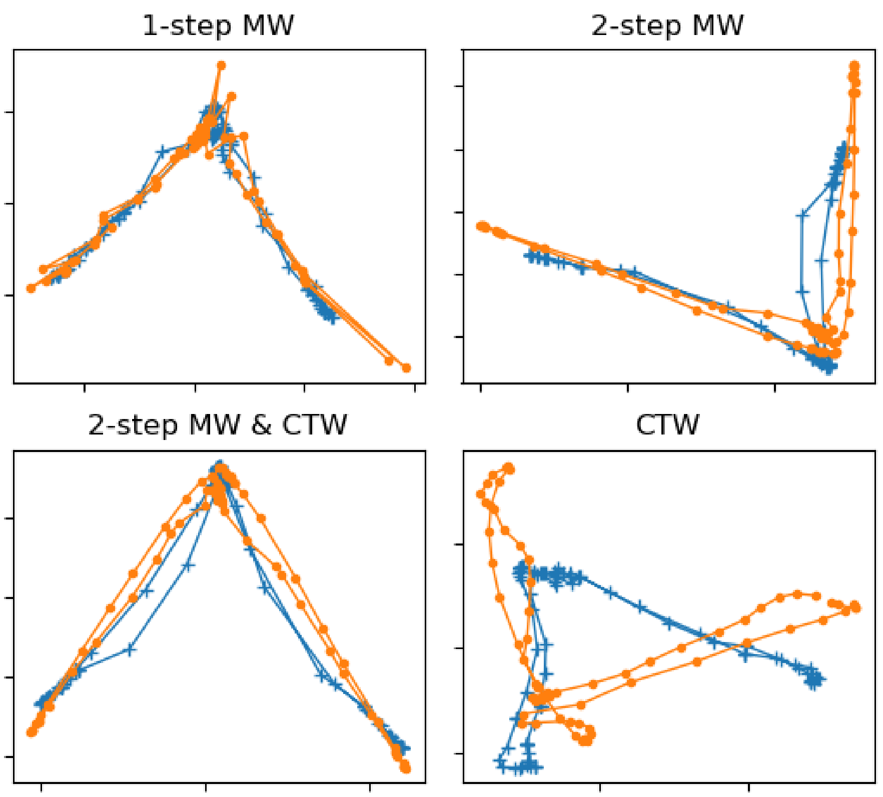


Figure 6.9: Two-dimensional embedding of the rotating cup image series (Figure 6.6).

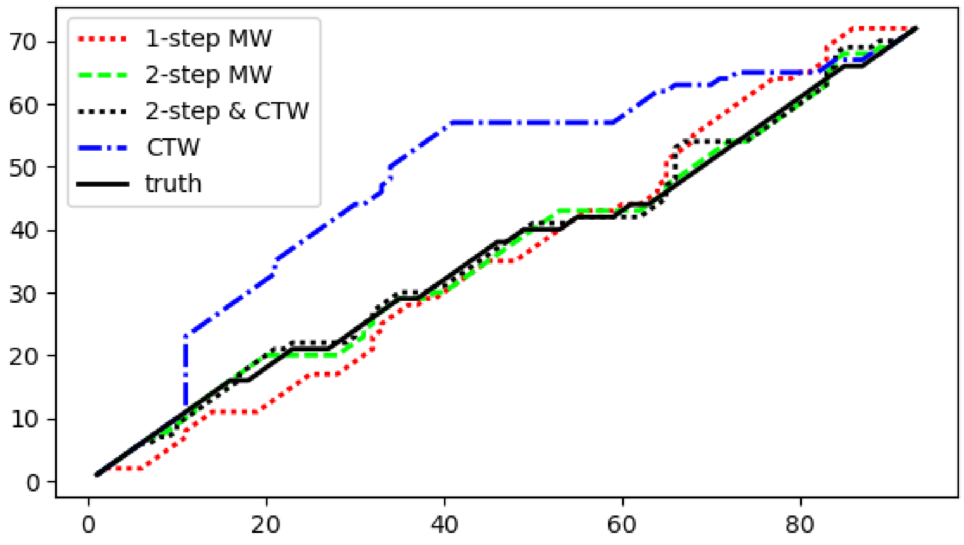


Figure 6.10: Warping paths for the rotating cup image series (Figure 6.6).

