Graph Embedding and Nonlinear Dimensionality Reduction

Blake Shaw

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011

ABSTRACT

Graph Embedding and Nonlinear Dimensionality Reduction

Blake Shaw

Traditionally, spectral methods such as principal component analysis (PCA) have been applied to many graph embedding and dimensionality reduction tasks. These methods aim to find low-dimensional representations of data that preserve its inherent structure. However, these methods often perform poorly when applied to data which does not lie exactly near a linear manifold. In this thesis, I present a set of novel graph embedding algorithms which extend spectral methods, allowing graph representations of high-dimensional data or networks to be accurately embedded in a low-dimensional space. I first propose minimum volume embedding (MVE) which, like other leading dimensionality reduction algorithms, first encodes the high-dimensional data as a nearest-neighbor graph, where the edge weights between neighbors correspond to kernel values between points, and then embeds this graph in a low-dimensional space. Next I present structure preserving embedding (SPE), an algorithm for embedding unweighted graphs where similarity between nodes is not known. SPE finds low-dimensional embeddings which explicitly preserve graph topology, meaning a connectivity algorithm, such as k-nearest neighbors, will recover the edges of the input graph from only the coordinates of the nodes after embedding. I further explore preserving graph structure during embedding, and find the concept applicable to dimensionality reduction, large-scale network visualization, and metric learning for link prediction. This thesis posits that simply preserving pairwise distances, as with many spectral methods, is insufficient for capturing the structure of many datasets and that preserving both local distances and graph topology is crucial for producing accurate low-dimensional representations of networks and high-dimensional data.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Graph embedding algorithms place vertices at points on a surface, typically Euclidean space, and connect vertices with an arc if the vertices have an edge between them. There exist many graph embedding algorithms with different goals ranging from drawing graphs so that they have certain aesthetic qualities to embedding graphs in VLSI chip designs to minimize fabrication cost [Chung, 1997, Battista et al., 1999]. For example, a common objective for graph drawing is to minimize edge crossings, where it has been shown that all graphs can be embedded in 3-dimensional Euclidean space without any edges crossing. Only for the specific case of planar graphs is a zero-crossing embedding possible in 2D. Graph embedding is a useful tool for approximately solving hard combinatorial problems geometrically. For instance, Arora et al. [2004] approximate the NP-hard sparsest cut problem by first embedding a graph and then using hyperspheres to separate points. From among these many possible objectives this thesis will focus on developing graph embedding algorithms suited for dimensionality reduction, visualization, and metric learning.

Embedding algorithms are helpful for gaining insight into complex data sets described by high-dimensional features. Inherent in many datasets is a small set of natural parameters which capture the important sources of variation in the data. Extracting these parameters reveals the underlying structure, allowing for better data exploration, visualization, and modeling of the complex system. For example, Sirovich and Kirby [1987] showed that although images of faces are often described by high-dimensional vectors of pixel intensities, the variations in face images can be accurately captured by a few dozen key dimensions, also known as "eigenfaces". Similar patterns emerge in high-dimensional data arising in many areas of research, such as bioinformatics, robot navigation, and natural language processing [Tenenbaum et al., 2000, Bowling et al., 2005]. The small number of hidden degrees of freedom that capture the different continuous modes of variation in the data describe a low-dimensional manifold which can be represented as a graph specifying which points on the manifold are neighbors. Properly embedding this graph can recover the hidden low-dimensional coordinates. We first present minimum volume embedding (MVE), an algorithm for nonlinear dimensionality reduction designed to extend principal component analysis (PCA) to be able to recover nonlinear manifolds by employing graph embedding. Both MVE and PCA maximize the amount of variance of the data that is captured by the low-dimensional embedding, an objective we define as *spectral fidelity*; however, where PCA preserves all distances between data points, MVE preserves only local distances, those accurately measured along the natural manifold of the data as specified by a graph built from strong local pairwise relationships. MVE uses semidefinite programming and spectral decomposition to directly optimize the eigenspectrum of the data to preserve as much energy as possible within the few dimensions available to the embedding, simultaneously minimizing energy in directions orthogonal to the embedding, keeping the data in a so-called minimum volume manifold.

Graphs not only are useful for encoding neighborhood information in high-dimensional datasets, but are a natural representation for network data as well. Similar to high-dimensional data, network data is difficult to visualize and represent in a low-dimensional space. Network data is prevalent in fields ranging from computer vision, to sociology, to computational biology. We are motivated to apply graph embedding to this natural setting: given only an unweighted graph, where vertices and edges represent pairwise interactions between entities, embed the graph such that the low-dimensional coordinates for every vertex implicitly encode the binary connectivity. It is difficult to properly apply MVE to unweighted graphs; one must choose an appropriate graph-based kernel or assume all neighbors to be equidistant. To address this problem setting directly, we present structure preserving embedding (SPE), an algorithm for embedding unweighted, undirected graphs in Euclidean space such that the resulting embedding is low-dimensional and preserves the global topology of the input graph. Topology is preserved if a connectivity algorithm, such as k-nearest neighbors, can recover the original edges of the input graph exactly from the coordinates of the vertices after embedding. Similar to MVE, SPE is formulated as a semidefinite program combined with a singular value decomposition (SDP + SVD). The objective of the SDP ensures that the learned kernel matrix is low-rank. The linear constraints of the SDP enforce the connectivity structure of the input graph. Traditional graph embedding techniques such as spectral embedding and Laplacian eigenmaps do not explicitly preserve topology and therefore the resulting visualizations are often less informative.

Preserving graph topology during embedding is applicable to a wide range of problem settings and allows for a variety of implementations. We further demonstrate that preserving graph topology is compatible with preserving the distances between certain points: introducing SPE style constraints into MVE (MVE+SP) further increases the quality of the embeddings of high-dimensional data; when local distances and graph topology are preserved, a nearest-neighbor classifier maintains its performance even after low-dimensional embedding. Furthermore, we present structure preserving metric learning (SPML) which extends SPE to the setting when nodes in a network are described by features as well as connectivity information, allowing for more accurate link prediction.

This thesis is organized as follows. First we introduce the problem of dimensionality reduction and graph embedding, highlighting related work, and discussing motivating problems. In Chapters 2 and 3 we introduce MVE and SPE respectively, focusing on two key themes: finding low-dimensional embeddings, and different ways of preserving graph structure. Next, in Chapters 4 and 5 we discuss two key extensions to SPE: introducing structure preserving constraints into MVE (MVE+SP), and solving SPE for large graphs using stochastic gradient descent. In Chapter 6 we introduce a new metric learning algorithm based on preserving graph structure (SPML) that allows for accurate link prediction in networks described by node features as well as connectivity.

1.1 Related Work

1.1.1 Dimensionality Reduction

There is a rich history of algorithms for dimensionality reduction, progressing towards more efficient tools for capturing the significant structures in ever-larger and more complex datasets. Principal component analysis (PCA) [Jolliffe, 1986] and Multidimensional Scaling (MDS) [Cox and M.Cox, 1994] are the canonical methods for dimensionality reduction [Williams, 2001]. MDS aims to find a set of low-dimensional coordinates that best preserves pairwise distances in the original high-dimensional space. PCA finds linear projections of the data which have the greatest variance, effectively rotating the data such that the first dimension captures the most variance of the data, the second dimension captures the second most, and so on. Both PCA and MDS are typically solved by spectral decomposition, and are equivalent when MDS is used with Euclidean distances. When the data lies near a linear manifold, PCA and MDS will recover a small set of coordinates which describes variations in the data aligned with this linear manifold.

However, PCA is limited as a purely linear method, and thus many extensions to PCA have been proposed, most notably kernel PCA [Schölkopf et al., 1998], which performs the spectral decomposition in kernel space and is therefore able to capture certain types of non-linear variations in the data via the choice of different kernels. More recent work such as Isomap [Tenenbaum et al., 2000], locally linear embedding (LLE) [Roweis and Saul, 2000], and Laplacian eigenmaps [Belkin and Niyogi, 2002] assume the data lies near a nonlinear manifold, and thus represent the data as a weighted graph in order to better honor those nonlinearities. Isomap essentially performs MDS but uses geodesic distances computed from a nearest-neighbor graph on the data. Locally linear embedding preserves local neighborhoods instead of distances, using the nearest-neighbors of each point to define reconstruction coefficients that are preserved during embedding. Laplacian eigenmaps performs a spectral decomposition of the graph Laplacian of the nearest-neighbor graph. The key intuition underlying these techniques is that it is important to measure distances along the manifold of the data and not through arbitrary dimensions in high-dimensional space; moving off the manifold corresponds to hallucinating datapoints that could not be generated by the underlying parameters that generate the data.

Maximum variance unfolding (MVU) [Weinberger et al., 2004] similarly operates on a graph formed from the data, preserving local distances but allowing large distances not measured along the manifold to vary as the manifold is "unfolded" by means of a semidefinite program (SDP). MVU is built on top of kernel principal component analysis, first learning a kernel matrix which better captures the structure of the nonlinear manifold, and then, like PCA, decomposing that kernel matrix by means of an SVD to find the d eigenvectors with the largest eigenvalues, which correspond to the d dimensions with the largest variance.

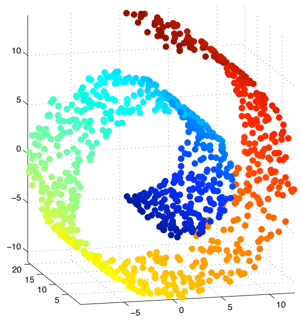
When using PCA for dimensionality reduction, there is corruption when eigenvectors with non-zero eigenvalues are truncated; pairwise distances in the reduced dimensionality space do not match the original distances. MVU addresses this problem implicitly to some degree by preserving only local distances; thus, in practice less eigenenergy is lost when the dimensions with smallest variance are truncated. In a previous publication [Shaw and Jebara, 2007], we introduce a cost function which explicitly minimizes the amount of eigenenergy lost by truncation, and thus better maintains as much eigenenergy as possible in d dimensions. And in a later article [Shaw and Jebara, 2011a], we show that this cost function is related to the spectral fidelity of the embedded data which we define as the percentage of original variance preserved. PCA maximizes spectral fidelity by definition. If we loosen the assumption that all pairwise distances should be preserved to all *local* pairwise distances should be preserved, we see that MVE is the natural algorithm for maximizing spectral fidelity while preserving local structure.

Many recent dimensionality reduction techniques begin by first learning a kernel matrix with specific properties and then decomposing that kernel matrix to create an embedding [Tenenbaum et al., 2000, Weinberger et al., 2004, Javanmard and Montanari, 2011]. Similar to how Isomap replaces Euclidean distances with geodesic distances before performing MDS, these methods learn a new kernel matrix for use with kernel PCA. MVU learns a kernel matrix which preserves local distances while maximizing variance in all dimensions. Action Respecting Embedding (ARE) [Bowling et al., 2005] learns a kernel matrix where movements in the low-dimensional space are specifically tied to metadata about which actions were taken by a robot to make that movement. Colored MVU [Song et al., 2008] similarly preserves local distances but exploits labels in the data to better separate classes in the embedding. Similar to these methods, MVE maximizes spectral fidelity while preserving local distances.

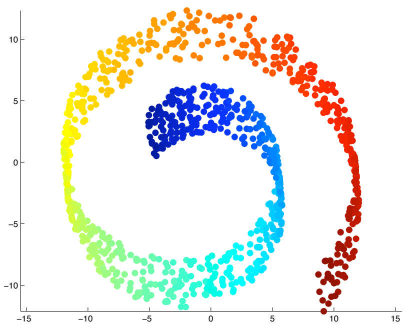
In Figure 1.1, we introduce the problem of dimensionality reduction with the classic swiss-roll example; the goal is to recover an accurate 2D embedding of 1000 datapoints sampled with noise from a smooth nonlinear manifold in 3D. In this comparison, PCA simply produces the 2D projection that maximizes variance. t-SNE [van der Maaten and Hinton, 2008], a popular method for visualization based on preserving a probability distribution during embedding, is also unable to capture the structure of the underlying manifold, seeming to cluster parts of the manifold together. Laplacian eigenmaps, MVU, and MVE produce 2D embeddings which properly show the variations along the manifold. In this thesis, we focus on recovering embeddings which capture the smooth variations aligned with the underlying manifold, and thus focus on techniques and datasets suited to manifold learning and graph embedding.

1.1.2 Graph Embedding

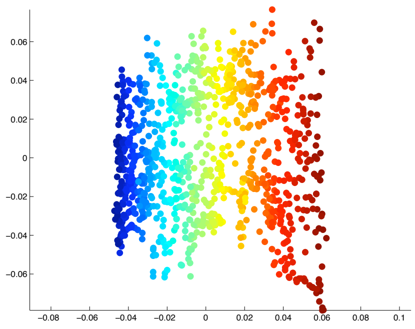
When the input data is a binary adjacency matrix as opposed to high-dimensional data, many of these graph-based dimensionality reduction algorithms can be applied directly. Spectral embedding essentially performs PCA on the adjacency matrix. Similarly, Laplacian eigenmaps employs a spectral decomposition of the graph Laplacian [Belkin and Niyogi, 2002]. In many cases these spectral methods offer significant improvements over spring embedding techniques, which often employ heuristics and are prone to getting stuck in local minima. However, in practice decomposing the adjacency matrix or graph Laplacian often yields many eigenvectors with non-zero eigenvalues, and thus choosing a small set of these eigenvectors as dimensions for embedding does not adequately represent the underlying structure of the input graph. To address this issue, we have developed structure preserving embedding [Shaw and Jebara, 2009], a convex method that learns a positive semidefinite kernel matrix whose spectral decomposition yields a small set of eigenvectors which preserve the topology of the input graph. Topology is preserved if applying a connectivity algorithm such as k-nearest neighbors, b-matching, or maximum weight spanning tree to an embedding



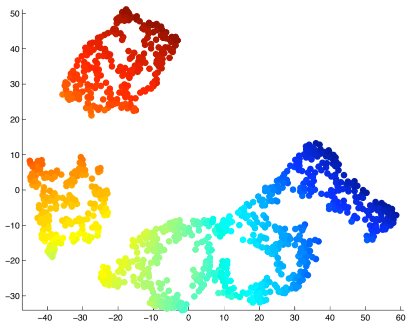
(a) Original Swiss Roll



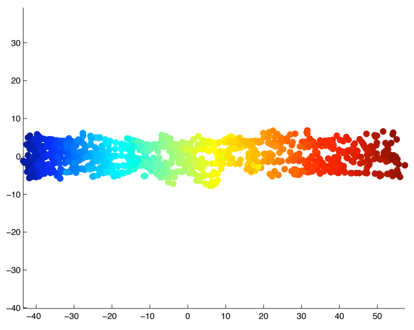
(b) PCA



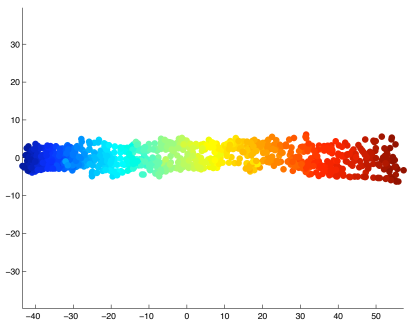
(c) Laplacian eigenmaps



(d) t-SNE

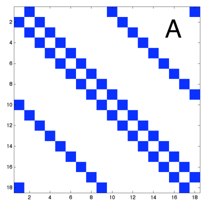


(e) MVU

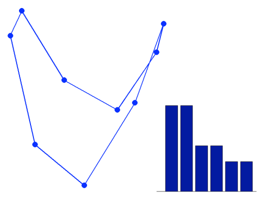


(f) MVE

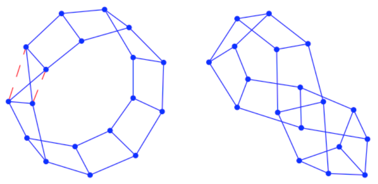
Figure 1.1: From 3D to 2D. The Swiss roll example illustrates the problem of recovering a low-dimensional manifold from nonlinear high-dimensional data. The 2D embedding recovered by PCA (b) does not capture the underlying manifold. t-SNE (d) appears to unnecessarily clump together parts of the underlying manifold. MVE, MVU, and Laplacian eigenmaps (f,e,c) recover the underlying 2D structure of the data.



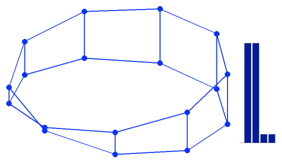
(a) Möbius Ladder Graph Adjacency Matrix



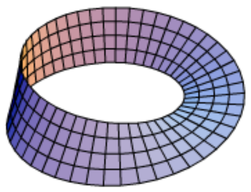
(b) Spectral Embedding



(c) Two Spring Embeddings



(d) SPE Embedding



(e) Möbius Band

Figure 1.2: Embedding the classical Möbius Ladder Graph. Given the adjacency matrix (a), the visualizations produced by spectral embedding and spring embedding (b and c) do not accurately capture the graph topology. The SPE embedding (d) uses few dimensions and matches the topology of the input graph exactly.

returns the input adjacency matrix exactly.

Traditional graph embedding algorithms such as spectral embedding or spring embedding techniques do not explicitly preserve the topology of the input graph. Consider Figure 1.2, which shows the results of using these algorithms to visualize the classical Möbius ladder graph. The spectral embedding seems to have collapsed certain pairs of nodes, and does not resemble a true Möbius band. Furthermore, the eigenspectrum indicates 6 dominant eigenvectors, when intuitively we expect to be able to represent the graph accurately using fewer than 6 dimensions. The left spring embedding is a good diagram of the input graph; however, we see that the characteristic twist of the Möbius strip is not accurately captured. A simple nearest-neighbor-finding algorithm would connect the nodes along the red dotted lines, not the blue ones specified by the adjacency matrix, and thus topology is not preserved. Because the objective function of spring embedding is non-convex, poor random initialization will often produce undesirable results, as shown with the spring embedding on the right. We are motivated to find simple tools for properly visualizing graphs such as the Möbius ladder, as well as graphs built from high-dimensional data such as the Swiss roll. In this thesis, we will focus on MVE and SPE, and highlight two important aspects of graph embedding: creating low-dimensional embeddings and preserving graph structure.

1.1.3 Optimization Techniques and Complexity

Many embedding algorithms are based on an eigendecomposition and thus their running time complexity scales cubicly with the number of data points. Furthermore, for methods based on learning a kernel matrix to be used with kPCA, the constraint that the kernel matrix must be a valid mercer kernel typically means the kernel learning algorithm involves a semidefinite program (SDP) which, when solved by an interior point method also scale cubicly in terms of the number of data points and constraints [Borchers, 1997]. SDPs have been successfully applied to many combinatorial problems on graphs such as the max-cut problem [Goemans and Williamson, 1995] and the vertex-cover problem [Kleinberg and Goemans, 1995]. Furthermore, there has been considerable research focused on methods for speeding up the underlying optimizations of embedding algorithms, and also on approximate methods for handling large-scale data.