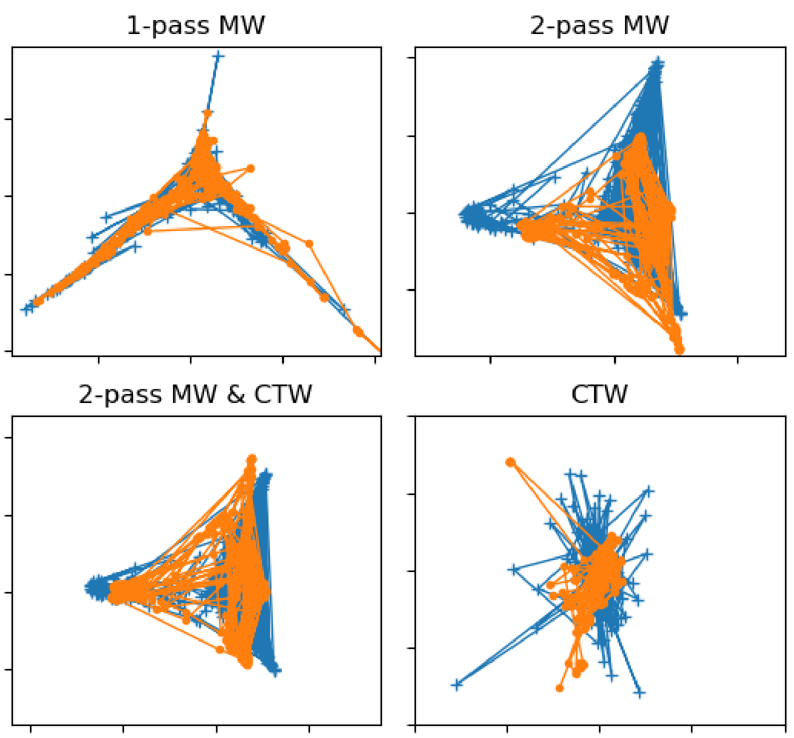


Figure 6.11: Two-dimensional embeddings of the measurement series of IMU and MOCAP sensors during the CMU kitchen task.

6.3.3 Kitchen data set

Our last experiment uses the kitchen data set from the CMU Quality of Life Grand Challenge [18], which records human subjects cooking a variety of dishes. Here, we attempt nonlinear alignments between the same subject and task across different sensors.

Our experiment considers two separate views of the same moment in time, during which the subject prepares a brownie. The two views are 9-dimensional inertial measurement unit (IMU) readings and 87-dimensional motion capture suit coordinates (MOCAP). Aligning two views of the same task provides a straightforward evaluation metric, because the time stamps on each reading yield ground-truth correspondence information. To make the problem computationally feasible, we subsampled the original data sets. Each manifold warping method performs better than CTW, based on the results shown below in Figure 6.11, Figure 6.12, and Table 6.1.