There have been notable speedups proposed for MVU which can similarly be applied to MVE. In Weinberger et al. [2005], the authors propose a landmark method where a small sample of points denoted as "landmarks" are embedded into a low-dimensional space via MVU and the rest of the points are linearly reconstructed in the low-dimensional space by their relationships to the landmarks. Similarly in Weinberger et al. [2007], the author proposes learning a kernel matrix which is factorized as a set of basis vectors formed from eigenvectors of the graph Laplacian and a smaller positive semidefinite (PSD) matrix which is optimized via an SDP. In A. Kleiner [2010], the authors propose a method for speeding up a variety of SDPs including MVU, by transforming the semidefinite program into a sequence of two-variable optimizations, exploiting the fact that linear combinations of PSD matrices are also PSD, and thus the explicit PSD constraint can be dropped if the initial iterate is

PSD and further iterates are linear combinations of outer products of rank 1 matrices.

To allow SPE to scale to large graphs we employ an SDP solver by Burer and Monteiro [2003] which solves the SDP using an optimization over a low-rank factorization; however the method still does not allow SPE to scale much past graphs with 1000 nodes. Zhuang et al. [2009] have since demonstrated a significant speedup to MVE and SPE by formulating the optimization as a saddle-point optimization which is solved via a fast iterative algorithm. Their method provides a 10-30x speedup for MVE and 50-90x speedup for SPE. To better address the need to embed large graphs, we introduce in Shaw and Jebara [2011b] a method for optimizing SPE based on stochastic gradient decent [Robbins and Monro, 1951, Bottou, 2003] which sacrifices convexity but in practice allows the algorithm to embed graphs of up to fifty thousand nodes. More details about large-scale SPE can be found in Section 5.

1.1.4 Metric Learning

There exists a natural connection between graph embedding and metric learning algorithms. MVE and SPE are essentially unsupervised metric learning algorithms, learning a set of distances between nodes or data points with special properties (specifically that they can can be represented by low-dimensional vectors). Metric learning is often applied to supervised problems such as classification by employing graph-based constraints [Weinberger and Saul, 2009, Chechik et al., 2010]. These methods first build a k-nearest neighbors (kNN) graph from training data with a fixed k, and then learn a Mahalanobis distance metric which tries to keep connected points with similar labels close while pushing away class impostors, pairs of points which are connected but of a different class. Fundamentally, these supervised methods aim to learn a distance metric such that applying a connectivity algorithm (for instance, k-nearest neighbors) under the metric will produce a graph where no point is connected to others which have different class labels. In practice, these constraints are enforced with some slack. Once the metric is learned, the class label for an unseen datapoint can be predicted by the majority vote of nearby points under the learned metric.

These SPE-like constraints for learning a metric that is a good indicator of class labels suggest that we can learn a metric that is a good indicator of the correct graph topology as well. To accomplish this goal we present an algorithm Structure Preserving Metric Learning

(SPML). Like these previous metric learning methods, SPML aims to learn a metric that, when combined with a connectivity algorithm and node features data, produces a graph that matches as closely as possible the original input connectivity structure. Instead of pushing away class impostors, SPML pushes away *graph impostors*, points which are close in terms of distance but which should remain unconnected for structural reasons indicated by the connectivity matrix. SPML is similar to SPE. SPE learns coordinates for nodes in a graph such that applying a connectivity algorithm to those node coordinates yields the original adjacency matrix. Where SPE learns an embedding explicitly from only the target adjacency matrix, SPML learns a metric corresponding to a linear transformation of additional node features given as input. Like SPE, SPML can be optimized by a cutting-plane approach similar to structured prediction algorithms, explicit SDP constraints, or fast gradient updates.

1.2 Notation and Background

Formally the problem of graph embedding can be stated as follows: given a graph $G = (V, E)$ with $n = |V|$ nodes and $|E|$ edges, find for each node a vector $\mathbf{y}_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$. In this thesis we focus on low-dimensional embeddings where d is small. Often the edges of the graph G are parametrized by an adjacency matrix $\mathbf{A} = \mathbb{B}^{N \times N}$ specifying which of the n nodes are connected. We can represent the embedding of n vertices as a single matrix: $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]$ of size $d \times n$. For purposes of dimensionality reduction it is useful to consider the nodes as corresponding to a set of n objects: $\mathbf{x}_i \in \Omega$ for $i = 1, \ldots, n$ where Ω is an arbitrary space (commonly $\mathbb{R}^D$ for some large D), and a valid kernel function $W_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ specifying the similarity between pairs of objects. $\mathbf{W}$ is referred to as the Gram matrix or kernel matrix, and for graph-embedding-based dimensionality reduction algorithms, $\mathbf{W}$ is used as the weights associated with the edges of G.

The fundamental optimization underlying traditional spectral methods as well as the algorithms proposed in this thesis is eigedecomposition. Given a square $n \times n$ matrix $\mathbf{K}$, the matrix can be factorized as $\mathbf{K} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$ where each λ_i is an eigenvalue of $\mathbf{K}$ with corresponding eigenvector $\mathbf{v}_i$. We can factorize $\mathbf{K} = \mathbf{Y}^\top \mathbf{Y}$ by constructing $\mathbf{Y}$ as the stacked eigenvectors of $\mathbf{K}$ scaled by the square root of their eigenvalues. This mapping from $\mathbf{K}$ to $\mathbf{Y}$ forms the basis for the kernel view of principal component analysis. If all of eigenvalues of $\mathbf{K}$ are non-negative, $\mathbf{K}$ is positive semi-definite, denoted as $\mathbf{K} \succeq 0$. We commonly exploit the useful property that $\mathrm{tr}(\mathbf{K}) = \sum_i \lambda_i$.

Another key optimization tool used in this work is semidefinite programming. A semidefinite program (SDP) is a convex optimization over a semidefinite matrix which has a linear objective function and a set of linear constraints, and takes the form:

$$\begin{aligned} &\max_{\mathbf{K}\succeq 0} \mathrm{tr}(\mathbf{E}\mathbf{K}) \\ &\text{s.t. } \mathrm{tr}(\mathbf{B}_i\mathbf{K}) = g_i \quad \text{for} \quad i = 1 \dots p \end{aligned}$$

where $\mathbf{E}$ denotes the linear objective, and each $\mathbf{B}_i$ and g_i denote one of of the p linear constraints of the optimization. An SDP is essentially a linear program (LP) with the additional constraint that the target variable must form a PSD.

Chapter 2

Minimum Volume Embedding

Minimum volume embedding (MVE) is an algorithm for non-linear dimensionality reduction and weighted graph embedding. In Section 2.1, we introduce the concept of spectral fidelity and illustrate its utility as a cost function for dimensionality reduction, formulating the problem as an optimization over the eigenvalues of a kernel matrix. We then introduce the sequential SDP machinery for optimizing general eigenfunctions of this class by combining semidefinite programming (SDP) with singular value decompositions (SVD). The dimensionality reduction algorithm MVE is then presented in Section 2.1.4 as a means for optimizing spectral fidelity. In Sections 2.1.5 and 2.2, MVE is expanded into a flexible framework for properly embedding many kinds of high-dimensional data, graphs, and trees. We consider a variety of algorithmic choices, including different kernels, graph-building techniques, and variations on the optimization constraints. In Section 2.3 we present a variety of synthetic and real-world experiments which highlight the improvements of MVE compared to other methods.

2.1 Fidelity and Objective Functions for Dimensionality Reduction

We closely analyze the objectives of PCA, MVU, and MVE and define a new quantity "spectral fidelity" as a natural measure for evaluating the quality of dimensionality reduction. Intuitively, the dimensionality of an embedding should match the number of underlying

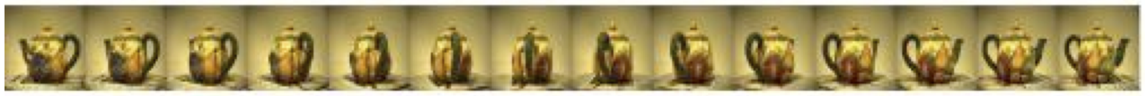


Figure 2.1: Images of a teapot rotating on a table. Although each image is represented by many pixels, there exists only one inherent degree of freedom in the data: the angle of the teapot [Weinberger et al., 2004].

parameters in the data. When a dataset consists of a large number of seemingly unrelated features, in many cases these features are nonlinearly dependent on a small set of hidden modes of variability inherent in the data. Consider, for example, images taken of a teapot rotating on a table [Weinberger et al., 2004] as shown in Figure 2.1; each image can be represented by a datapoint in $\mathbb{R}^D$, where the D dimensions correspond to a rasterized list of pixel intensity values. In this case there is one feature per pixel, and these D features drastically change from one image to the next. However, there exists only one underlying mode of variation in this dataset: the angle of rotation of the teapot. An embedding with high spectral fidelity would have only one significant dimension, corresponding exactly to this angle of rotation.

Specifically, spectral fidelity measures how much of the variance of the original data is preserved by the embedding. PCA naturally maximizes spectral fidelity, while preserving all pairwise distances in the data. We show that by using a sequential SDP technique MVE also maximizes spectral fidelity while preserving only local distances measured along the natural manifold of the data thus addressing a major shortcoming of PCA by being able to capture nonlinear modes of variation.

2.1.1 PCA and MVU

PCA centers and rotates the data such that the d-dimensions of the embedding have the highest variance. Given as input the $n \times n$ Gram matrix $\mathbf{W}$ where $W_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{W}$ is first centered:

$$\hat{\mathbf{W}} = \mathbf{W} - \frac{1}{n}\mathbf{W}\mathbf{1} - \frac{1}{n}\mathbf{1}^\top\mathbf{W} + \frac{1}{n^2}\mathbf{1}^\top\mathbf{W}\mathbf{1},$$

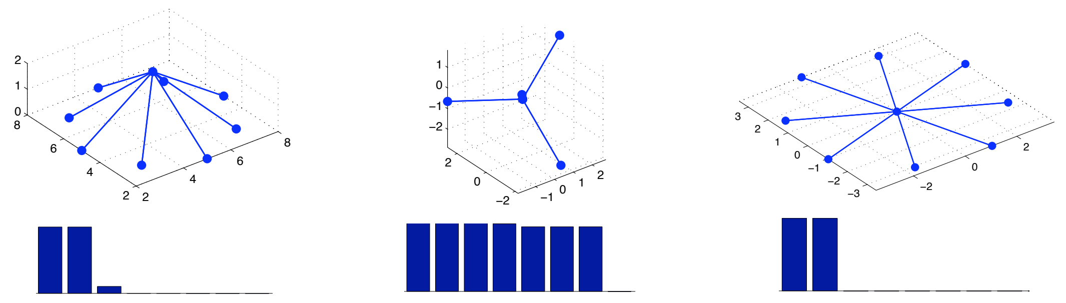
where $\mathbf{1}$ is the ones matrix of size $n \times n$. $\hat{\mathbf{W}}$ is then decomposed into its eigenvectors and eigenvalues $\hat{\mathbf{W}}\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Choosing the d eigenvectors of $\hat{\mathbf{W}}$ with largest eigenvalues as each of the d dimensions of the embedding, $\mathbf{y}_i = \sqrt{\lambda_i}\mathbf{v}_i$ for $i = 1, \ldots, d$ where $\lambda_i > \lambda_{i+1}$, PCA maximizes the amount of variance captured in the top d dimensions.

If there are less than d-dimensions with non-zero variance then PCA has perfectly preserved all pairwise relationships in the data while reducing the dimensionality to d. However, as the number of dimensions with non-zero variance grows larger than d, the embedding more poorly approximates all the pairwise distances, because PCA has truncated dimensions with important information. If the data lies near a nonlinear manifold embedded in high-dimensional space, we cannot trust large distances which are not measured along the manifold. All entries of the Gram matrix $\mathbf{W}$ cannot be trusted equally since the simple linear kernel function used to create $\mathbf{W}$ rarely represents perfect geodesic distances between all pairs of points. If we relax the constraint that all pairwise distances must be preserved exactly to the constraint that only local distances must be preserved, is it possible to use fewer dimensions, and thus also not truncate meaningful variations in the data?

Maximum variance unfolding (MVU) is built upon this intuition, preserving only local distances described by a graph on the data, while unfolding the graph to capture nonlinearities. MVU learns a kernel matrix $\mathbf{K}$ by optimizing

$$\begin{aligned}
&\max_{\mathbf{K}\in\mathcal{K}} \operatorname{tr}(\mathbf{K}) \\
&\text{s.t. } \mathcal{K} = \left\{ \forall \mathbf{K} \in \mathbb{R}^{n\times n} \;\middle|\; \begin{array}{l} \mathbf{K} \succeq 0 \\ \sum_{ij} K_{ij} = 0 \\ K_{ii} + K_{jj} - 2K_{ij} = W_{ii} + W_{jj} - 2W_{ij} \;\; \forall_{i,j} \text{ s.t. } A_{ij} = 1 \end{array} \right\}.
\end{aligned}$$

The binary adjacency matrix $\mathbf{A} \in \mathbb{B}^{n\times n}$ specifies which local distances are preserved. The optimization is also constrained to ensure $\mathbf{K}$ is centered and a valid kernel matrix. Maximizing the trace of $\mathbf{K}$ corresponds to "unfolding" the graph. Similar to PCA, after learning $\mathbf{K}$, the embedding is found by using the d eigenvectors of $\mathbf{K}$ with largest eigenvalues $\mathbf{y}_i = \sqrt{\lambda_i}\mathbf{v}_i$ for $i = 1, \ldots, d$ where $\mathbf{K}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ and $\lambda_i > \lambda_{i+1}$. In practice, MVU trades off preserving all distances for preserving only local distances but using far fewer dimensions. Unfortunately, the objective function $\max_{\mathbf{K}\in\mathcal{K}} \operatorname{tr}(\mathbf{K})$ can be problematic. Maximizing the trace is equivalent to maximizing variance in all dimensions, essentially pulling the data apart in all directions; this objective can cause a variety of degeneracies, as illustrated by the star graph example in Figure 2.2 and the S-curve example in Figure 2.3. We see that MVU pulls the star graph into an n-dimensional ball. Pulling the data apart in all dimensions runs counter to the intuition of reducing dimensionality. Ideally, we would like to maximize variance only in the d dimensions available to the embedding, and not drive variance into all n dimensions as is the case with MVU.

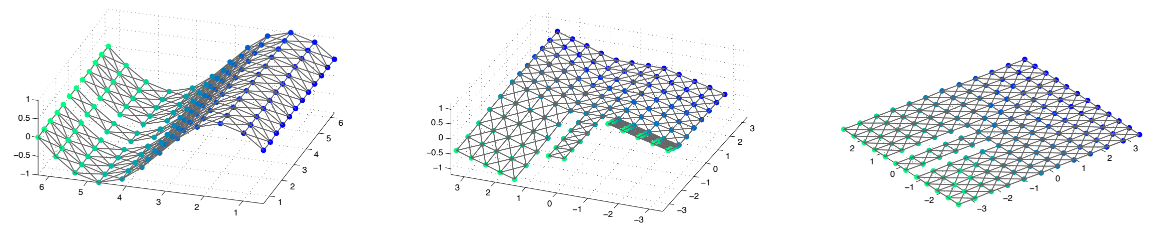


(a) Original (b) MVU (c) MVE

Figure 2.2: The star graph example illustrates a common deficiency with MVU. Below each embedding is the eigenspectrum. We see that MVU puffs the data out into an n-dimensional ball, where MVE recovers the inherent 2D structure in the data.

2.1.2 Fidelity

As shown in Jolliffe [1986] and Schölkopf et al. [1998], the first d eigenvectors found by PCA carry more variance than any other d orthogonal directions, and the variance of the data captured by each eigenvector is proportional to the corresponding eigenvalue [Srebro, 2004]. From this perspective, PCA can be seen as maximizing the ratio of the sum of top eigenvalues to the sum of all eigenvalues. Furthermore, because $\operatorname{tr}(\mathbf{K}) = \sum_{i=1}^{n} \lambda_i$, we see that MVU is trying to maximize all eigenvalues and thus maximize variance. This



(a) Original (b) MVU (c) MVE

Figure 2.3: The torn S-curve example shows when the 2D manifold is ripped, MVU incorrectly pulls the data into 3D, where MVE recovers the original 2D manifold.

connection between the variance of each dimension of the embedding and the eigenvalues of $\mathbf{K}$ begs the question: is it possible to maximize the amount of variance captured by the d dimensions available for embedding, similar to PCA, but only preserve local distances, similar to MVU?

Definition 1 *Given a kernel matrix $\mathbf{K}$, the spectral fidelity of a d-dimensional embedding $\mathbf{Y}$ computed from the d eigenvectors of $\mathbf{K}$ with largest eigenvalues is the normalized sum of the eigenvalues captured by the embedding: $F(\mathbf{K}, d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$ where $\mathbf{K}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ and $\lambda_i > \lambda_{i+1}$.*

Because the eigenvalues of $\mathbf{K}$ are directly related to the variance of the original data, spectral fidelity measures how much variance is preserved in a low-dimensional embedding. PCA naturally maximizes spectral fidelity by choosing the orthonormal leading eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$ as the embedding such that $\mathbf{K} = \sum_{i=1}^{d} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$. Now consider the following formulation for maximizing spectral fidelity:

$$\max_{\mathbf{K}} F_\beta(\mathbf{K}, d) \quad = \quad \max_{\mathbf{K}} \beta \sum_{i=1}^{d} \lambda_i - \sum_{i=1}^{n} \lambda_i. \tag{2.1}$$

It is clear that, for some choice of $\beta \in \mathbb{R}^+$, $\max_{\mathbf{K}} F_\beta(\mathbf{K}, d) = \max_{\mathbf{K}} F(\mathbf{K}, d)$. In practice β is tuned to maximize F($\mathbf{K}$, d). From this perspective, setting $d = n$ and $\beta = 2$ recovers the cost function for MVU. Note that MVU maximizes spectral fidelity to some degree; however, in essence MVU is pulling the data in n dimensions. We want to directly optimize this spectral function for a small value of d. We see that Equation 2.1 is precisely the objective proposed for the MVE algorithm [Shaw and Jebara, 2007] when $\beta = 2$, maximizing the amount of eigen-energy in the first few dimensions and explicitly minimizing the truncation loss associated with the PCA step. Thus MVE is a natural extension to PCA; both maximize spectral fidelity; MVE simply relaxes the constraint of preserving all distances to preserving local distances aligned with the underlying manifold.

Note that maximizing spectral fidelity alone is not a proper objective for dimensionality reduction. Spectral fidelity captures how much loss of variance there is when moving from a N dimensional embedding to a d dimensional embedding. Therefore, spectral fidelity is only useful when coupled with a set of constraints that exactly preserve some important property such as preserving local distances. Given that all local distances are preserved in $\mathbf{K}$, maximizing fidelity will minimize any loss of variance associated with truncating the embedding at d dimensions.

2.1.3 Optimizing Spectral Functions

In order to optimize Equation 2.1, we need to develop machinery for optimizing over the eigenvalues of a positive semidefinite kernel matrix [Boyd]. Building on the work from Overton and Womersley [1993], we show that spectral functions of the form

$$g(\mathbf{K}) \;=\; \sum_i \alpha_i \lambda_i \tag{2.2}$$

are convex, and can be optimized by alternating SDP and SVD optimizations. It is easy to see that $F_2(\mathbf{K}, d) = g(\mathbf{K})$ when $\alpha_i = 1$ for $i = 1, \ldots, d$ and $\alpha_i = -1$ for $i = d+1, \ldots, n$.