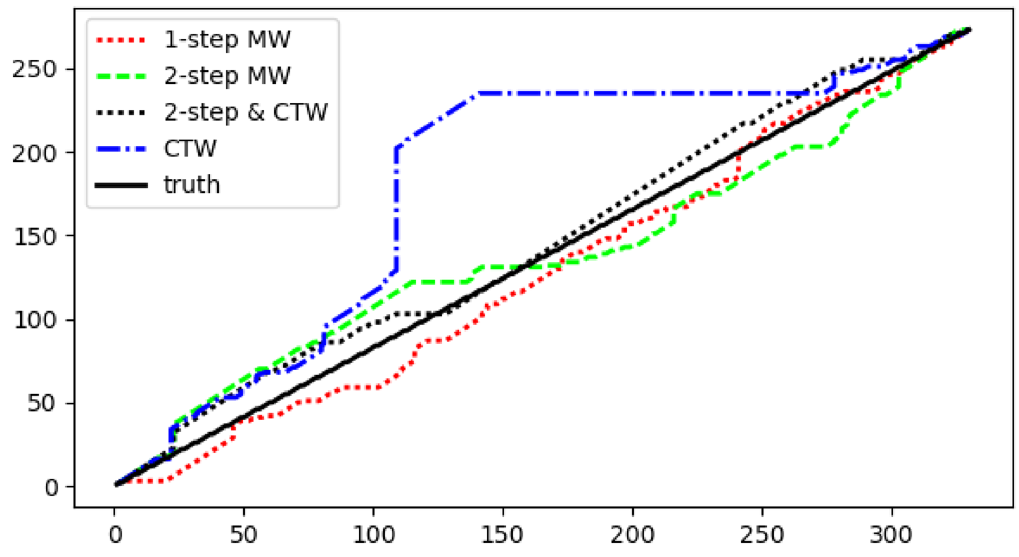


Figure 6.12: Warping paths for each algorithm's alignment of the sensor data from the CMU kitchen task.

6.4 Discussion

Due to the lack of a linearity constraint, manifold warping consistently performs better than canonical time warping when the inputs lie on manifolds that are not accessible via linear transformations. Even in the linear case, the alignment quality of manifold warping is equivalent to that of CTW.

Importantly, this improved alignment quality does not impose significant runtime overhead. Both algorithms rely on the same DTW step, and tuned implementations of manifold alignment are comparable in running time to the CCA step used in CTW. While each manifold warping iteration is marginally slower than a similar CTW iteration, manifold warping tends to converge with fewer steps.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Summary

Manifold discovery is the foundation upon which all effective manifold learning algorithms are built. Without an accurate and reliable representation of the latent manifold structures in the dataset of interest, even the most sophisticated graph-based manifold learning methods will fail.

The task of learning this manifold structure is often relegated to neighborhood graph construction methods like k-nearest neighbors, which, while simple and efficient to implement, require an expert's oversight to avoid choosing bad hyperparameters. This problem is especially pernicious in applications with high-dimensional, real-world data sets, for which global structure is rarely known a priori.

In this thesis, we proposed new methods for constructing high-quality manifold approximation graphs that reduce the burden of parameter tuning and result in improved outcomes when used in manifold learning tasks.

First, we introduced new ways to measure the manifold-approximation quality of a graph, independent of a manifold learning task. We observed the way a graph's global geodesic structure changes as edges are added, then proposed an efficient incremental graph construction algorithm that provides the means to generate these graph quality signals concurrently with graph construction. This combination of methods provides a mechanism for making informed parameter settings before the graph is used for any manifold learning task applications, simplifying the manifold learning process considerably.

Next, we proposed the Manifold Spanning Graph, a graph construction algorithm that connects local tangent subspaces with edges that preserve both locality and linearity. This novel algorithm is designed to be parameter-insensitive, and requires little to no tuning to produce high quality graphs.

The Mutual Neighbors Graph algorithm makes use of a cost function that guides the addition of new edges to an initial sparse subgraph. The costs encode multiple desired quality measures, incorporating standard manifold learning assumptions directly into the process of graph construction. This algorithm exploits information about known edges on the manifold to produce effective graphs, without relying on expert-tuned parameters.

Finally, we introduce Manifold Warping, a manifold alignment method for sequential datasets that relies on manifold approximation graphs to balance intra-dataset structure preservation against inter-dataset correspondences. The use of manifold learning methods enables both linear and nonlinear alignment strategies, offering flexibility without substantially increasing algorithm complexity.

7.2 Caveats

The success of manifold learning methods on a particular dataset is primarily reliant on the degree to which the data conforms to the manifold hypothesis. For example, a dataset sampled uniformly from $\mathbb{R}^n$ will not benefit from manifold learning approaches over traditional machine learning, no matter how sophisticated. In general, the more closely a dataset follows the key assumptions of manifold structure (i.e., local linearity and smoothness), the more effective manifold discovery and learning will be.

In addition to considerations of manifold structure, dataset size is also an important factor when considering the use of manifold learning methods. If each instance in a dataset is used as a graph vertex, the process of adding edges to learn the graph is usually at least quadratic in the number of vertices. Furthermore, most popular graph-based manifold learning methods require super-linear running time relative to the size of the dataset.

Methods have been proposed to choose anchor points on which to build graphs representing large datasets [81, 61], but in general, manifold learning methods are not especially well-suited to large, out-of-core learning problems.

7.3 Future Work

This thesis leaves several directions open to future study in the pursuit of effective algorithms for graph-based manifold discovery.

7.3.1 Directed Graph Construction

The methods proposed in this thesis have each assumed that any edge connecting two vertices represents a symmetric relationship. For example, the Euclidean distance between two points a and b is the same whether one measures from a to b or from b to a.

This property does not hold in all domains, however. In datasets with a temporal component, relationships may only exist for pairs of points which respect the time ordering. This condition is especially relevant in reinforcement learning domains, for which the actions that cause transitions between states in the environment are often non-reversible [45].

To discovery the manifolds representing datasets with asymmetric relationships, it will be necessary to develop construction algorithms for *directed graphs*.

7.3.2 Nonlinear Metrics

Each of the graph construction methods in this thesis contain some notion of a distance metric that is used to quantify how close pairs of points are. The straight-line Euclidean distance metric is an appropriate choice in most cases, especially when the features of the dataset are simple vectors in the input space.

There are datasets for which Euclidean distances are wholly inappropriate, however, and in these cases an alternate distance metric must be used. These datasets include image corpora, for which correlation measures are often more informative, or spectroscopy databases, where spectral angle similarity is commonly used to determine the relationship between instances.

Future work on specialized graph construction methods for non-Euclidean distances has great potential for improving manifold discovery outcomes. Simple neighbor-based methods can be agnostic to the way distances are calculated, but more sophisticated algorithms may be able to exploit this information effectively.

7.3.3 Active Manifold Discovery

The traditional model of manifold learning assumes that all available samples from the input dataset are available at the outset of manifold discovery. This is the paradigm that the graph construction methods in this thesis follow, with no additional vertices added or removed from the graph after initialization.

Many real-world application settings do not permit this static view of a dataset, however. In some reinforcement learning domains, a full characterization of the environment is not available until the agent discovers it via exploration. In the active learning paradigm, new input points come at a cost, so learning cannot wait until a sufficiently large dataset has been collected.

This limitation provides an interesting avenue for future research in manifold discovery. Existing graph construction algorithms will need to be modified to model the uncertainty inherent in working with partially-observed datasets. These algorithms should also permit the addition of new data, either as new vertices in the graph or as additional evidence for or against existing edge connections.

BIBLIOGRAPHY

[1] Anderson, T.W. *An introduction to multivariate statistical analysis*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. Wiley-Interscience, 2003.

[2] Ausiello, Giorgio, Italiano, Giuseppe F, Spaccamela, Alberto Marchetti, and Nanni, Umberto. On-line computation of minimal and maximal length paths. *Theoretical Computer Science 95*, 2 (1992), 245–261.

[3] Balasubramanian, Mukund, and Schwartz, Eric L. The isomap algorithm and topological stability. *Science 295*, 5552 (2002), 7–7.

[4] Belkin, M., and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems 14* (2001), 585–591.

[5] Belkin, Mikhail, Niyogi, Partha, and Sindhwani, Vikas. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research 7* (2006), 2399–2434.

[6] Bengio, Yoshua, and Monperrus, Martin. Non-local manifold tangent learning. *Advances in Neural Information Processing Systems 17* (2005), 129–136.

[7] Berge, Claude, and Minieka, Edward. *Graphs and hypergraphs*, vol. 7. North-Holland publishing company Amsterdam, 1973.

[8] Bishop, Christopher M, Svensén, Markus, and Williams, Christopher KI. Gtm: The generative topographic mapping. *Neural computation 10*, 1 (1998), 215–234.