Theorem 2 *The sum of the top d eigenvalues of a positive semidefinite matrix $\mathbf{K}$, $f_d(\mathbf{K}) = \sum_{i=1}^{d} \lambda_i$ is convex in $\mathbf{K}$.*

See Overton and Womersley [1993] for proof.

Theorem 3 *The weighted sum of the eigenvalues of a positive semidefinite matrix $\mathbf{K}$ $g(\mathbf{K}) = \sum_{i=1}^{n} \alpha_i \lambda_i$ is convex in $\mathbf{K}$ if the weights and eigenvalues are arranged in nonincreasing order, $\alpha_i \geq \alpha_{i+1}$ and $\lambda_i \geq \lambda_{i+1}$.*

Proof Rewrite the function $g(\mathbf{K})$ as a combination of $f_d(\mathbf{K})$ functions as follows:

$$g(\mathbf{K}) \;=\; \alpha_n f_n(\mathbf{K}) + \sum_{i=1}^{n-1} (\alpha_i - \alpha_{i+1})\, f_i(\mathbf{K})$$

Verify that the terms in the parentheses are non-negative since $\alpha_i \geq \alpha_{i+1}$. Thus we have a conic combination of functions $f_d(\mathbf{K})$ which are all convex in $\mathbf{K}$ which shows that $g(\mathbf{K})$ convex in $\mathbf{K}$. ■

Clearly, maximizing this convex function could yield local maxima; however, because the function is convex, fast convergence is possible by variational maximization of a lower bound. Furthermore, by initializing at the PCA or MVU solution, the method is guaranteed to increase the spectral fidelity from these typically good starting solutions.

Consider rewriting Equation 2.2 in the following way which makes the relationship between $\mathbf{K}$, its eigenvalues λ_i and its eigenvectors $\mathbf{v}_i$ more explicit. Furthermore, assume we are given a set of linear constraints $\mathbf{K} \in \mathcal{K}$ which preserve local distances as in the MVU algorithm. To maximize spectral fidelity, we will optimize

$$\begin{array}{ll} \max\limits_{\mathbf{K}\in\mathcal{K}} & g(\mathbf{K}) \;=\; \max\limits_{\mathbf{K}\in\mathcal{K}} \sum_{i=1}^{n} \alpha_i \lambda_i \\ \text{s.t.} & \mathbf{K}\mathbf{v}_i = \lambda_i \mathbf{v}_i,\; \mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij},\; \lambda_i \geq \lambda_{i+1} \geq 0\; \alpha_i \geq \alpha_{i+1},\; \forall i,j \end{array} \tag{2.3}$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Now consider applying the following theorem, the proof of which can be found in 8.1.1.

Theorem 4 *For any $\mathbf{K} \in \mathbb{R}^{n\times n}$ such that $\mathbf{K} \succeq 0$, $\mathbf{K}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}, \lambda_i \geq \lambda_{i+1}$, and $\alpha_i \geq \alpha_{i+1}$ for $i = 1 \ldots n$:*

$$\sum_{i=1}^{n} \alpha_i \lambda_i = \max_{\substack{\mathbf{u}_1,\ldots,\mathbf{u}_n \\ \mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}}} \operatorname{tr}\left(\mathbf{K} \sum_{i=1}^{n} \alpha_i \mathbf{u}_i \mathbf{u}_i^\top\right).$$

Theorem 4 allows us write a variational version of the problem in Equation 2.3 which is straightforward to maximize. The variational version makes the choice of eigenvectors an additional parameter:

$$\max_{\mathbf{K}\in\mathcal{K}} \max_{\substack{\mathbf{u}_1,\ldots,\mathbf{u}_n \\ \mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}}} \operatorname{tr}\left(\mathbf{K}\sum_{i=1}^{n}\alpha_i \mathbf{u}_i\mathbf{u}_i^\top\right). \tag{2.4}$$

When we maximize over the additional parameter, we get the formulation in Equation 2.3. Thus, we can now approach the eigenfunction maximization problem as an alternating maximization found by iterating between solving for the best $\mathbf{K}$ given $\mathbf{u}_1, \ldots, \mathbf{u}_n$ and then solving for the best $\mathbf{u}_1, \ldots, \mathbf{u}_n$ given $\mathbf{K}$. This can be accomplished efficiently by interleaving SDP and SVD optimizations. Furthermore, we now have the machinery to maximize spectral fidelity by setting $\alpha_{1,\ldots,d} = \beta - 1$ and $\alpha_{d+1,\ldots,n} = -1$ and maximizing Equation 2.3.

2.1.4 Minimum Volume Embedding Algorithm

Using the framework presented above for optimizing spectral functions, we present minimum volume embedding (MVE), an efficient algorithm which directly optimizes the eigenspectrum of high-dimensional data in order to produce a low-dimensional embedding with maximum spectral fidelity. We first present a simplified view of the core algorithm, and then explore a variety of extensions and adaptations.

MVE reduces dimensionality by solving a graph embedding problem, first finding a sparse graph G that captures strong local pairwise affinities in the data, and then "unfolding" this graph to create a low-dimensional embedding. Each of the n nodes in G typically represents a high-dimensional data point $\mathbf{x}_i \in \mathbb{R}^D$ for $i = 1, \ldots, n$, but each node can represent any object which allows for pairwise similarities to be computed using a valid kernel function. The similarities between nodes are captured by a Gram matrix of kernel values $\mathbf{W} \in \mathbb{R}^{n\times n}$. The edges of G are represented as an adjacency matrix $\mathbf{A} \in \mathbb{B}^{n\times n}$ which indicates strong pairwise affinities as defined by $\mathbf{W}$. For example, $\mathbf{W}$ could be formed using a linear kernel function on the data, and $\mathbf{A}$ formed by an algorithm such as k-nearest neighbors. We discuss other choices for kernels and graph-building algorithms in Section 2.1.5.

Given this sparse graph G, our goal is to compute a low-dimensional embedding of G: $\mathbf{y}_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$, where $d \ll D$ and the underlying structure of G is preserved.

We assume that G represents a nonlinear manifold of low dimension arbitrarily embedded in a high-dimensional space; from this perspective our task is then to rotate and unfold the graph in order to produce a new set of coordinates, the axes of which more directly correspond to sources of variations along the underlying manifold.

MVE learns a kernel matrix $\mathbf{K}$ by maximizing a lower bound on spectral fidelity, subject to the constraints that $\mathbf{K}$ is positive-semidefinite, centered, and local distances are preserved. $\mathbf{K}$ is then decomposed using kernel PCA to find the d dimensions for the embedding. The objective function,

$$\max_{\mathbf{K}\in\mathcal{K}} F_\beta(\mathbf{K}, d) \quad = \quad \max_{\mathbf{K}\in\mathcal{K}} \beta \sum_{i=1}^{d} \lambda_i - \sum_{i=1}^{n} \lambda_i \tag{2.5}$$

essentially unfolds and flattens the data, while the constraints

$$\mathcal{K} = \left\{ \forall \mathbf{K} \in \mathbb{R}^{n\times n} \,\middle|\, \begin{array}{l} \mathbf{K} \succeq 0 \\ \sum_{ij} K_{ij} = 0 \\ K_{ii} + K_{jj} - 2K_{ij} = W_{ii} + W_{jj} - 2W_{ij} \;\; \forall_{i,j} \text{ s.t. } A_{ij} = 1 \end{array} \right\} \tag{2.6}$$

enforce the graph structure, essentially acting as rigid rods between neighboring points. We optimize Equation 2.5 by iteratively maximizing the variational lower bound derived in Section 2.1.3. The ideal solution involves sweeping β across various settings to find when fidelity is precisely minimized, however we have found strong empirical validation for setting $\beta = 2$. Algorithm 1 summarizes the steps of graph embedding using MVE, and the following theorem guarantees the convergence of the algorithm, the proof of which can be found in 8.1.2.

Theorem 5 *The iterative MVE algorithm is guaranteed to monotonically increase the objective function* $F_\beta(\mathbf{K}, d) = \beta\sum_{i=1}^{d}\lambda_i - \sum_{i=1}^{n}\lambda_i$ *where* $\mathbf{K} \in \mathbb{R}^{n\times n}, \mathbf{K} \succeq 0, \mathbf{K}v_i = \lambda_i v_i, v_i^\top v_j = \delta_{ij}, \lambda_i \geq \lambda_{i+1}$ *and* $\beta \geq 0$.

MVE is implemented in Matlab, using the SDP optimization package CSDP [Borchers, 1997]. Code is available online [Shaw and Jebara, 2008]. The running time of one iteration of MVE is identical to that of MVU: $O(n^3 + \mathcal{C}^3)$ where n is the number of data points and $\mathcal{C}$ is the number of constraints of the optimization, and $\mathcal{C}$ scales with the number of nonzero entries in $\mathbf{A}$.

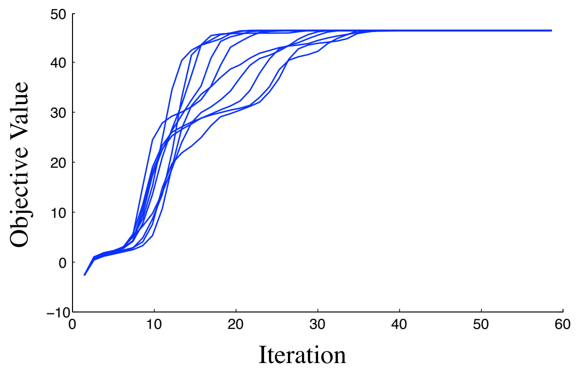


Figure 2.4: The objective value for 10 runs of MVE with random initializations (instead of initializing with PCA or MVU). We see that despite different initial conditions MVE converges reliably to the same solution. The dataset for this experiment is a subset of the 'Twos' dataset shown in more detail in Figure 2.5.

In practice, even a few iterations of MVE greatly improves upon either the PCA or MVU solution, and typically MVE requires less than a dozen iterations to converge. Each iteration of MVE is guaranteed to increase the objective function and thus improve the spectral fidelity of the initial solution (such as PCA or MVU). Although we have not guaranteed that MVE finds a global optimum, we empirically show that MVE reliably converges to the same solution despite drastically different initial conditions. In Figure 2.4, we see 10 random initializations of MVE for $d = 1$ on a subset of the Twos dataset (shown in greater detail in Section 2.3). Instead of initializing with the PCA or MVU solution, $\mathbf{W}$ is randomly initialized. Smart initialization can significantly reduce the number of iterations of the MVE optimization but does not in practice change the optimum found.

2.1.5 Kernel Functions and Graphs

The primary input to MVE is a Gram matrix $\mathbf{W}$ and sparse connectivity matrix $\mathbf{A}$. These matrices describe which pairwise distances defined by $\mathbf{W}$ are preserved in the embedding learned by MVE.

Definition 6 *Define the distance between a pair of points* (i, j) *with respect to a given positive semidefinite Gram matrix* $\mathbf{W}$, *as* $D_{ij} = W_{ii} + W_{jj} - 2W_{ij}$.

Algorithm 1 Minimum Volume Embedding

```
Input: a Gram matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$, sparse connectivity $\mathbf{A} \in \mathbb{B}^{n\times n}$, and parameters $d, \beta, \kappa$.

1: Initialize $\mathbf{K} \leftarrow \mathbf{W}$
2: $\mathcal{K} \leftarrow \{\mathbf{K} \succeq 0, \sum_{ij} K_{ij} = 0, K_{ii} + K_{jj} - 2K_{ij} = W_{ii} + W_{jj} - 2W_{ij}\ \forall_{i,j}$ s.t. $A_{ij} = 1\}$
3: repeat
4:     Solve for the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and eigenvalues and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ of $\mathbf{K}$
       using an SVD.
5:     $\mathbf{B} \leftarrow \beta \sum_{i=1}^{d} \mathbf{v}_i \mathbf{v}_i^\top - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^\top$
6:     $\mathbf{K} \leftarrow \hat{\mathbf{K}}$
7:     $\hat{\mathbf{K}} \leftarrow \mathrm{argmax}_{\mathbf{K}\in\mathcal{K}}\, \mathrm{tr}(\mathbf{KB})$ {Found via SDP}
8: until $\|\mathbf{K} - \hat{\mathbf{K}}\| \leq \kappa$
9: Perform SVD on $\hat{\mathbf{K}}$ to compute the $d$ leading eigenvectors $\hat{\mathbf{v}}_i$ and corresponding eigen-
   values $\hat{\lambda}_i$, and set $\mathbf{y}_i \leftarrow \sqrt{\hat{\lambda}_i}\hat{\mathbf{v}}_i$ for $i = 1, \ldots, d$
10: return $\mathbf{y}_1, \ldots, \mathbf{y}_d$
```

We see that if $\mathbf{W}$ is formed via a linear kernel function, then $\mathbf{D}$ is the matrix of squared Euclidean distances between all pairs of points. If one is given a kernel describing the exact distances between all pairs of points, running MVE is unnecessary; kPCA will yield a low-dimensional embedding which best preserves those distances. However, in practice one is rarely given a well-suited kernel that approximates variations only along a nonlinear manifold. Thus MVE uses a sparse graph $\mathbf{A}$ to specify which distances are aligned with the manifold, and should be preserved exactly, and which distances are measured along spurious directions in high-dimensional space and should be free to vary. Running MVE with a fully connected $\mathbf{A}$ matrix is equivalent to using kPCA because the graph is structurally rigid [Asimov and Roth, 1978]; as $\mathbf{A}$ becomes more sparse, the graph becomes flexible and MVE is given more freedom to unfold the underlying manifold. The relationship between the rigidity of these graphs and the existence of unique embedding solutions is an active area of research in rigidity theory [Singer and Cucuringu, 2009, Javanmard and Montanari, 2011].

Because MVE requires only a Gram matrix describing the pairwise relationships in the data, MVE is well-suited to embedding datasets arising from fields which use specialized kernels, such as bioinformatics or natural language processing [Lodhi et al., 2002]. The matrix $\mathbf{W}$ need only be a positive semidefinite Gram matrix. Given a set of n objects: $\mathbf{x}_i \in \Omega$ for $i = 1, \ldots, n$, one computes $\mathbf{W}$ via a kernel function $\mathbf{W}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. For high-dimensional data, common choices of $k(\mathbf{x}_i, \mathbf{x}_j)$ include the linear, polynomial and RBF kernels.[1] Also note that MVE is to able to handle input of a distance matrix specifying pariwise distances d_{ij} instead of the Gram matrix $\mathbf{W}$. This setting might be useful for embedding objects where only explicit distances are known, and no kernel function is specified. In this case, the distance preserving constraint of MVE can be modified to be $K_{ii} + K_{jj} - 2K_{ij} = \sqrt{d_{ij}}\ \forall_{i,j}$ s.t. $A_{ij} = 1$.

In certain datasets, such as those arising from social networks, the sparse connectivity structure $\mathbf{A}$ is given explicitly as input; it is then straightforward to run MVE. Otherwise, $\mathbf{A}$ must be found directly from the Gram matrix $\mathbf{W}$. $\mathbf{A}$ specifies a sparse graph; when $A_{ij} = 1$ points i and j are neighbors, and they can be thought of as being connected by a rigid rod that preserves the distance between them during unfolding. We present a variety of methods for finding the connectivity structure $\mathbf{A}$.

Thresholding

The simplest method for connecting neighbors is to connect all points whose pairwise distances (as defined by $\mathbf{W}$) are less than a threshold parameter ϵ, setting $A_{ij} = 1$ for any $D_{ij} < \epsilon$ and setting $A_{ij} = 0$ elsewhere. This method is also known as ϵ-balls.

[1]

Linear	$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
Polynomial	$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j + 1)^p$
RBF	$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\lvert\lvert\mathbf{x}_i - \mathbf{x}_j\rvert\rvert^2}{2\sigma^2}\right)$

k-nearest Neighbors

The k-nearest neighbor (kNN) algorithm finds an $\mathbf{A} \in \mathbb{B}^{n \times n}$ in which each node is connected to the k nodes to which it is the closest:

$$\arg\min_{\mathbf{A}} \quad \sum_{ij} D_{ij} A_{ij} \;\text{ s.t. }\; \sum_j A_{ij} = k, A_{ii} = 0, A_{ij} \in \{0, 1\} \;\forall_{i,j}.$$

The algorithm minimizes $\sum_{ij} D_{ij} A_{ij}$ by greedily selecting for each node a the k smallest distances of D_{aj} and setting those corresponding entries in A_{aj} to 1 and 0 elsewhere, where k is a parameter of the algorithm. In this article, we refer to kNN as the symmetric version, where $\mathbf{A}$ is symmetrized by setting $A_{ij} = \max(A_{ij}, A_{ji})$. It is also possible to symmetrize $\mathbf{A}$ by setting $A_{ij} = \min(A_{ij}, A_{ji})$.

b-matching

Given a set of desired degrees for each node $b_1, \ldots, b_n$, we can find the best connectivity matrix that maintains the constraints that the ith node must have exactly b_i neighbors:

$$\arg\min_{\mathbf{A}} \quad \sum_{ij} D_{ij} A_{ij} \;\text{ s.t. }\; \sum_j A_{ij} = b_i, A_{ij} = A_{ji}, A_{ii} = 0, A_{ij} \in \{0, 1\} \;\forall_{i,j}.$$

Note that when all of the degrees are uniformly set to a single value b we get a b-regular graph. This graph is different from the corresponding kNN graph when $k = b$ because the kNN graph may contain nodes that have degree much higher or lower than k due to symmetrization. For example, although each node picks k of its neighbors to connect to, there are no constraints limiting how many times a node can be picked as another node's neighbor. In practice, certain "popular" nodes can have high degree because although they only choose k neighbors, many more than k other nodes pick that node as a neighbor. The graphs produced by b-matching often produce a smoother manifold. In Section 2.3 we show a comparison on the AOL search dataset that illustrates the differences between kNN and b-matching. In our experiments we use fast b-matching code based on loopy belief propagation by Huang and Jebara [2007].

Minimum Weight Spanning tree

The minimum weight spanning tree algorithm finds a subgraph that connects all nodes together forming a tree whose edge distances are minimal

$$\arg\min_{\mathbf{A}} \quad \sum_{ij} D_{ij} A_{ij} \quad \text{s.t.} \quad A_{ij} \in \{0,1\} \; \forall_{i,j}, \;\; \mathbf{A} \in \mathcal{T}$$

where $\mathcal{T}$ is the set of all possible adjacency matrices corresponding to trees with n nodes. Fast computation of a minimum weight spanning tree can be accomplished with Prim's algorithm [Prim, 1957].

Embedding Networks with MVE

If the input Gram matrix $\mathbf{W}$ comes directly from the weights of a sparse weighted graph or network, we can use $\mathbf{W}$ directly with MVE and set $A_{ij} = 1$ where $W_{ij} > 0$ and zero elsewhere. MVE provides a strong advantage (compared to MVU for example) when the connectivity structure is not a uniformly sampled mesh but takes on different structures, such as the "hubs and spokes" structure found in many network datasets, as illustrated by the star graph example in Figure 2.2 and discussed in Section 2.1.1. This advantage allows us to visualize many networks represented as weighted graphs by directly running MVE on the network using the sparse connectivity implied by $\mathbf{W}$ (instead of the connectivity from an algorithm such as k-nearest neighbors). Results on a coauthorship network are shown in Section 2.3. MVE can be used to embed trees as well. In this case it is useful to add additional constraints to the optimization to prevent the embedding from collapsing from sparse connectivity. An additional set of constraints is added that prevents any pairwise distances from shrinking:

$$K_{ii} + K_{jj} - 2K_{ij} \geq W_{ii} + W_{jj} - 2W_{ij} \quad \forall_{i,j} \text{ when } A_{ij} = 0.$$

We apply this formulation, called MVE-full, to phylogenetic data and provide a comparison with other methods in Section 2.3.

2.2 Preserving Structure

It is crucial to preserve the important relationships in the data as specified by **A**. We consider a variety of different methods for formulating a set of linear constraints on **K** to preserve the structure of **A** during embedding. The simplest method is to preserve local distances: adding a constraint for each edge in **A** that specifies that the distance between a pair of points must be preserved before and after the embedding

$$K_{ii} + K_{jj} - 2K_{ij} = W_{ii} + W_{jj} - 2W_{ij} \quad \forall_{i,j} \quad \text{s.t.} \quad A_{ij} = 1.$$

A stronger condition is to preserve local isometry, preserving not only local distances but also local angles [Weinberger et al., 2004]. This corresponds to preserving the distances between neighbors, and between the neighbors of neighbors as well:

$$K_{ii} + K_{jj} - 2K_{ij} = W_{ii} + W_{jj} - 2W_{ij} \ \forall_{i,j} \ \text{s.t.} \ A_{ij} = 1 \ \text{or} \ \exists_k \ \text{where} \ A_{ik} = 1 \ \text{and} \ A_{kj} = 1.$$

Despite preserving local distances and local angles, a connectivity algorithm may not recover the same **A** from both the original data and the embedding. We refer to satisfying this stricter constraint as preserving the topology of the input data and provide more details on preserving graph topology with MVE in Section 4 after we have introduced structure preserving embedding in the next section.

2.3 MVE Experiments

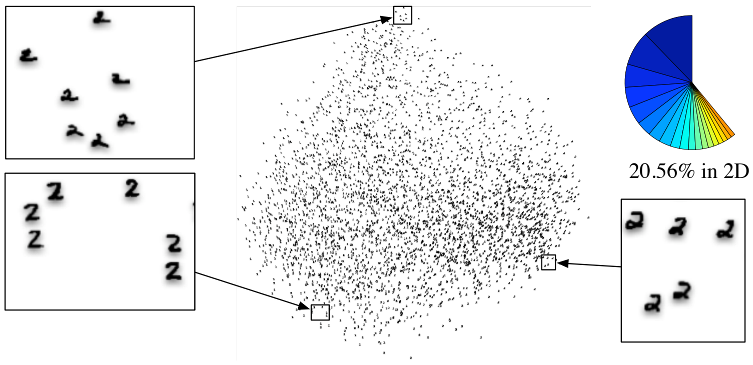
The following experiments compare MVE, MVU, and PCA on a variety of synthetic and real-world datasets, including images of handwritten twos, a coauthorship network, phylogenetic trees, and search engine data. We provide a qualitative comparison of the visualizations as well as quantitative comparison of the spectral fidelity of the different embeddings. A good visualization of data should preserve the local pairwise relationships as precisely as possible. In many cases, visualizing data in 2D requires truncating extra dimensions with useful information, and thus pairwise local distances in the 2D space are less accurately represented. Therefore, the spectral fidelity of an embedding, the amount of the variance that is captured in 2D, is a strong indicator of how accurately the visualization preserves local distances. Note that in these experiments, PCA, MVU, and MVE all preserve local distances exactly in terms of **K**, however algorithms that score higher in terms of spectral fidelity for $d = 2$ better represent **K** using only 2 dimensions and thus better represent the original local distances. In Table 2.1 we provide a summary of the spectral fidelity scores of all the experiments which directly compare PCA, MVU, and MVE. For all MVE experiments we set $\beta = 2$, and use a linear kernel function for both MVE and MVU.

	PCA	MVU	MVE
Swiss Roll	72.5%	99.9%	**100**%
Star Graph	95.0%	29.9%	**100**%
Torn S-Curve	97.3%	96.7%	**100**%
Handwritten Twos	20.6%	80.8%	**98.8**%
Coauthorship Network	5.9%	95.3%	**99.2**%

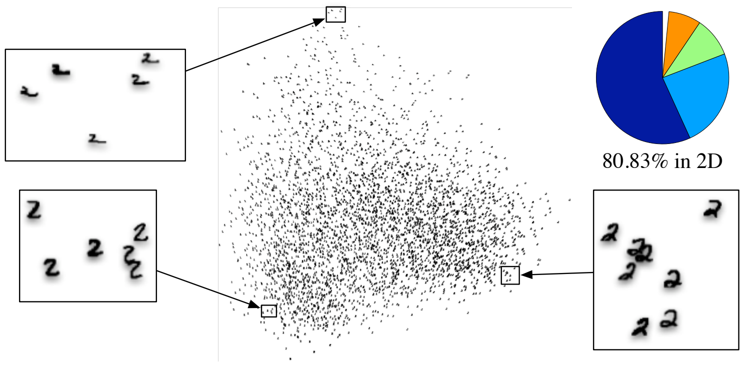
Table 2.1: Fidelity of 2D embeddings as percentage of total eigenvalue summation for PCA, MVU, and MVE. MVE consistently captures more of the variance of the data in 2D. The maximum spectral fidelity score is shown in bold.

In Figure 2.5 we see 4000 images of handwritten twos from the MNIST dataset [LeCun et al., 2001]. For MVU and MVE the connectivity matrix was constructed using b-matching graph with b=4. Considering the MVE embedding more closely, we see that on the right there are twos with looped bottoms, where on the left they look like the letter ‘z’, and at the top they become squashed. The eigenspectrum of the data shows that PCA needs many dimensions for embedding, and thus the top 2D projection shown here is less informative than the embeddings from MVU and MVE. Similarly, MVU only captures 81% of the variance in 2D while MVE captures almost 99% offering a more accurate 2D visualization.

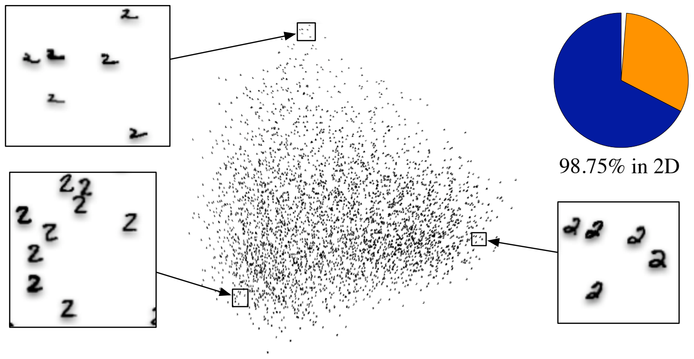
In Figure 2.6 we show a coauthorship network of 379 scientists [Newman, 2006]. In this experiment, we are given as input a score for how often each pair of authors publish together. These scores define a sparse weighted graph that can be directly embedded with



(a) PCA



(b) MVU



(c) MVE

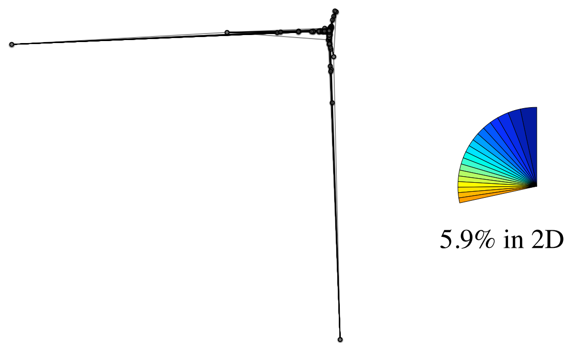
Figure 2.5: 4000 images of handwritten twos from the MNIST dataset embedded with PCA, MVU, and MVE. Next to each method is the normalized eigenspectrum represented as a pie chart, and the percent of the variance captured in 2D. With a spectral fidelity score of 98.75%, MVE almost perfectly represents the original local distances using only 2 dimensions.

MVE and MVU without using a kernel or connectivity algorithm. Similarly, PCA is also performed directly on the sparse weighted graph. The eigenspectrum for PCA indicates that the algorithm requires many dimensions to represent the data, and the corresponding embedding clearly shows each node being puffed out into a high-dimensional space. The MVU embedding captures a large amount of the variance of the data; however, MVU appears to collapse many of the subgroups, such as the one on the far right, where the MVE embedding represents this data better in 2D, capturing almost all of the variance described by the original graph. An expanded plot of the MVE embedding with author labels is shown in Figure 2.9.

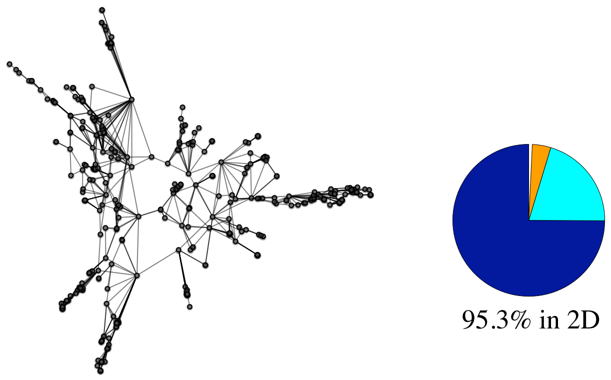
Figures 2.7 and 2.8 show the results of visualizing the phylogenetic trees of 30 species of salamanders and 56 species of crustaceans. The data consists of features which describe the external anatomy of each species, as well the phylogenetic tree [Kemp et al., 2003]. We use a linear kernel function on the anatomy features as input, and the phylogenetic tree as connectivity. It is clear that MVE-full provides the best visualization, preserving 98.7% of the variance in 2D. Figure 2.10 compares the k-nearest neighbors and b-matching connectivity algorithms using MVU and MVE. The dataset consists of the top 200 search terms for AOL [Sadetsky, 2006], and the features are the frequency of searches by over 100,000 users. The parameters k and b were both set to 4 for this experiment. We see that when using kNN, certain search terms (such as "billing" in this example) become highly connected hubs, causing MVU and MVE to crowd other search terms around these popular nodes. The b-matching constraints however, force each node to have precisely b neighbors, and thus yields a more regular conectivity and a smoother manifold of search terms. In the MVE b-matching manifold, shopping search terms are placed in the bottom left, and travel/flight related terms are placed in the upper left. In the other embeddings, these natural groupings are harder to identify.

2.3.1 An Application to Spatiotemporal Data

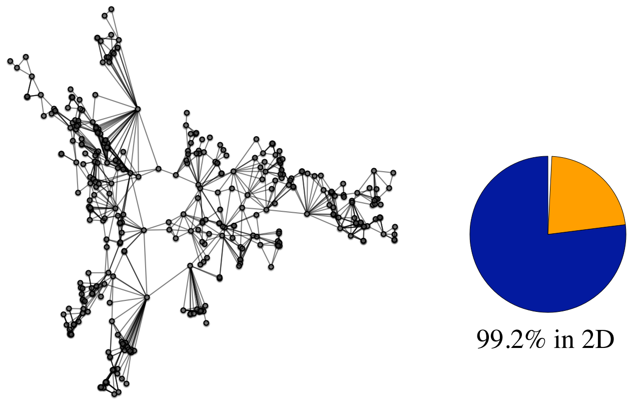
We next consider an application of manifold learning to spatiotemporal data. On average over 500,000 yellow taxi cab trips are recorded every day in New York City, capturing the collective movement patterns of millions of individuals. This massive high-dimensional



(a) PCA

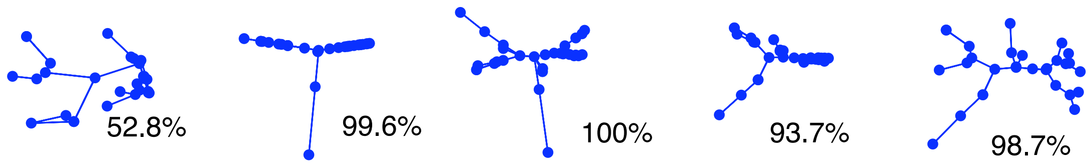


(b) MVU



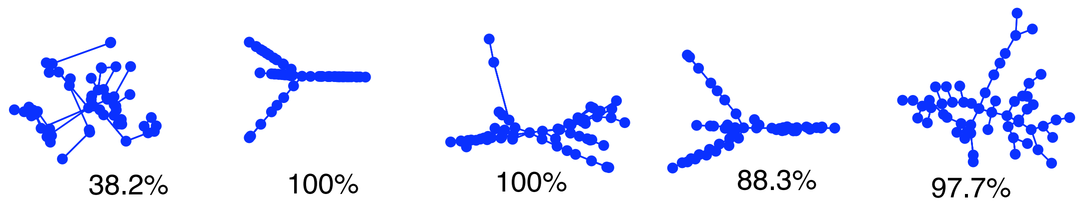
(c) MVE

Figure 2.6: A coauthorship network of 379 scientists in network theory embedded by PCA, MVU, and MVE. We see that PCA is inadequate for visualizing the structure of this weighted graph, capturing only 5.9% of the variance in 2D. MVU captures a significant amount of variance, but collapses and folds offshoots of the main network. MVE provides the clearest visualization of this dataset, capturing the relationships in the data almost perfectly in 2D.



(a) PCA (b) MVU (c) MVE (d) MVU-full (e) MVE-full

Figure 2.7: Embeddings of the phylogenetic tree for 30 species of salamanders.



(a) PCA (b) MVU (c) MVE (d) MVU-full (e) MVE-full

Figure 2.8: Embeddings of the phylogenetic tree for 56 species of crustaceans.

dataset necessitates tools such as dimensionality reduction and clustering algorithms in order to better understand the flow patterns of an urban area. Each city block can be described by a high-dimensional vector of counts representing how much flow there is to and from all other city blocks for each hour of the week. From these high-dimensional vectors we can apply Minimum Volume Embedding (MVE) [Shaw and Jebara, 2007] or spectral clustering [Ng et al., 2001] to visualize the similarities between places. Furthermore, we can identify the hubs and authorities of the flow network. Similar to how PageRank [Page et al., 1999] finds the most authoritative places on the web, we can compute PlaceRank to find authoritative places in the physical world. These algorithms applied to this spatiotemporal dataset offer a unique perspective on the collective behavior of millions of people moving around New York City.

The input dataset consists of the latitude, longitude, and timestamps for the start and endpoints of 22.5 million New York taxi cab trips spanning 6 months between January and June 2009. For our analysis we consider the 2000 busiest city blocks, each of which has a minimum of 20 pickups or dropoffs per day. The flow data is represented as a directed adjacency matrix $\mathbf{A}^t \in \mathbb{R}^{n\times n}$ for each of the $T = 168$ hours in a week where each entry $A^t_{i,j}$ counts the number of trips from block i to block j in weekhour t. The similarity between places can be expressed as a linear kernel: $\mathbf{W} = \sum_{t=1}^{T} \frac{1}{2}\left(\mathbf{A}^t\mathbf{A}^{t\top} + \mathbf{A}^{t\top}\mathbf{A}^t\right)$, where two places are considered similar if they have similar amounts of flow to other places at similar times of the week. The flow data is publicly available at `http://www.metablake.com/data/nyflow/`.

Given $\mathbf{W}$, we can apply Minimum Volume Embedding (MVE) to learn low-dimensional coordinates for each place $\mathbf{y}_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$ which best capture the variance of the original high-dimensional data where each place is described by $D = 672,000$ flow features corresponding to the in/out flow for all 2000 blocks stratified by the 168 weekhours ($D = 672,000 = 2 \times 2000 \times 168$). When applying MVE with $d = 3$, we find that over 99% of the variance of these original features can be preserved. In Figure 2.11(a) we translate these three coordinates to RGB to color code each city block, allowing us to visualize the smooth modes of variation in the flow patterns of places in the city. We see a clear uptown vs. downtown distinction (light green vs. pink), and major hubs such as Times Square and Grand Central station (red) clearly standing out from the areas around them. Figures 2.11(b) and 2.11(c) show the result of applying spectral clustering to the Gram matrix $\mathbf{W}$ to assign one of $k = 12$ labels to each place, thus visualizing the natural neighborhoods that emerge out of the data. It is suprising that this map looks very similar to a neighborhood map of New York City without including any explicit neighborhood information. Figures 2.11(d) and 2.11(e) show heat maps representing the hubs and authorities in the cab flow network which were computed by finding the stationary vector of the total flow adjacency matrix $\mathbf{A} = \sum_{t=1}^{T} \mathbf{A}^t$ and its transpose using the power method. Places with high authority values have inbound traffic from many other high authority places; similarly places with high hub values have outbound traffic to many other hubs. The algorithms in Figure 2.3.1 provide a set of complementary views of this unique high-dimensional dataset, allowing us to easily visualize patterns in the flow of taxis between New York City blocks.