[9] Boucher, Thomas, Carey, CJ, Mahadevan, Sridhar, and Dyar, M Darby. Aligning mixed manifolds. *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

[10] Bradley, Paul S, and Mangasarian, Olvi L. k-plane clustering. *Journal of Global Optimization 16*, 1 (2000), 23–32.

[11] Brand, Matthew. Charting a manifold. In *Advances in neural information processing systems* (2002), pp. 961–968.

[12] Brandes, Ulrik. A faster algorithm for betweenness centrality*. *Journal of Mathematical Sociology 25*, 2 (2001), 163–177.

[13] Carreira-Perpinán, Miguel A, and Wang, Weiran. Lass: A simple assignment model with laplacian smoothing. *arXiv preprint arXiv:1405.5960* (2014).

[14] Cayton, Lawrence. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep* (2005), 1–17.

[15] Chapelle, O, Schölkopf, B, and Zien, A. Semi-supervised learning.

[16] Coifman, Ronald R, and Lafon, Stéphane. Diffusion maps. *Appl. Comput. Harmon. Anal 21* (2006), 5–30.

[17] Davis, Jason V, Kulis, Brian, Jain, Prateek, Sra, Suvrit, and Dhillon, Inderjit S. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 209–216.

[18] De la Torre, F., Hodgins, J., Bargteil, A., Martin, X., Macey, J., Collado, A., and Beltran, P. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database.

[19] de Sousa, Celso André R, Rezende, Solange O, and Batista, Gustavo EAPA. Influence of graph construction on semi-supervised learning. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 160–175.

[20] Deleforge, Antoine, and Horaud, Radu. 2d sound-source localization on the binaural manifold. In *Machine Learning for Signal Processing* (2012), IEEE, pp. 1–6.

[21] Demartines, Pierre, and Hérault, Jeanny. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *Neural Networks, IEEE Transactions on 8*, 1 (1997), 148–154.

[22] Demetrescu, Camil, Finocchi, Irene, and Italiano, G. Handbook on data structures and applications, chapter 36: Dynamic graphs. *CRC Press 6*, 7 (2005), 8.

[23] Demetrescu, Camil, and Italiano, Giuseppe F. A new approach to dynamic all pairs shortest paths. *Journal of the ACM (JACM) 51*, 6 (2004), 968–992.

[24] Dijkstra, Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

[25] Dong, Wei, Moses, Charikar, and Li, Kai. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web* (2011), ACM, pp. 577–586.

[26] Donoho, David L, and Grimes, Carrie. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences 100*, 10 (2003), 5591–5596.

[27] Elhamifar, Ehsan, and Vidal, René. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 2790–2797.

[28] Elhamifar, Ehsan, and Vidal, René. Sparse manifold clustering and embedding. In *Advances in neural information processing systems* (2011), pp. 55–63.

[29] Floyd, Robert W. Algorithm 97: shortest path. *Communications of the ACM 5*, 6 (1962), 345.

[30] Fredman, Michael L, and Tarjan, Robert Endre. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM) 34*, 3 (1987), 596–615.

[31] Freeman, Linton C. A set of measures of centrality based on betweenness. *Sociometry* (1977).

[32] Frey, Brendan J, and Dueck, Delbert. Clustering by passing messages between data points. *science 315*, 5814 (2007), 972–976.

[33] Gashler, Michael, Ventura, Dan, and Martinez, Tony. Manifold learning by graduated optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41*, 6 (2011), 1458–1470.

[34] Gashler, Mike, and Martinez, Tony. Tangent space guided intelligent neighbor finding. In *Neural Networks (IJCNN), The 2011 International Joint Conference on* (2011), IEEE, pp. 2617–2624.

[35] Gashler, Mike, and Martinez, Tony. Robust manifold learning with cyclecut. *Connection Science 24*, 1 (2012), 57–69.

[36] Ghazvininejad, Marjan, Mahdieh, Mostafa, Rabiee, Hamid R, Roshan, Parisa Khanipour, and Rohban, Mohammad Hossein. Isograph: Neighbourhood graph construction based on geodesic distance for semi-supervised learning. In *2011 IEEE 11th International Conference on Data Mining* (2011), IEEE, pp. 191–200.