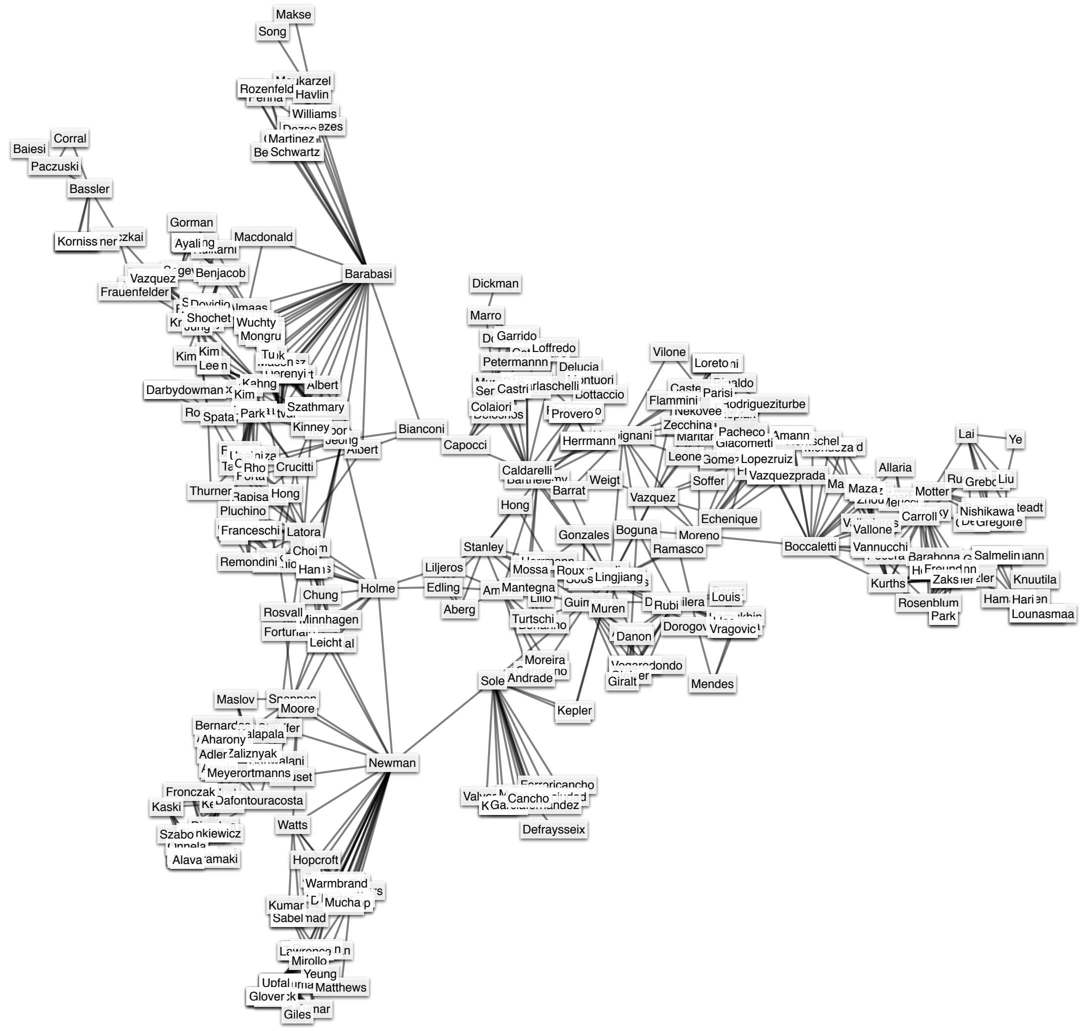
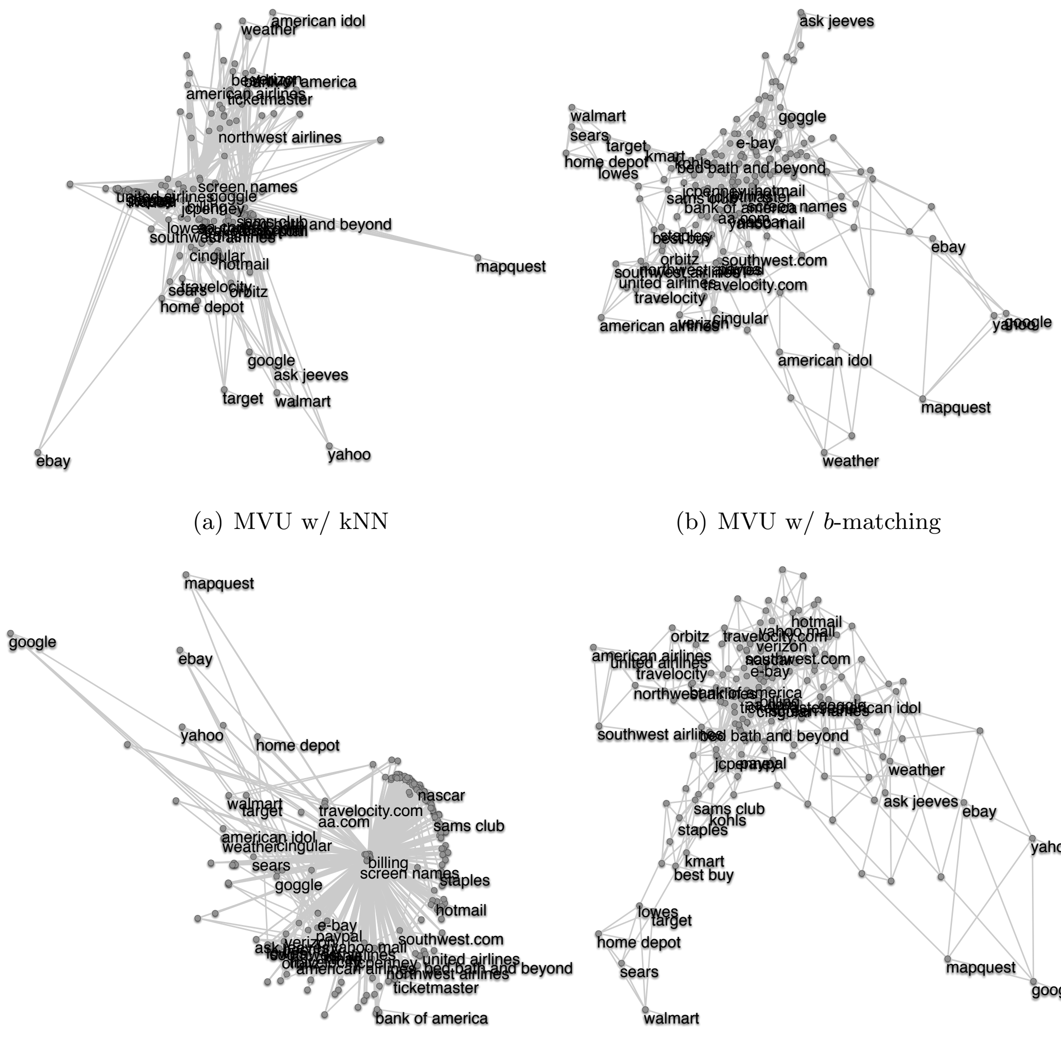


Figure 2.9: MVE embedding with author labels of the network science coauthorship data shown in Figure 2.6. Note the clear major hubs in the network, such as Barabasi and Newman.



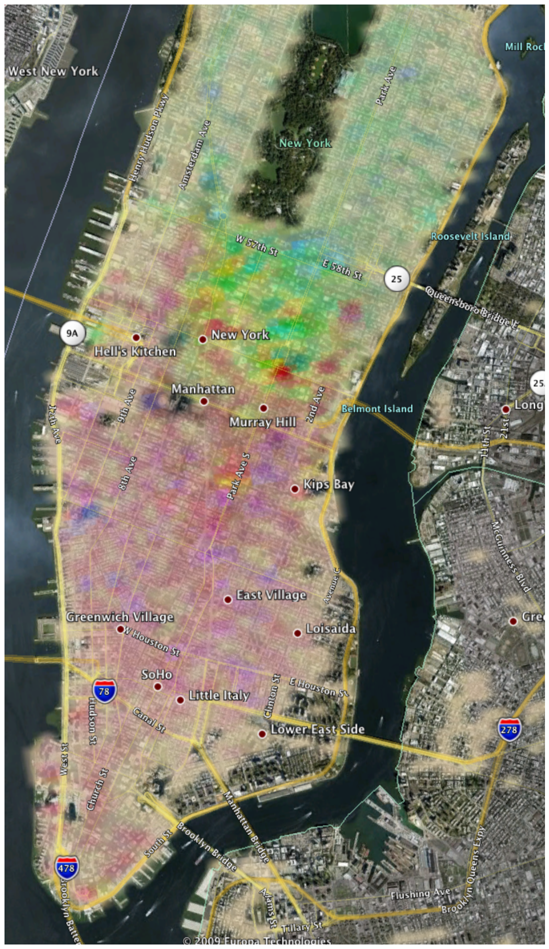
(a) MVU w/ kNN

(b) MVU w/ b-matching

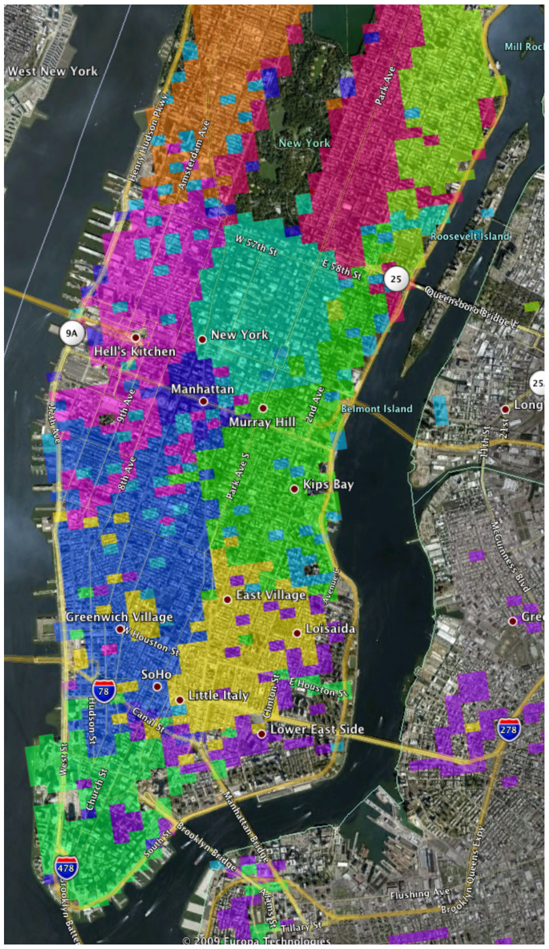
(c) MVE w/ kNN

(d) MVE w/ b-matching

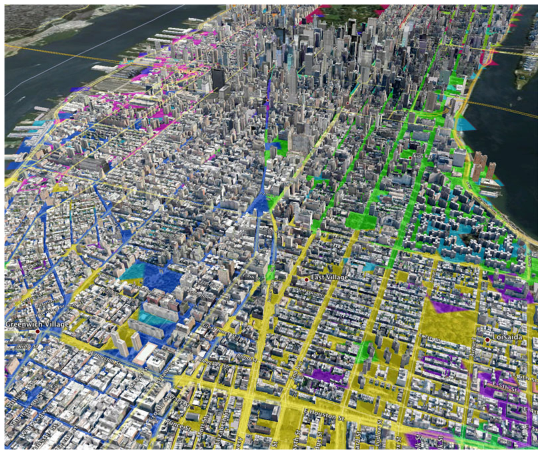
Figure 2.10: Embeddings of the top 200 AOL search terms. 40 of the search terms have been labeled. This figure compares MVU and MVE with k-nearest neighbors and b-matching as connectivity algorithms. Note that highly connected popular nodes can dominate the kNN embedding causing a hub and spokes structure, where the b-matching constraints create regular graphs and smoother manifolds.



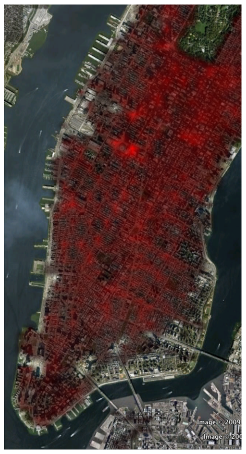
(a) MVE



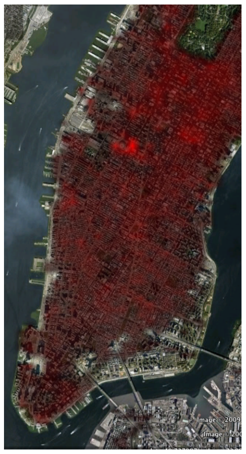
(b) Spectral Clustering



(c) Spectral Clustering (downtown)



(d) Hubs



(e) Authorities

Figure 2.11: Minimum Volume Embedding, Spectral Clustering and PlaceRank Hubs and Authorities applied to NYC taxi data

Chapter 3

Structure Preserving Embedding

To embed graphs which are unweighted and undirected, we need to formulate new objectives and constraints. Because we are given only a binary adjacency matrix as input $\mathbf{A} \in \mathbb{B}^{n\times n}$, there are no distances to preserve and thus MVE is not suited for this task. To address this setting we present structure preserving embedding (SPE). In Section 3.1, we formalize the concept of preserving graph structure as opposed to local distances, and show how structure preserving constraints can be defined as a set of linear inequalities for a variety of different connectivity algorithms. In Section 3.2 we derive an objective function suited to embedding unweighted undirected graphs which favors low-dimensional embeddings close to the spectral embedding solution. The SPE algorithm is built using the SDP + SVD framework, combining the low-dimensional objective with structure preserving constraints. Section 3.3 describes the SPE algorithm in detail, and Section 3.4 shows experiments using SPE to embed a variety of real and synthetic graphs.

3.1 Preserving Graph Topology with Linear Constraints

Our goal is to learn an embedding which preserves the topology of an unweighted undirected input graph defined as an adjacency matrix $\mathbf{A} \in \mathbb{B}^{n\times n}$. The embedding is represented by a positive semidefinite Gram matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ whose spectral decomposition yields the low-dimensional embedding . We propose the following definition for preserving graph structure.

Definition 7 *Given a graph with adjacency matrix* $\mathbf{A}$*, an embedding represented as a Gram matrix* $\mathbf{K}$ *is structure preserving with respect to a connectivity algorithm* $\mathcal{G}$*, if* $\mathcal{G}(\mathbf{K}) = \mathbf{A}$*.*

The connectivity algorithm $\mathcal{G} : \mathbb{R}^{n\times n} \to \mathbb{B}^{n\times n}$ accepts as input a kernel matrix $\mathbf{K}$, and outputs an adjacency matrix $\tilde{\mathbf{A}} = \mathcal{G}(\mathbf{K})$. Several popular choices for $\mathcal{G}$ include k-nearest neighbors, epsilon-balls, maximum weight spanning trees, and maximum weight generalized matching (also known as b-matching). The following sections show that for these choices of $\mathcal{G}$ it is possible to enumerate *linear* constraints on $\mathbf{K}$ to ensure that $\mathcal{G}(\mathbf{K}) = \mathbf{A}$.

3.1.1 Nearest Neighbor Graphs

Preserving the structure of graphs created by nearest neighbor algorithms requires enumerating only a small set of linear constraints on $\mathbf{K}$. The k-nearest neighbor algorithm ($k$nn) discussed in Section 2.1.5 connects each node to the k neighbors to which the node has shortest distance, where k is an input parameter; therefore, we must enforce that for each node the distances to all other nodes to which it is not connected must be larger than the distance to the furthest connected neighbor of that node:

$$D_{ij} > (1 - A_{ij}) \max_m (A_{im} D_{im}).$$

Recall Definition 6 which states that the distance between a pair of points (i, j) with respect to a given positive semidefinite kernel matrix is a linear function of $\mathbf{K}$: $D_{ij} = K_{ii} + K_{jj} - 2K_{ij}$. Similarly, preserving an ϵ-neighborhood graph can be achieved with the linear constraints on $\mathbf{K}$:

$$D_{ij}(A_{ij} - \frac{1}{2}) \leq \epsilon(A_{ij} - \frac{1}{2}).$$

If for each node the connected distances are less than the unconnected distances (or some ϵ), a greedy algorithm such as $k$nn or ϵ-balls will find the exact same neighbors for that node, and thus the connectivity computed from $\mathbf{K}$ is exactly $\mathbf{A}$.

3.1.2 Maximum Weight Subgraphs

Unlike nearest neighbor algorithms which select edges greedily for each node, maximum weight subgraph algorithms select edges from a weighted graph to produce a subgraph which has total maximal weight [Fremuth-Paeger and Jungnickel, 1999].

Definition 8 *Given a kernel matrix* $\mathbf{K}$*, define the weight between two points* (i, j) *as the negated pairwise distance between them:* $Z_{ij} = -D_{ij} = -K_{ii} - K_{jj} + 2K_{ij}$.

For example, b-matching finds the maximum weight subgraph while also enforcing that every node has a fixed degree b_i for $i = 1...n$:

$$\begin{aligned}\mathcal{G}(\mathbf{K}) = \quad & \underset{\tilde{\mathbf{A}}}{\operatorname{argmax}} \sum_{ij} Z_{ij} \tilde{A}_{ij} \\ & \text{s.t.} \ \sum_j \tilde{A}_{ij} = b_i, \quad \tilde{A}_{ij} = \tilde{A}_{ji}, \quad \tilde{A}_{ii} = 0, \quad \tilde{A}_{ij} \in \mathbb{B}.\end{aligned}$$

Similarly, a maximum weight spanning tree algorithm finds the maximum weight subgraph such that $\mathcal{G}(\mathbf{K}) \in \mathcal{T}$, where $\mathcal{T}$ is the set of all tree graphs:

$$\mathcal{G}(\mathbf{K}) \ = \ \underset{\tilde{\mathbf{A}}}{\operatorname{argmax}} \sum_{ij} Z_{ij} \tilde{A}_{ij} \ \text{ s.t. } \ \tilde{\mathbf{A}} \in \mathcal{T}.$$

Once again, linear constraints force $\mathbf{K}$ to preserve the structure of the input graph. Unfortunately, for such algorithms, we must avoid enumerating too many constraints of the form:

$$\sum_{ij} Z_{ij} A_{ij} \geq \sum_{ij} Z_{ij} \tilde{A}_{ij}.$$

In order to avoid an exponential enumeration, the most violated inequalities can be introduced sequentially using a cutting plane approach discussed in more detail in Section 3.3.

3.2 A Low-Rank Objective Function

Popular graph embedding methods such as spectral embedding and Laplacian eigenmaps employ spectral decomposition of the adjacency matrix $\mathbf{A}$, graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, or normalized graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D} = \operatorname{diag}(\mathbf{A}\mathbf{1})$ [Belkin and Niyogi, 2002, Koren, 2005]. Spectral embedding uses the leading d eigenvectors of $\mathbf{A}$ as coordinates, where Laplacian eigenmaps uses the d eigenvectors of $\mathbf{L}$ with the smallest non-zero eigenvalues. Note that for regular graphs, embeddings computed from the graph Laplacian or the adjacency matrix are equivalent. These techniques may produce embeddings with few coordinates by using the most dominant eigenvectors first; however, in practice the matrices $\mathbf{A}$ or $\mathbf{L}$ are often not low rank, and many eigenvectors have non-zero eigenvalues.

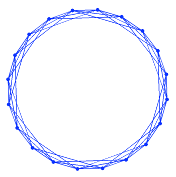
Similar to these spectral methods, we would like to decompose $\mathbf{K}$ and use the top d eigenvectors as coordinates. The linear structure-preserving constraints from the previous section define a convex hull of possible matrices $\mathbf{K}$. From this set we would like to choose a $\mathbf{K}$ that is low-rank. Thus when we use the eigenvectors of $\mathbf{K}$ as coordinates for the nodes, only low-dimensional coordinates will be necessary. Consider choosing the objective function $\max_{\mathbf{K}\succeq 0} \operatorname{tr}(\mathbf{KA})$. When optimized without structure preserving constraints, it yields a $\mathbf{K}$ that agrees exactly with a low-rank version of spectral embedding, using the leading eigenvector of $\mathbf{A}$ as the embedding. We have not yet introduced structure preserving constraints, and are only limiting the trace norm of $\mathbf{K}$ to avoid the objective function from growing unboundedly, and constraining $\mathbf{K}$ to be positive semidefinite. We claim that this objective function attempts to recover a low-rank version of spectral embedding. The proof of the following theorem can be found in 8.1.3.