[37] Gruber, Peter, and Theis, Fabian J. Grassmann clustering. *Proc. EUSIPCO 2006* (2006).

[38] Hadid, Abdenour, and Pietikäinen, Matti. Efficient locally linear embeddings of imperfect manifolds. In *Machine learning and data mining in pattern recognition.* Springer, 2003, pp. 188–201.

[39] Ham, Jihun, Lee, Daniel, and Saul, Lawrence. Semisupervised alignment of manifolds. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence* (2005), vol. 10, pp. 120–127.

[40] He, Xiaofei, Yan, Shuicheng, Hu, Yuxiao, Niyogi, Partha, and Zhang, Hong-Jiang. Face recognition using laplacianfaces. *IEEE transactions on pattern analysis and machine intelligence 27*, 3 (2005), 328–340.

[41] Hinton, Geoffrey E, Dayan, Peter, and Revow, Michael. Modeling the manifolds of images of handwritten digits. *Neural Networks, IEEE Transactions on 8*, 1 (1997), 65–74.

[42] Hotelling, H. Relations between two sets of variates. *Biometrika 28* (1936), 321–377.

[43] Huang, Bert, and Jebara, Tony. Maximum likelihood graph structure estimation with degree distributions. In *Analyzing Graphs: Theory and Applications, NIPS Workshop* (2008), vol. 14.

[44] Hubert, Lawrence, and Arabie, Phipps. Comparing partitions. *Journal of classification 2*, 1 (1985), 193–218.

[45] Johns, Jeff, and Mahadevan, Sridhar. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 385–392.

[46] Jolliffe, I.T. *Principal component analysis*. Springer series in statistics. Springer-Verlag, 2002.

[47] Kambhatla, Nandakishore, and Leen, Todd K. Dimension reduction by local principal component analysis. *Neural Computation 9*, 7 (1997), 1493–1516.

[48] Kégl, Balázs. Intrinsic dimension estimation using packing numbers. In *Advances in neural information processing systems* (2002), pp. 681–688.

[49] Kohonen, Teuvo. The self-organizing map. *Proceedings of the IEEE 78*, 9 (1990), 1464–1480.

[50] Konidaris, George, Osentoski, Sarah, and Thomas, Philip. Value function approximation in reinforcement learning using the fourier basis. In *Twenty-Fifth AAAI Conference on Artificial Intelligence* (2011).

[51] Kruskal, Joseph B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society 7*, 1 (1956), 48–50.

[52] Kruskal, Joseph B., and Wish, Myron. *Multidimensional scaling*, vol. 11. Sage, 1978.

[53] LeCun, Yann, and Cortes, Corinna. The mnist database of handwritten digits, 1998.

[54] Lee, John Aldo, Verleysen, Michel, et al. How to project circular manifolds using geodesic distances? In *ESANN* (2004), Citeseer, pp. 223–230.

[55] Lee, John M. Smooth manifolds. In *Introduction to Smooth Manifolds*. Springer, 2003, pp. 1–29.

[56] Levina, Elizaveta, and Bickel, Peter J. Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems* (2004), pp. 777–784.

[57] Lin, Tong, and Zha, Hongbin. Riemannian manifold learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*, 5 (2008), 796–809.

[58] Liu, Bo, Jing, Liping, Yu, Jian, and Li, Jia. Robust graph learning via constrained elastic-net regularization. *Neurocomputing 171* (2016), 299–312.

[59] Liu, R., Hao, R., and Su, Z. Mixture of manifolds clustering via low rank embedding. *Journal of Information & Computational Science 8* (2011), 725–737.

[60] Liu, Wei, and Chang, Shih-Fu. Robust multi-class transductive learning with graphs. In *Computer Vision and Pattern Recognition, 2009* (2009), IEEE, pp. 381–388.

[61] Liu, Wei, He, Junfeng, and Chang, Shih-Fu. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (2010), pp. 679–686.

[62] Ma, Y., and Fu, Y. *Manifold Learning Theory and Applications*. CRC Press, 2011.

[63] Maaten, Laurens van der, and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605.

[64] Mahadevan, Sridhar, and Maggioni, Mauro. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research 8*, 2169-2231 (2007), 16.

[65] Maier, Markus, Luxburg, Ulrike V, and Hein, Matthias. Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems* (2008).

[66] Martinetz, Thomas, and Schulten, Klaus. Topology representing networks. *Neural Networks 7*, 3 (1994), 507–522.

[67] Mekuz, Nathan, and Tsotsos, John K. Parameterless isomap with adaptive neighborhood selection. In *Pattern Recognition*. Springer, 2006, pp. 364–373.

[68] Narayanan, Hariharan, and Mitter, Sanjoy. Sample complexity of testing the manifold hypothesis. In *Advances in Neural Information Processing Systems* (2010), pp. 1786–1794.

[69] Nene, Sameer A, Nayar, Shree K, and Murase, Hiroshi. Columbia object image library (COIL-100). Tech. rep., CUCS-006-96, February 1996.

[70] Nene, Sameer A, Nayar, Shree K, Murase, Hiroshi, et al. Columbia object image library (COIL-20). Tech. rep., CUCS-005-96, 1996.

[71] Ng, Andrew Y, Jordan, Michael I, Weiss, Yair, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems 2* (2002), 849–856.

[72] Nguyen-Tuong, Duy, and Peters, Jan. Model learning for robot control: a survey. *Cognitive processing 12*, 4 (2011), 319–340.

[73] Pan, Sinno Jialin, and Yang, Qiang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on 22*, 10 (2010), 1345–1359.

[74] Raina, Rajat, Battle, Alexis, Lee, Honglak, Packer, Benjamin, and Ng, Andrew Y. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. 759–766.

[75] Roweis, S.T., and Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 2323–232 (2000).

[76] Rowland, Todd. Manifold. From MathWorld—A Wolfram Web Resource. Last visited on March 30, 2015.

[77] Sakoe, H., and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on 26*, 1 (1978), 43–49.

[78] Samko, Oksana, Marshall, A David, and Rosin, Paul L. Selection of the optimal parameter value for the isomap algorithm. *Pattern Recognition Letters 27*, 9 (2006), 968–979.

[79] Saul, Lawrence K, and Roweis, Sam T. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *The Journal of Machine Learning Research 4* (2003), 119–155.

[80] Shi, Jianbo, and Malik, Jitendra. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 8 (2000), 888–905.

[81] Silva, Vin D, and Tenenbaum, Joshua B. Global versus local methods in nonlinear dimensionality reduction. In *Advances in neural information processing systems* (2002), pp. 705–712.

[82] Souvenir, R., and Pless, R. Manifold clustering. In *Tenth IEEE International Conference on Computer Vision* (2005), vol. 1, IEEE, pp. 648–653.

[83] Sutton, Richard S, and Barto, Andrew G. *Introduction to reinforcement learning*. MIT Press, 1998.

[84] Tarjan, Robert. Depth-first search and linear graph algorithms. *SIAM journal on computing 1*, 2 (1972), 146–160.

[85] Tenenbaum, J. B., Silva, V. De, and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science* (2000).

[86] Tenenbaum, Joshua B. Mapping a manifold of perceptual observations. *Advances in neural information processing systems* (1998), 682–688.

[87] Thorup, Mikkel. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM (JACM) 46*, 3 (1999), 362–394.

[88] Tipping, Michael E, and Bishop, Christopher M. Mixtures of probabilistic principal component analyzers. *Neural computation 11*, 2 (1999), 443–482.

[89] Van der Maaten, Laurens, and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, 2579-2605 (2008), 85.

[90] von Neumann, J. Functional operators. vol. ii. the geometry of orthogonal spaces, volume 22 (reprint of 1933 notes) of annals of math. *Studies. Princeton University Press* (1950).