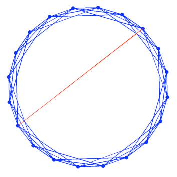
Theorem 9 *For any symmetric binary matrix $\mathbf{A}$, such that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}$, $\lambda_i \geq \lambda_{i+1}$, $A_{ii} = 0$ for $i = 1...n$, if $\lambda_1 > \lambda_i$ for $i = 2...n$, the objective function $\max_{\mathbf{K}\succeq 0} \operatorname{tr}(\mathbf{KA})$ subject to $\operatorname{tr}(\mathbf{K}) \leq 1$ is optimized by setting $\mathbf{K} = \mathbf{v}_1\mathbf{v}_1^\top$. If there exists an i such that $\lambda_1 = \lambda_i$, the objective function will recover $\mathbf{K} = \sum_{i|\lambda_i=\lambda_1} \beta_i \mathbf{v}_i \mathbf{v}_i^\top$ where $\beta_i \geq 0$ for $i = 1...n$.*

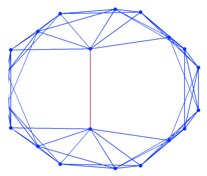
In practice, optimizing this objective function without constraints is equivalent to running spectral embedding and using the top eigenvector as the embedding coordinates, or if there are ties for the top eigenvector using a conic combination of the top eigenvectors. In the next section we will see that when structure preserving constraints are introduced, the embedding will be corrected, typically requiring more than 1 dimension, but ensuring that the structure of the input graph can be reconstructed directly from the embedding.

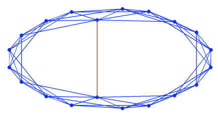
3.3 Structure Preserving Embedding Algorithm

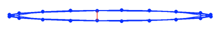
Structure preserving embedding, like MVE, employs a semidefinite program (SDP) to learn $\mathbf{K}$, and then decomposes $\mathbf{K}$ with a spectral decomposition to find the embedding coordinates


(a) Ring Graph


(b) w/ noise


(c) $C = 1000$


(d) $C = 5$


(e) $C = 2.5$

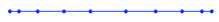

(f) $C = 0$

Figure 3.1: By adjusting the input paramter C for Algorithm 2, SPE is able to handle noisy graphs. From left to right and top to bottom, we see a perfect ring graph embedded by SPE, a noisy line added to the graph at random , and then the results of using SPE on the noisy graph with C set to 1000, 5, 2.5, and 1. Note when C is small, SPE reproduces the rank-1 spectral embedding.

$\mathbf{y}_i \in \mathbb{R}^d$ for $i = 1, \ldots, n$. The SDP has constraints $\mathcal{K} = \{\mathbf{K} \succeq 0, \mathrm{tr}(\mathbf{K}) \leq 1, \sum_{\mathrm{ij}} \mathrm{K}_{\mathrm{ij}} = 0\}$ which limit the trace norm of $\mathbf{K}$ and ensure $\mathbf{K}$ is centered. Depending on the choice of connectivity algorithm $\mathcal{G}$, structure preserving constraints are introduced either initially (for greedy methods such as k-nearest neighbors), or iteratively using a cutting-plane method (for maximum weight subgraph methods). All constraints other than the semidefinite constraint are linear in $\mathbf{K}$ and share a global slack variable ξ. The convex objective function of the SDP $\max_{\mathbf{K} \in \mathcal{K}, \xi \geq 0}\ \mathrm{tr}(\mathbf{KA}) - \mathrm{C}\xi$ ensures a low-rank $\mathbf{K}$ that will have few dominant eigenvectors and thus d will typically be small. The additional term $C\xi$ allows some violations of the structure preserving constraints, and the use of a finite C assures a solution to the SDP is always possible. As shown with other cutting-plane techniques [Finley and Joachims, 2008], the global slack variable ξ encourages fast convergence and helps avoid numerical problems. Figure 3.1 illustrates the effects of varying C on a synthetic graph with added noise.

When the connectivity algorithm $\mathcal{G}$ is k-nearest neighbors, the SDP does not need to be

Algorithm 2 Structure Preserving Embedding for k-nearest neighbor constraints.

```
Input: $\mathbf{A} \in \mathbb{B}^{n\times n}$, connectivity algorithm $\mathcal{G}$, and parameter $C$
1: $\mathcal{K} \leftarrow \{\mathbf{K} \succeq 0, \mathrm{tr}(\mathbf{K}) \leq 1, \sum_{\mathrm{ij}} \mathrm{K_{ij}} = 0\}$
2: Solve SDP $\tilde{\mathbf{K}} = \arg\max_{\mathbf{K}\in\mathcal{K},\xi\geq 0} \mathrm{tr}(\mathbf{KA}) - \mathrm{C}\xi$ s.t. $D_{ij} > (1 - A_{ij}) \max_m(A_{im}D_{im}) - \xi$
3: Perform SVD on $\hat{\mathbf{K}}$ to compute the $n$ leading eigenvectors $\hat{\mathbf{v}}_i$ and corresponding eigenvalues $\hat{\lambda}_i$, and set $\mathbf{y}_i \leftarrow \sqrt{\hat{\lambda}_i}\hat{\mathbf{v}}_i$ for $i = 1, \ldots, n$
4: return $\mathbf{y}_1, \ldots, \mathbf{y}_n$
```

iterated. Algorithm 2 outlines the two-step algorithm. When the connectivity algorithm $\mathcal{G}$ is a maximum weight subgraph method (such as b-matching), we need to select constraints from a possibly exponentially large set of the form:

$$\mathrm{tr}(\mathbf{ZA}) - \mathrm{tr}(\mathbf{Z}\tilde{\mathbf{A}}) \geq \triangle(\tilde{\mathbf{A}}, \mathbf{A}) - \xi \quad \forall \tilde{\mathbf{A}} \in \mathcal{G}$$

where $\triangle(\tilde{\mathbf{A}}, \mathbf{A}) = \frac{1}{n^2} \sum_{ij} |\tilde{A}_{ij} - A_{ij}|$, and $\tilde{\mathbf{A}} \in \mathcal{G}$ states that $\tilde{\mathbf{A}}$ is in the family of graphs formed by a connectivity algorithm $\mathcal{G}$, such as b-matchings or trees. Fundamentally this constraint is enforcing that the true embedding solution (represented here as $\mathbf{Z}$ which is a linear function of $\mathbf{K}$) produces a higher weight match with the input graph than with any other graphs by some margin $\triangle(\tilde{\mathbf{A}}, \mathbf{A})$. We select and add these cutting-plane constraints sequentially, first running the optimization without any constraints, (yielding the rank-1 spectral embedding solution), and then adding the most violated constraint and running the optimization again. We can find the most violated constraint at each iteration by computing the adjacency matrix $\tilde{\mathbf{A}}$ that maximizes $\mathrm{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{A}})$ s.t. $\tilde{\mathbf{A}} \in \mathcal{G}$ using a maximum weight subgraph method, where $\tilde{\mathbf{Z}}$ is computed from $\tilde{\mathbf{K}}$, the learned kernel matrix from the previous iteration. Once we have $\tilde{\mathbf{A}}$ we can introduce the constraint

$$(\mathrm{tr}(\mathbf{ZA}) - \mathrm{tr}(\mathbf{Z}\tilde{\mathbf{A}})) \geq \triangle(\tilde{\mathbf{A}}, \mathbf{A}) - \xi.$$

The algorithm converges when

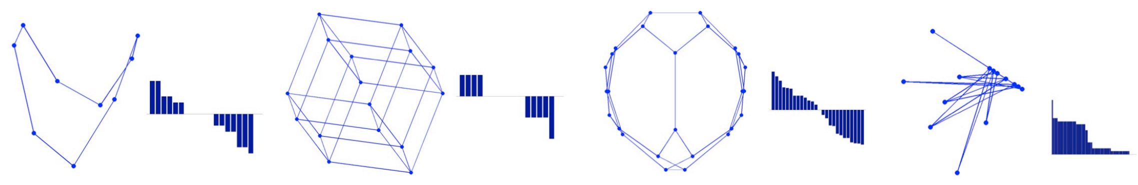
$$|\mathrm{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{A}}) - \mathrm{tr}(\tilde{\mathbf{Z}}\mathbf{A})| \leq \kappa,$$

where κ is an input parameter. Algorithm 3 outlines the steps of SPE with cutting plane constraints.

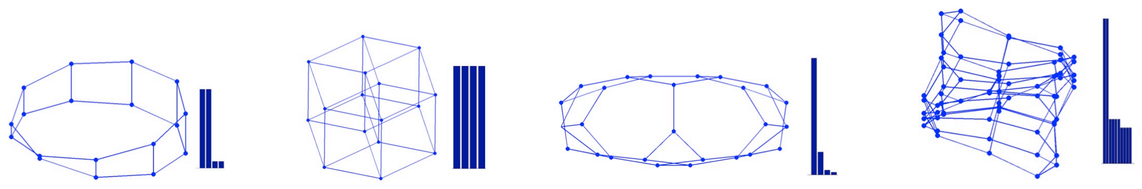
Algorithm 3 Structure Preserving Embedding with cutting-plane constraints

```
Input: $\mathbf{A} \in \mathbb{B}^{n \times n}$, connectivity algorithm $\mathcal{G}$, and parameters $C, \kappa$
1: $\mathcal{K} \leftarrow \{\mathbf{K} \succeq 0, \mathrm{tr}(\mathbf{K}) \leq 1, \sum_{\mathrm{ij}} \mathrm{K_{ij}} = 0\}$
2: repeat
3:    $\tilde{\mathbf{K}} \leftarrow \mathrm{argmax}_{\mathbf{K} \in \mathcal{K}, \xi \geq 0} \mathrm{tr}(\mathbf{KA}) - \mathrm{C}\xi$ {Found via SDP}
4:    $\tilde{\mathbf{Z}} \leftarrow 2\tilde{\mathbf{K}} - \mathrm{diag}(\tilde{\mathbf{K}})\mathbf{1}^\top - \mathbf{1}\mathrm{diag}(\tilde{\mathbf{K}})^\top$
5:    $\tilde{\mathbf{A}} \leftarrow \mathrm{argmax}_{\mathbf{A}} \mathrm{tr}(\tilde{\mathbf{Z}}\mathbf{A})$ s.t. $\tilde{\mathbf{A}} \in \mathcal{G}$ {Find biggest violator}
6:    if $|\mathrm{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{A}}) - \mathrm{tr}(\tilde{\mathbf{Z}}\mathbf{A})| \geq \kappa$ then
7:       add constraint $\mathrm{tr}(\mathbf{ZA}) - \mathrm{tr}(\mathbf{Z}\tilde{\mathbf{A}}) > \triangle(\tilde{\mathbf{A}}, \mathbf{A}) - \xi$
8:    end if
9: until $|\mathrm{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{A}}) - \mathrm{tr}(\tilde{\mathbf{Z}}\mathbf{A})| \leq \kappa$
10: Perform SVD on $\hat{\mathbf{K}}$ to compute the $n$ leading eigenvectors $\hat{\mathbf{v}}_i$ and corresponding eigen-
    values $\hat{\lambda}_i$, and set $\mathbf{y}_i \leftarrow \sqrt{\hat{\lambda}_i}\hat{\mathbf{v}}_i$ for $i = 1, \ldots, n$
11: return $\mathbf{y}_1, \ldots, \mathbf{y}_n$
```

The complexity of the semidefinite program is $O(n^3 + \mathcal{C}^3)$ where $\mathcal{C}$ denotes the number of constraints. For k-nearest neighbor graphs there are many constraints, $\mathcal{C}$ scales with the number of edges in the input graph; however, in practice a large number of these constraints are inactive, and therefore a working set method is effective. For maximum weight subgraphs, we add constraints iteratively, and therefore only a small set of cutting plane constraints need to be enforced. It has been shown that cutting plane algorithms with quadratic objectives and linear constraints converge in polynomial time [Finley and Joachims, 2008]; unfortunately, these guarantees do not immediately apply to our semidefinite program; however, cutting plane algorithms have been successfully deployed in settings beyond structured prediction and quadratic programming [Sontag and Jaakkola, 2008]. We have been able to run SPE on graphs with approximately 1000 nodes and 10000 edges. SPE is implemented in MATLAB, and uses CSDP and SDP-LR as the underlying SDP solvers for the greedy and iterative versions respectively [Burer and Monteiro, 2003]. The SDP-LR solver is a natural fit for our low-rank objective, recasting the optimization over $\mathbf{K} \in \mathbb{R}^{n \times n}$ as an optimization over $\mathbf{R} \in \mathbb{R}^{n \times d}$ where $\mathbf{K} = \mathbf{R}^\top\mathbf{R}$ and $d < N$.



(a) Möbius Ladder (b) Tesseract (c) Celmins Swart Snark (d) Balaban 10-cage

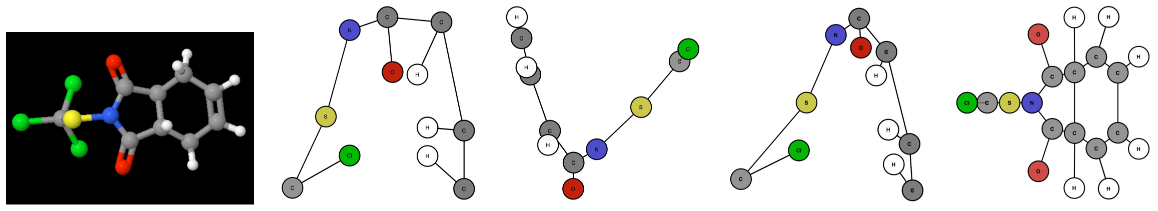


(e) Möbius Ladder (f) Tesseract (g) Celmins Swart Snark (h) Balaban 10-cage

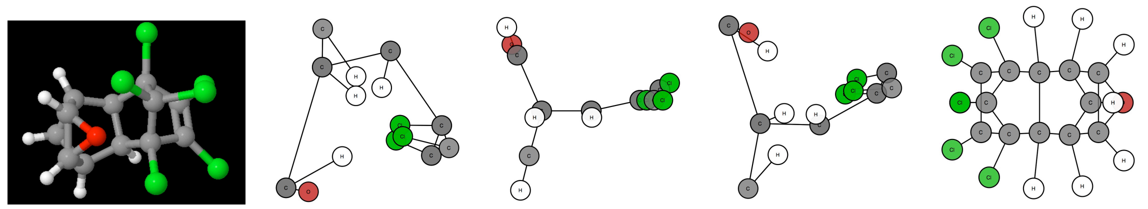
Figure 3.2: Classical graphs embedded with spectral embedding (top row a-d), and embedded with SPE w/ kNN (bottom row e-h). Eigenspectra are shown to the right. SPE finds a small number of dimensions that highlight many of the symmetries of these graphs.

3.4 SPE Experiments

We validate SPE by comparing it to purely spectral methods for visualizing a variety of synthetic and real-world datasets. Spectral methods often find many dominant eigenvectors, and thus the top 2D projection hides many meaningful coordinates which indicate the structure of the input graph. Figure 3.2 shows a variety of classical graphs visualized by SPE and spectral embedding. The corresponding eigenspectrum is shown next to each embedding. We see that the SPE embeddings are compact and visually informative. In Figure 3.3 we apply SPE to visualizing molecules. The physical 3D embedding of each molecule is shown on the left. The goal of the embedding algorithms is to embed the data in 2D using only the connectivity information of the molecule (which atoms are bonded together). SPE is able to find coordinates for the atoms which more clearly resemble the true physical structure of the molecule. Figures 3.2 and Figure 3.3 were created using SPE with k-nearest neighbor constraints. Figure 3.4 shows a visualization of the links between 981 political blogs [Adamic and Glance, 2005]. Because of the scale-free degree distribution of links between these websites, Laplacian eigenmaps finds many possible dimensions, and



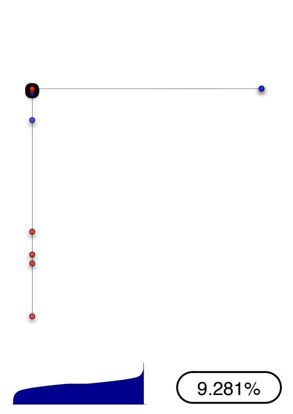
(a) Molecule TR015 (b) Spectral (c) Laplacian (d) Norm. Lap. (e) SPE



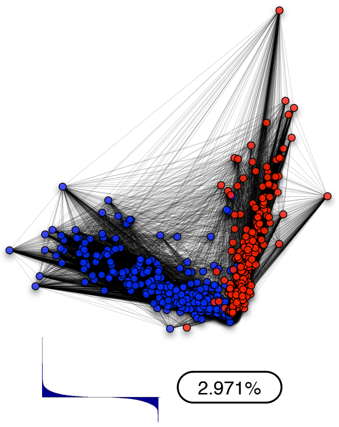
(f) Molecule TR012 (g) Spectral (h) Laplacian (i) Norm. Lap. (j) SPE

Figure 3.3: Two comparisons of molecule embeddings (top row and bottom). The SPE w/kNN embedding (right) more closely resembles the true physical embedding of the molecule (left), despite being given only connectivity information.

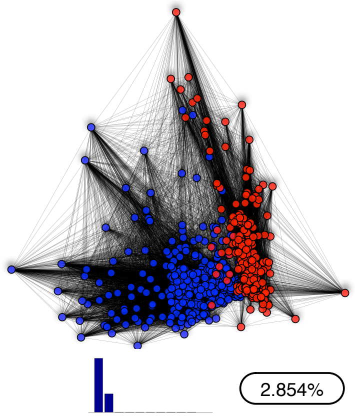
creates a high-dimensional hubs-and-spokes embedding, requiring nodes with high-degree to have neighbors very far away. Many of the dimensions found by spectral embedding are redundant; SPE is able to accurately represent the structure of this network using only 2 dimensions (as indicated by the eigenspectrum below each embedding). Also next to each eigenspectrum is the percent error from reconstructing the adjacency matrix using only 2 dimensions, where error is defined as the Hamming distance between the adjacency matrix reconstructed from the low-dimensional embedding, and the original adjacency matrix: $\triangle(\tilde{\mathbf{A}}, \mathbf{A}) = \frac{1}{n^2}\sum_{ij}|\tilde{A}_{ij} - A_{ij}|$. The SPE embedding was created using b-matching constraints.



(a) Normalized Laplacian



(b) Spectral Embedding



(c) SPE

Figure 3.4: The link structure of 981 political blogs. Conservative are labeled red, and liberal blue. Also shown is the % error of the adjacency matrix created from the 2D embedding, and the resulting eigenspectrum from each method. Not shown here is un-normalized Laplacian eigenmaps, whose embedding is similar to the normalized case and reconstruction error is 55.8%. Note that SPE is able to achieve the lowest error rate, representing the data using only 2 dimensions.

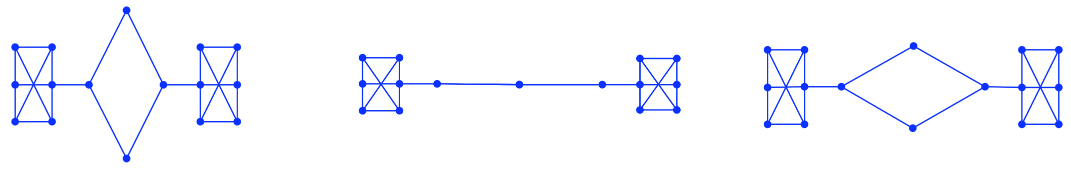
Chapter 4

Dimensionality Reduction with Structure Preserving Constraints

We have shown that MVE maximizes the spectral fidelity of a low-dimensional embedding, preserving a set of local distances while driving variance into the top d dimensions. Like other manifold learning methods for dimensionality reduction, MVE unfolds a graph built from reliable pairwise relationships in the data. During the unfolding of this graph, distances between connected nodes are constrained; however, distances between unconnected nodes are free to vary, and thus a graph built from the embedding might no longer match the graph built from the original data. Consider the toy problem illustrated in Figure 4.1. All distances along edges are preserved, but MVU (or similarly MVE) has collapsed pairs of points in such a way that they are now closer than the original neighbors of those points.

By modifying the MVE algorithm to include the structure preserving constraints developed for SPE, we can constrain the embeddings to preserve topology as well as local distances. Combining the distance preserving constraints from MVE with the k-nearest neighbor constraints from SPE yields the following new set of constraints for MVE+SP:

$$\mathcal{K} = \left\{ \forall \mathbf{K} \in \mathbb{R}^{N\times N} \;\middle|\; \begin{array}{l} \mathbf{K} \succeq 0 \\ \sum_{ij} K_{ij} = 0 \\ D_{ij} = W_{ii} + W_{jj} - W_{ij} - W_{ji} \quad \forall_{i,j} \text{ s.t. } A_{ij} = 1 \\ D_{ij} > (1 - A_{ij}) \max_m (A_{im} D_{im}) \quad \forall_{i,j} \end{array} \right\}. \tag{4.1}$$



(a) Original Barbell Graph (b) Embedded with MVU (c) Embedded with MVE+SP

Figure 4.1: The "barbell graph" (left) is a toy example which illustrates a common problem with unfolding-based methods for dimensionality reduction. In certain cases, unfolding can collapse parts of the graph on top of itself. We see that the diamond in the middle gets collapsed to a single point by MVU (middle). Adding graph topology preserving constraints (right) prevents the diamond from collapsing. Note MVE produces a similar embedding to MVU in this example.

Algorithm 4 Minimum Volume Embedding w/ Structure Preserving Constraints

Input: a Gram matrix $\mathbf{W} \in \mathbb{R}^{n \times n}$, sparse connectivity $\mathbf{A} \in \mathbb{B}^{n \times n}$, and parameters d, β, κ.

```
1: Initialize $\mathbf{K} \leftarrow \mathbf{W}$
2: $\mathcal{K} \leftarrow \{\mathbf{K} \succeq 0, \sum_{ij} K_{ij} = 0\}$
3: Add constraints to $\mathcal{K} : \{D_{ij} = W_{ii} + W_{jj} - 2W_{ij} \ \forall_{i,j} \text{ s.t. } A_{ij} = 1\}$
4: Add constraints to $\mathcal{K} : \{D_{ij} > (1 - A_{ij}) \max_m (A_{im} D_{im}) - \xi \ \forall_{i,j}\}$
5: repeat
6:    Solve for the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and eigenvalues and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ of $\mathbf{K}$ using an SVD.
7:    $\mathbf{B} \leftarrow \beta \sum_{i=1}^{d} \mathbf{v}_i \mathbf{v}_i^\top - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^\top$
8:    $\mathbf{K} \leftarrow \hat{\mathbf{K}}$
9:    $\hat{\mathbf{K}} \leftarrow \text{argmax}_{\mathbf{K} \in \mathcal{K}} \operatorname{tr}(\mathbf{KB})$ {Found via SDP}
10: until $\|\mathbf{K} - \hat{\mathbf{K}}\| \leq \kappa$
11: Perform SVD on $\hat{\mathbf{K}}$ to compute the $d$ leading eigenvectors $\hat{\mathbf{v}}_i$ and corresponding eigenvalues $\hat{\lambda}_i$, and set $\mathbf{y}_i \leftarrow \sqrt{\hat{\lambda}_i} \hat{\mathbf{v}}_i$ for $i = 1, \ldots, d$
12: return $\mathbf{y}_1, \ldots, \mathbf{y}_d$
```

The steps of MVE+SP are shown in Algorithm 4. Similarly, it is possible to iterate MVE+SP, enforcing cutting plane constraints to preserve the graph structure corresponding to a maximum weight subgraph method. In the next Section we demonstrate that without explicitly preserving graph topology, many dimensionality reduction algorithms will alter the local topology of each point, despite preserving local distances, and thus produce less accurate embeddings.

4.1 MVE+SP Experiments

The tendency for manifold-learning algorithms to collapse parts of the manifold is evident with real-world data as well as synthetic examples (as shown earlier in Figure 4.1). Preserving a k-nearest neighbor graph during dimensionality reduction means a nearest-neighbor classifier is guaranteed to perform equally well using the low-dimensional embedding as with the original high-dimensional dataset. Without explicitly preserving graph topology, many unsupervised dimensionality reduction algorithms will drastically alter local neighborhoods of each point, despite preserving all local distances, and thus these methods will perform poorly when used for classification. Table 4.1 shows the improvement of MVE+SP over other methods in terms of classification accuracy using a 1-nearest neighbor classifier on a variety of 2D embeddings.

For each UCI dataset in Table 4.1, we sampled 120 points from the two largest classes, and embedded the data in 2D using KPCA, MVU, MVE, and MVE+SP. After embedding we measured the resulting performance of a 1NN classifier on the low-dimensional embedding. Cross-validation was used to find the optimal value of k for each algorithm (except KPCA) and dataset, where k specifies the number of neighbors used to build a graph with kNN from the input data. The data was split 60/20/20 into training/cross-validation/testing groups, and Table 4.1 shows the results of averaging 50 folds. The All-Dimension column on the right indicates the performance of a 1NN classifier on the original data using all features. MVE+SP performs better than all other low-dimensional methods, and despite using only 2 features per datapoint, yields higher accuracy than using all of the original features for the datasets Ionosphere, Ecoli, and OptDigits. In these cases,

	KPCA	MVU	MVE	MVE+SP	All-Dimensions
Ion.	0.660 ± 0.101	0.850 ± 0.062	0.812 ± 0.077	$\mathbf{0.871 \pm 0.065}$	0.788 ± 0.085
Cars	0.661 ± 0.075	0.701 ± 0.110	0.716 ± 0.082	$\mathbf{0.781 \pm 0.073}$	0.793 ± 0.083
Derm.	0.588 ± 0.089	0.636 ± 0.081	0.648 ± 0.094	$\mathbf{0.663 \pm 0.096}$	0.763 ± 0.080
Ecoli	0.949 ± 0.036	0.956 ± 0.040	0.948 ± 0.038	$\mathbf{0.960 \pm 0.039}$	0.956 ± 0.034
Wine	0.680 ± 0.088	0.685 ± 0.090	0.683 ± 0.090	$\mathbf{0.697 \pm 0.079}$	0.715 ± 0.079
4 vs. 9	0.944 ± 0.043	0.992 ± 0.016	0.996 ± 0.013	$\mathbf{0.998 \pm 0.010}$	0.986 ± 0.025

Table 4.1: Average classification accuracy and standard deviation of a 1-nearest neighbor classifier on UCI datasets: Ionosphere, Cars, Dermatology, Ecoli, Wine, and OptDigits 4 vs 9. MVE+SP has higher accuracy than other low-dimensional methods and also beats All-Dimensions on Ionosphere, Ecoli, and OptDigits. Other class pairs for OptDigits are not shown since all methods achieved near 100% accuracy.

MVE+SP finds coordinates which capture the key modes of variation in the data, disregarding noisy dimensions that reduce accuracy when using all input features. It is clear from this experiment that manifold-collapsing is an unfortunate consequence when reducing dimensionality, even when using algorithms which are not graph-based such as KPCA. Only by explicitly preserving graph topology during dimensionality reduction can we guarantee that we preserve an input graph exactly, not just its local properties.

MVE+SP is an unsupervised dimensionality reduction method and thus is not intended to compete with supervised techniques such as linear discriminant analysis (LDA) or large margin nearest neighbors (LMNN) [R. Fisher, 1936, Weinberger and Saul, 2009]. However, it is clear from the experiments above that using dimensionality reduction algorithms as a pre-processing step before subsequent classification tasks, a common setup, can reduce the performance of a classifier. MVE+SP, however, is capturing a more accurate low-dimensional embedding of the data and is maintaining and sometimes helping classification rates. Preserving local distances and global topology while aggressively unfolding a graph built from the data appears to address the deficiencies of other techniques.

Chapter 5

Fast Solvers for Large-Scale Graph Embedding

In this section, we present a low-rank approximation to the original SPE algorithm, implemented using a fast custom solver based on projected stochastic gradient descent, which allows SPE to scale to larger networks.

5.1 Algorithm

Given a network of n nodes represented as a graph with adjacency matrix $\mathbf{A} \in \mathbb{B}^{n \times n}$, SPE finds an embedding $\mathbf{L} \in \mathbb{R}^{d \times n}$ such that d is small and running a connectivity algorithm such as k-nearest neighbors on $\mathbf{L}$ returns $\mathbf{A}$. As first proposed, SPE learns a matrix $\mathbf{K}$ via a semidefinite program (SDP) and then decomposes $\mathbf{K} = \mathbf{L}^\top \mathbf{L}$ by performing singular value decomposition. In contrast, we propose optimizing $\mathbf{L}$ directly. Although for $d < N$, this problem is now non-convex, because of the stochastic nature of the optimizer we have found the algorithm does not suffer from local minima in practice.

SPE for greedy nearest-neighbor constraints solves the following SDP:

$$\begin{aligned}&\max_{\mathbf{K} \in \mathcal{K}} \operatorname{tr}(\mathbf{K}\mathbf{A})\\ &D_{ij} > (1 - A_{ij}) \max_m (A_{im} D_{im}) \ \forall_{i,j}\end{aligned}$$

where $D_{ij} = K_{ii} + K_{jj} - 2K_{ij}$ and $\mathcal{K} = \{\mathbf{K} \succeq 0,\ \operatorname{tr}(\mathbf{K}) \leq 1, \sum_{ij} K_{ij} = 0\}$. The constraints require the embedding of each node to be more distant from its non-neighbors than its neighbors. Let $S = \{\mathbf{E}_1, \mathbf{E}_2, ...\mathbf{E}_m\}$ be the set of all triplet constraints, where each $\mathbf{E}_l$ is a constraint matrix corresponding to a triplet (i, j, k) such that $A_{ij} = 1$ and $A_{ik} = 0$. This set of all triplets clearly subsumes the distance constraints above, and allows each individual constraint to be written as $\text{tr}(\mathbf{E}_l\mathbf{K}) > 0$ where $\text{tr}(\mathbf{E}_l\mathbf{K}) = K_{jj} - 2K_{ij} + 2K_{ik} - K_{kk}$. Temporarily dropping the centering and scaling constraints, we can now formulate the SDP above as maximizing the following objective function over $\mathbf{L}$:

$$f(\mathbf{L}) = \rho\text{tr}(\mathbf{L}^\top\mathbf{L}\mathbf{A}) - \sum_{l\in S}\max(\text{tr}(\mathbf{E}_l\mathbf{L}^\top\mathbf{L}), 0).$$

Note that we have introduced a Lagrange multiplier ρ as an additional parameter which trades off between the loss term and regularization term. We will maximize $f(\mathbf{L})$ via projected stochastic subgradient decent. Define the subgradient in terms of a single randomly chosen triplet:

$$\nabla(f(\mathbf{L}), \mathbf{E}_l) = \begin{cases} 2\mathbf{L}(\rho\mathbf{A} - \mathbf{E}_l) \text{ if } \ \text{tr}(\mathbf{E}_l\mathbf{L}^\top\mathbf{L}) > 0 \\ 0 \ \ \text{otherwise} \end{cases}$$

and for each randomly chosen triplet constraint E_l, if $\text{tr}(\mathbf{E}_l\mathbf{L}^\top\mathbf{L}) > 0$ then update $\mathbf{L}$ according to:

$$\mathbf{L}_{t+1} = \mathbf{L}_t + \eta\nabla(f(\mathbf{L}_t), \mathbf{E}_l)$$

where the step-size $\eta = \frac{1}{\sqrt{t}}$ [Nedic and Bertsekas, 2001]. After each step, we can use projection to enforce that $\text{tr}(\mathbf{L}^\top\mathbf{L}) \leq 1$ and $\sum_{ij}(\mathbf{L}^\top\mathbf{L})_{ij} = 0$, by subtracting the mean from $\mathbf{L}$ and dividing each entry of $\mathbf{L}$ by its Frobenius norm. $\mathbf{L}$ is initialized either randomly or from the solution of spectral embedding or Laplacian eigenmaps [Belkin and Niyogi, 2002]. Algorithm 5 shows the steps of Large-scale Structure Preserving Embedding.

In practice, instead of optimizing over a single randomly chosen triplet at each iteration, we find it useful to randomly select a node at each iteration, and use the gradient computed from all *impostor* triplets, since it is only for these triplets that a gradient step is taken. As shown in Figure 5.1 an impostor is a node which violates the neighborhood of another node. For each impostor triplet $\{i, j, k\}$, i is the randomly chosen target node, j is the furthest connected neighbor of i and k is a node unconnected to i but currently closer than j.

Algorithm 5 Large-Scale Structure Preserving Embedding

```
Input: $\mathbf{A} \in \mathbb{B}^{n \times n}$, dimensionality $d$, regularizer parameter $\rho$, and maximum iterations $T$
 1: Initialize $\mathbf{L}_0 \leftarrow \text{rand}(d, n)$
    (or optionally initialize to spectral embedding or Laplacian eigenmaps solution)
 2: $t \leftarrow 0$
 3: repeat
 4:   $\eta_t \leftarrow \frac{1}{\sqrt{t+1}}$
 5:   $i \leftarrow \text{rand}(1 \ldots n)$
 6:   $j = \arg\min_j \| L_i - L_j \|_2 \;\; \forall_j$ s.t. $A(i, j) = 1$
 7:   $\mathbf{E} \leftarrow \text{zeros}(n \times n)$, $\mathbf{E}_{jj} \leftarrow 1$, $\mathbf{E}_{ij} \leftarrow -1$, $\mathbf{E}_{ji} \leftarrow -1$
 8:   for all $k$ s.t. $\| L_i - L_k \|_2 < \| L_i - L_j \|_2$ AND $A(i, k) = 0$ do
 9:     $\mathbf{E}_{ik} \leftarrow 1$, $\mathbf{E}_{ki} \leftarrow 1$, $\mathbf{E}_{kk} \leftarrow -1$
10:   end for
11:   $\nabla_t \leftarrow 2\mathbf{L}_t (\rho\mathbf{A} - \mathbf{E})$
12:   $\mathbf{L}_{t+1} \leftarrow \mathbf{L}_t + \eta_t \nabla_t$
13:   {Subtract out mean}
14:   $\mathbf{L}_{t+1} = \frac{\mathbf{L}_{t+1}}{\|\mathbf{L}_{t+1}\|_2}$ {Project on to unit sphere}
15:   $t \leftarrow t + 1$
16: until $t \geq T$
17: return $\mathbf{L}$
```

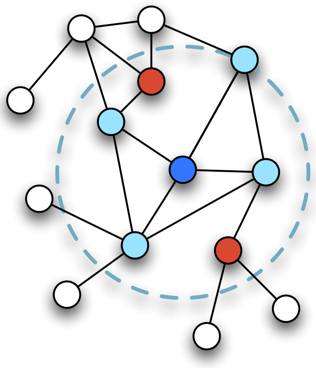
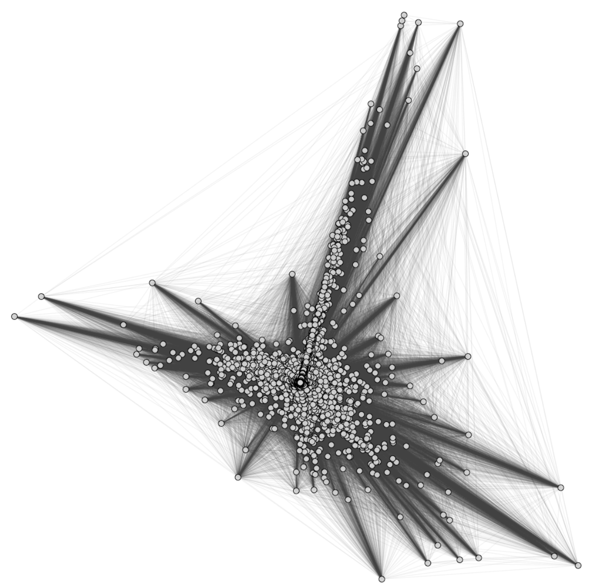
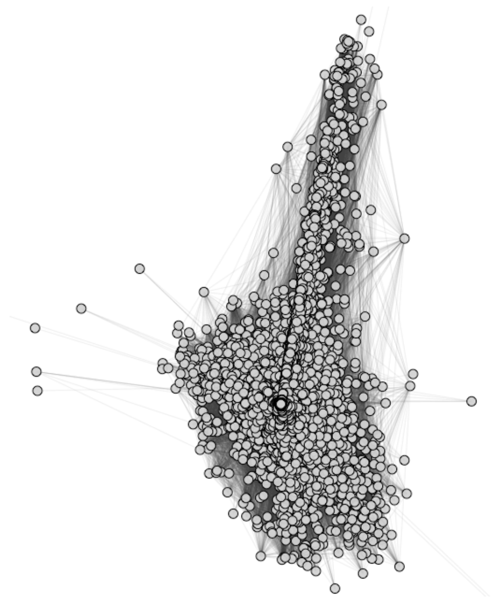


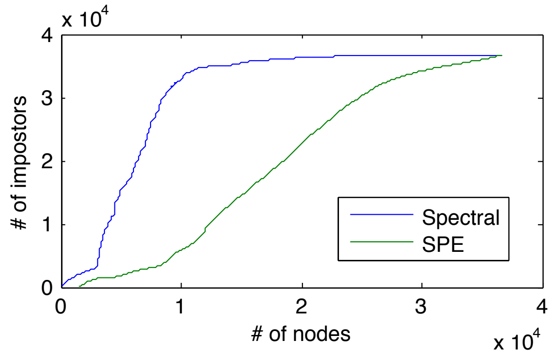
Figure 5.1: The red nodes are identified as impostors to the neighborhood of the center node (dark blue), because the impostors (red) are closer than the furthest of the connected nodes (light blue).

5.2 Experiments

In Figure 5.2 we see two embeddings of the Enron email network [Leskovec et al., 2005]. Each of the 36692 nodes in the network represents a person, and there exist edges between each pair of people who have communicated via email. Because of the high degree of many of the nodes in the network, it is likely impossible to find a 2D embedding which preserves topology exactly – meaning all nodes have zero impostors. The network may require a higher dimensional embedding. However, we see that the 2D visualization produced by SPE has far fewer impostors than that produced by spectral embedding, and thus provides a more accurate visualization.


(a) Spectral embedding


(b) SPE-SGD



(c) Impostors

Figure 5.2: The Enron email network embedded into 2D by spectral embedding (a), and SPE-SGD (b). The plot on the right shows how many nodes have fewer than x impostors. We see that embedding this network into 2D yields many impostors; however, on average, nodes in the SPE embedding have many fewer impostors than nodes in the spectral embedding.