[91] Vu, H.T., Carey, CJ, and Mahadevan, Sridhar. Manifold warping: Manifold alignment over time. In *AAAI* (2012).

[92] Wang, C., and Mahadevan, S. Manifold alignment using procrustes analysis. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 1120–1127.

[93] Wang, C., and Mahadevan, S. Manifold alignment preserving global geometry. *The 23rd International Joint conference on Artificial Intelligence* (2013).

[94] Wang, Chang, and Mahadevan, Sridhar. A general framework for manifold alignment. In *AAAI Fall Symposium on Manifold Learning and its Applications* (2009), pp. 53–58.

[95] Wang, Fei, and Zhang, Changshui. Label propagation through linear neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on 20*, 1 (2008), 55–67.

[96] Wang, Y., Jiang, Y., Wu, Y., and Zhou, Z.H. Multi-manifold clustering. In *PRICAI 2010: Trends in Artificial Intelligence*. Springer, 2010, pp. 280–291.

[97] Wang, Yong, Jiang, Yuan, Wu, Yi, and Zhou, Zhi-Hua. Local and structural consistency for multi-manifold clustering. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence* (2011), vol. 22, Citeseer, p. 1559.

[98] Wang, Yong, Jiang, Yuan, Wu, Yi, and Zhou, Zhi-Hua. Spectral clustering on multiple manifolds. *Neural Networks, IEEE Transactions on 22*, 7 (2011), 1149–1161.

[99] Weinberger, Kilian Q, and Saul, Lawrence K. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research 10* (2009), 207–244.

[100] Wu, Yiming, and Chan, Kap Luk. An extended isomap algorithm for learning multi-class manifold. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on* (2004), vol. 6, IEEE, pp. 3429–3433.

[101] Yale, Paul B. *Geometry and symmetry*. Courier Dover Publications, 1968.

[102] Yan, Shuicheng, and Wang, Huan. Semi-supervised learning by sparse representation. In *SDM* (2009), SIAM, pp. 792–801.

[103] Yang, Bo, and Chen, Songcan. Sample-dependent graph construction with application to dimensionality reduction. *Neurocomputing 74*, 1 (2010), 301–314.

[104] Yang, Liu, and Jin, Rong. Distance metric learning: A comprehensive survey. *Michigan State Universiy 2* (2006).

[105] Zemel, Richard S, and Carreira-Perpiñán, Miguel Á. Proximity graphs for clustering and manifold learning. In *Advances in neural information processing systems* (2004), pp. 225–232.

[106] Zhang, Limei, Chen, Songcan, and Qiao, Lishan. Graph optimization for dimensionality reduction with sparsity constraints. *Pattern Recognition 45*, 3 (2012), 1205–1210.

[107] Zhang, Zhenyue, Wang, Jing, and Zha, Hongyuan. Adaptive manifold learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 2 (2012), 253–265.

[108] Zhang, Zhenyue, and Zha, Hongyuan. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *arXiv preprint cs/0212008* (2002).

[109] Zhang, Zhenyue, and Zha, Hongyuan. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing 26*, 1 (2005), 313–338.

[110] Zhou, Dengyong, Bousquet, Olivier, Lal, Thomas Navin, Weston, Jason, and Schölkopf, Bernhard. Learning with local and global consistency. *Advances in neural information processing systems 16*, 16 (2004), 321–328.

[111] Zhou, Dengyong, Bousquet, Olivier, Lal, Thomas Navin, Weston, Jason, and Schölkopf, Bernhard. Learning with local and global consistency. *Advances in neural information processing systems 16*, 16 (2004), 321–328.

[112] Zhou, F., and De La Torre, F. Canonical time warping for alignment of human behavior. *Advances in Neural Information Processing Systems (NIPS)* (2009).

[113] Zhu, Xiaojin, and Ghahramani, Zoubin. Learning from labeled and unlabeled data with label propagation. Tech. rep., CMU-CALD-02-107, Carnegie Mellon University, 2002.

[114] Zhu, Xiaojin, Lafferty, John, and Rosenfeld, Ronald. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, School of Computer Science, 2005.