Chapter 6

Learning a Metric from a Network

Many real-world networks are described by both connectivity information and features for every node. In this chapter, we present *structure preserving metric learning* (SPML), an algorithm for learning a Mahalanobis distance metric from a network features such that the learned distances are tied to the inherent connectivity structure of the network. Like *structure preserving embedding*, SPML learns a metric which is structure preserving, meaning a connectivity algorithm such as k-nearest neighbors will yield the correct connectivity when applied using the distances from the learned metric. We show a variety of synthetic and real-world experiments where SPML predicts link patterns from node features more accurately than standard techniques. We further demonstrate a method for optimizing SPML based on stochastic gradient descent which removes the running-time dependency on the size of the network and allows the method to easily scale to networks of thousands of nodes and millions of edges.

6.1 Introduction

The proliferation of social networks on the web has spurred many significant advances in modeling networks Airoldi et al. [2008], Chang and Blei [2010], Leskovec and Horvitz [2008], Leskovec et al. [2005], Namata et al. [2010], Newman [2003], Yang et al. [2009]. However, while many efforts have been focused on modeling networks as weighted or unweighted graphs Newman [2004], or constructing features from links to describe the nodes in a network Middendorf et al. [2004], Xu and Li [2006], few techniques have focused on real-world network data which consists of both node features in addition to connectivity information. Many social networks are of this form; on services such as Facebook, Twitter, or LinkedIn, there are profiles which describe each person, as well as the connections they make. The relationship between a node's features and connections is often not explicit. For example, people "friend" each other on Facebook for a variety of reasons: perhaps they share similar parts of their profile such as their school or major, or perhaps they have completely different profiles. We want to learn the relationship between profiles and links from massive social networks such that we can better predict who is likely to connect. To model this relationship, one could simply model each link independently, where one simply learns what characteristics of two profiles imply a possible link. However, this approach completely ignores the structural characteristics of the links in the network. We posit that modeling independent links is insufficient, and in order to better model these networks one must account for the inherent topology of the network as well as the interactions between the features of nodes. We thus propose *structure preserving metric learning* (SPML), a method for learning a distance metric between nodes that preserves the structural network behavior seen in data.

6.1.1 Background

Metric learning algorithms have been successfully applied to many supervised learning tasks such as classification [Chechik et al., 2010, Weinberger and Saul, 2009]. These methods first build a k-nearest neighbors (kNN) graph from training data with a fixed k, and then learn a Mahalanobis distance metric which tries to keep connected points with similar labels close while pushing away class impostors, pairs of points which are connected but of different classes. Fundamentally, these supervised methods aim to learn a distance metric such that applying a connectivity algorithm (for instance, k-nearest neighbors) under the metric will produce a graph where no point is connected to others with different class labels. In practice, these constraints are enforced with slack. Once the metric is learned, the class label for an unseen datapoint can be predicted by the majority vote of nearby points under the learned metric.

Unfortunately, these metric learning algorithms are not easily applied when we are given a network as input instead of class labels for each point. Under this new regime, we want to learn a metric such that points connected in the network are close and points which are unconnected are more distant. Intuitively, certain features or groups of features should influence how nodes connect, and thus it should be possible to learn a mapping from features to connectivity such that the mapping respects the underlying topological structure of the network. Like previous metric learning methods, SPML learns a metric which reconciles the input features with some auxiliary information such as class labels. In this case, instead of pushing away class impostors, SPML pushes away *graph impostors*, points which are close in terms of distance but which should remain unconnected in order to preserve the topology of the network. Thus SPML learns a metric where the learned distances are inherently tied to the original input connectivity.

Preserving graph topology is possible by enforcing simple linear constraints on distances between nodes Shaw and Jebara [2009]. By adapting the constraints from *structure preserving embedding*, we formulate simple linear structure preserving constraints for metric learning that enforce that neighbors of each node are closer than all others. Furthermore, we adapt these constraints for an online setting similar to PEGASOS Shalev-Shwartz et al. [To appear] and OASIS Chechik et al. [2010], such that we can apply SPML to large networks by optimizing with stochastic gradient descent (SGD) [Robbins and Monro, 1951, Bottou, 2003].

6.2 Structure preserving metric learning

Given as input an adjacency matrix $\mathbf{A} \in \mathbb{B}^{n \times n}$, and node features $\mathbf{X} \in \mathbb{R}^{d \times n}$, *structure preserving metric learning* (SPML) learns a Mahalanobis distance metric parameterized by a positive semidefinite (PSD) matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, where $\mathbf{M} \succeq 0$. The distance between two points under the metric is defined as

$$D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j).$$

When the metric is the identity $\mathbf{M} = I_d$, $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$ represents the squared Euclidean distance between the i'th and j'th points. Learning $\mathbf{M}$ is equivalent to learning a linear scaling on the input features $\mathbf{LX}$ where $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$ and $\mathbf{L} \in \mathbb{R}^{d\times d}$. SPML learns an $\mathbf{M}$ which is *structure preserving*, as defined in Definition 10. Given a *connectivity algorithm* $\mathcal{G}$, SPML learns a metric such that applying $\mathcal{G}$ to the input data using the learned metric produces the input adjacency matrix exactly.[1] Possible choices for $\mathcal{G}$ include maximum weight b-matching, k-nearest neighbors, ϵ-neighborhoods, or maximum weight spanning tree.

Definition 10 *Given a graph with adjacency matrix* $\mathbf{A}$*, a distance metric parametrized by* $\mathbf{M} \in \mathbb{R}^{d\times d}$ *is* ***structure preserving*** *with respect to a connectivity algorithm* $\mathcal{G}$*, if* $\mathcal{G}(\mathbf{X}, \mathbf{M}) = \mathbf{A}$.

6.2.1 Preserving graph topology with linear constraints

To preserve graph topology, we use the same linear constraints as *structure preserving embedding* (SPE) Shaw and Jebara [2009], but apply them to $\mathbf{M}$, which parameterizes the distances between points. A useful tool for defining distances as linear constraints on $\mathbf{M}$ is the transformation

$$D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top\mathbf{M}\mathbf{x}_i + \mathbf{x}_j^\top\mathbf{M}\mathbf{x}_j - \mathbf{x}_i^\top\mathbf{M}\mathbf{x}_j - \mathbf{x}_j^\top\mathbf{M}\mathbf{x}_i, \tag{6.1}$$

which allows linear constraints on the distances to be written as linear constraints on the $\mathbf{M}$ matrix. For different connectivity schemes below, we present linear constraints which enforce graph structure to be preserved.

Nearest neighbor graphs The *k-nearest neighbor algorithm* (k-nn) connects each node to the k neighbors to which the node has shortest distance, where k is an input parameter; therefore, setting k to the true degree for each node, the distances to all disconnected nodes must be larger than the distance to the farthest connected neighbor:

$$D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) > (1 - A_{ij}) \max_l (A_{il} D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)), \forall i, j.$$

[1]In the remainder of the paper, we interchangeably use $\mathcal{G}$ to denote the set of feasible graphs and the algorithm used to find the optimal connectivity within the set of feasible graphs.

Similarly, preserving an ϵ-neighborhood graph obeys linear constraints on $\mathbf{M}$: $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon$, $\forall\{i, j | A_{ij} = 1\}$, and $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq \epsilon$, $\forall\{i, j | A_{ij} = 0\}$. If for each node the connected distances are less than the unconnected distances (or some ϵ), i.e., the metric obeys the above linear constraints, Definition 10 is satisfied, and thus the connectivity computed under the learned metric $\mathbf{M}$ is exactly $\mathbf{A}$.

Maximum weight subgraphs Unlike nearest neighbor algorithms, which select edges greedily for each node, maximum weight subgraph algorithms select edges from a weighted graph to produce a subgraph which has total maximal weight [Fremuth-Paeger and Jungnickel, 1999]. Given a metric parametrized by $\mathbf{M}$, let the weight between two points (i, j) be the negated pairwise distance between them: $Z_{ij} = -D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = -(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$. For example, *maximum weight b-matching* finds the maximum weight subgraph while also enforcing that every node has a fixed degree b_i for each i'th node. The formulation for *maximum weight spanning tree* is similar. Unfortunately, like for SPE, preserving structure for these algorithms requires enforcing many linear constraints of the form: $\mathrm{tr}(\mathbf{Z}^\top \mathbf{A}) \geq \mathrm{tr}(\mathbf{Z}^\top \tilde{\mathbf{A}}), \forall \tilde{\mathbf{A}} \in \mathcal{G}$. This reveals one critical difference between structure preserving constraints of these algorithms and those of nearest-neighbor graphs: there are exponentially many linear constraints. To avoid an exponential enumeration, the most violated inequalities can be introduced sequentially using a cutting-plane approach as shown in the next section.

6.2.2 Algorithm derivation

By combining the linear constraints from the previous section with a Frobenius norm (denoted $||\cdot||_{\mathrm{F}}$) regularizer on $\mathbf{M}$ and regularization parameter ρ, we have a simple semidefinite program (SDP) which learns an $\mathbf{M}$ that is structure preserving and has minimal complexity. Algorithm 6 summarizes the naive implementation of SPML when the connectivity algorithm is k-nearest neighbors, which is optimized by a standard SDP solver. For maximum weight subgraph connectivity (e.g., b-matching), we use a *cutting-plane* method Joachims et al. [2009], iteratively finding the worst violating constraint and adding it to a working-set. We can find the most violated constraint at each iteration by computing the adjacency matrix $\tilde{\mathbf{A}}$ that maximizes $\text{tr}(\tilde{\mathbf{Z}}\tilde{\mathbf{A}})$ s.t. $\tilde{\mathbf{A}} \in \mathcal{G}$, which can be done using various methods Fremuth-Paeger and Jungnickel [1999], Huang and Jebara [2007, 2011]. Each added constraint enforces that the total weight along the edges of the true graph is greater than total weight of any other graph by some margin. Algorithm 7 shows the steps for SPML with cutting-plane constraints.

Algorithm 6 Structure preserving metric learning with nearest neighbor constraints

```
Input: $\mathbf{A} \in \mathbb{B}^{n\times n}$, $\mathbf{X} \in \mathbb{R}^{d\times n}$, and parameter $\rho$
 1: $\mathcal{K} = \{\mathbf{M} \succeq 0, D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq (1 - A_{ij}) \max_l (A_{il} D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_l)) + 1 - \xi \;\; \forall_{i,j}\}$
 2: $\tilde{\mathbf{M}} \leftarrow \text{argmin}_{\mathbf{M}\in\mathcal{K}} \frac{\rho}{2}||\mathbf{M}||_{\text{F}}^2 + \xi$ {Found via SDP}
 3: return $\tilde{\mathbf{M}}$
```

Algorithm 7 Structure preserving metric learning with cutting-plane constraints

```
Input: $\mathbf{A} \in \mathbb{B}^{n\times n}$, $\mathbf{X} \in \mathbb{R}^{d\times n}$, connectivity algorithm $\mathcal{G}$, and parameters $\rho, \kappa$
 1: $\mathcal{K} = \{\mathbf{M} \succeq 0\}$
 2: repeat
 3:   $\tilde{\mathbf{M}} \leftarrow \text{argmin}_{\mathbf{M}\in\mathcal{K}} \frac{\rho}{2}||\mathbf{M}||_{\text{F}}^2 + \xi$ {Found via SDP}
 4:   $\tilde{\mathbf{Z}} \leftarrow 2\mathbf{X}^\top\tilde{\mathbf{M}}\mathbf{X} - \text{diag}(\mathbf{X}^\top\tilde{\mathbf{M}}\mathbf{X})\mathbf{1}^\top - \mathbf{1}\text{diag}(\mathbf{X}^\top\tilde{\mathbf{M}}\mathbf{X})^\top$
 5:   $\tilde{\mathbf{A}} \leftarrow \text{argmax}_{\tilde{\mathbf{A}}} \text{tr}(\tilde{\mathbf{Z}}^\top\tilde{\mathbf{A}})$ s.t. $\tilde{\mathbf{A}} \in \mathcal{G}$ {Find worst violator}
 6:   if $|\text{tr}(\tilde{\mathbf{Z}}^\top\tilde{\mathbf{A}}) - \text{tr}(\tilde{\mathbf{Z}}^\top\mathbf{A})| \geq \kappa$ then
 7:     add constraint to $\mathcal{K}$ : $\text{tr}(\mathbf{Z}^\top\mathbf{A}) - \text{tr}(\mathbf{Z}^\top\tilde{\mathbf{A}}) > 1 - \xi$
 8:   end if
 9: until $|\text{tr}(\tilde{\mathbf{Z}}^\top\tilde{\mathbf{A}}) - \text{tr}(\tilde{\mathbf{Z}}^\top\mathbf{A})| \leq \kappa$
10: return $\tilde{\mathbf{M}}$
```

Unfortunately, for networks larger than a few hundred nodes or for high-dimensional features, these SDPs do not scale adequately. The complexity of the SDP scales with the number of variables and constraints, yielding a worst-case time of $O(d^3 + \mathcal{C}^3)$ where $\mathcal{C} = O(n^2)$. By temporarily omitting the PSD requirement on $\mathbf{M}$, Algorithm 7 becomes equivalent to a one-class *structural support vector machine* (structural SVM). Stochastic SVM algorithms have been recently developed that have convergence time with no dependence on input size Shalev-Shwartz et al. [2007]. Therefore, we develop a large-scale algorithm based on projected stochastic subgradient descent. The proposed adaptation removes the dependence on n, where each iteration of the algorithm is $O(d^2)$, sampling one random constraint at a time. We can rewrite the optimization as unconstrained over an objective function with a hinge-loss on the structure preserving constraints:

$$f(\mathbf{M}) = \frac{\rho}{2}||\mathbf{M}||_{\mathrm{F}}^2 - \frac{1}{|S|}\sum_{(i,j,k)\in S} \max(D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) + 1, 0).$$

Here the constraints have been written in terms of hinge-losses over triplets, each consisting of a node, its neighbor and its non-neighbor. The set of all such triplets is $S = \{(i,j,k)|A_{ij} = 1, A_{ik} = 0\}$. Using the distance transformation in Equation 6.1, each of the $|S|$ constraints can be written using a sparse matrix $\mathbf{E}^{(i,j,k)}$, where

$$E_{jj}^{(i,j,k)} = 1,\ E_{ik}^{(i,j,k)} = 1,\ ,E_{ki}^{(i,j,k)} = 1,\ ,E_{ij}^{(i,j,k)} = -1,\ E_{ji}^{(i,j,k)} = -1,\ ,E_{kk}^{(i,j,k)} = -1,$$

and whose other entries are zero. By construction, sparse matrix multiplication of $\mathbf{E}^{(i,j,k)}$ indexes the proper elements related to nodes i, j, and k, such that $\mathrm{tr}(\mathbf{E}^{(i,j,k)}\mathbf{X}^\top\mathbf{M}\mathbf{X})$ is equal to $D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k)$. The subgradient of f at $\mathbf{M}$ is then

$$\nabla f = \rho\mathbf{M} + \frac{1}{|S|}\sum_{(i,j,k)\in S_+} \mathbf{X}\mathbf{E}^{(i,j,k)}\mathbf{X}^\top,$$

where $S_+ = \{(i,j,k)|D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) + 1 > 0\}$. If for all triplets this quantity is negative, there exists no unconnected neighbor of a point which is closer than a point's farthest connected neighbor – precisely the structure preserving criterion for nearest neighbor algorithms. In practice, we optimize this objective function via stochastic subgradient descent. We sample a batch of triplets, replacing S in the objective function with a random subset of S of size B. If a true metric is necessary, we intermittently project $\mathbf{M}$ onto the PSD cone. We use a learning rate of $\eta_t \leftarrow \frac{1}{\rho t}$ similar to stochastic SVMs [Shalev-Shwartz et al., 2007] . Full details about constructing the constraint matrices and minimizing the objective are shown in Algorithm 8.

6.2.3 Analysis

In this section, we provide analysis for the scaling behavior of SPML using SGD. A primary insight is that, since Algorithm 8 regularizes with the L_2 norm and penalizes with hinge-loss,

Algorithm 8 Structure preserving metric learning with nearest neighbor constraints and optimization with projected stochastic subgradient descent

```
Input: $\mathbf{A} \in \mathbb{B}^{n \times n}$, $\mathbf{X} \in \mathbb{R}^{d \times n}$, and parameters $\rho, T, B$
 1: $\mathbf{M}_1 \leftarrow \mathbf{I}_d$
 2: for $t$ from 1 to $T - 1$ do
 3:     $\eta_t \leftarrow \frac{1}{\rho t}$
 4:     $\mathbf{E} \leftarrow \mathbf{0}_{n,n}$
 5:     for $b$ from 1 to $B$ do
 6:         $(i, j, k) \leftarrow$ Sample random triplet from $S = \{(i, j, k) \mid A_{ij} = 1, A_{ik} = 0\}$
 7:         if $D_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_j) - D_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_k) + 1 > 0$ then
 8:             $\mathbf{E}_{jj} \leftarrow \mathbf{E}_{jj} + 1$, $\mathbf{E}_{ik} \leftarrow \mathbf{E}_{ik} + 1$, $\mathbf{E}_{ki} \leftarrow \mathbf{E}_{ki} + 1$
 9:             $\mathbf{E}_{ij} \leftarrow \mathbf{E}_{ij} - 1$, $\mathbf{E}_{ji} \leftarrow \mathbf{E}_{ji} - 1$, $\mathbf{E}_{kk} \leftarrow \mathbf{E}_{kk} - 1$
10:         end if
11:     end for
12:     $\nabla_t \leftarrow \mathbf{X}\mathbf{E}\mathbf{X}^\top + \rho \mathbf{M}_t$
13:     $\mathbf{M}_{t+1} \leftarrow \mathbf{M}_t - \eta_t \nabla_t$
14:     Optional: $\mathbf{M}_{t+1} \leftarrow [\mathbf{M}_{t+1}]^+$ {Project onto the PSD cone}
15: end for
16: return $\mathbf{M}_T$
```

omitting the positive semidefinite requirement for $\mathbf{M}$ and vectorizing $\mathbf{M}$ makes the algorithm equivalent to a one-class, linear support vector machine with $O(n^3)$ input vectors. Thus, the stochastic optimization is an instance of the PEGAGOS algorithm Shalev-Shwartz et al. [2007], albeit a cleverly constructed one. The running time of PEGASOS does not depend on the input size, and instead only scales with the dimensionality, the desired optimization error on the objective function ϵ and the regularization parameter ρ. The optimization error ϵ is defined as the difference between the found objective value and the true optimal objective value, $f(\tilde{\mathbf{M}}) - \min_{\mathbf{M}} f(\mathbf{M})$.

Theorem 11 *Assume that the data is bounded such that* $\max_{(i,j,k)\in S} ||\mathbf{X}\mathbf{E}^{(i,j,k)}\mathbf{X}^\top||_{\mathrm{F}}^2 \leq R$, *and* $R \geq 1$. *During Algorithm 8 at iteration* T, *with* $\rho \leq 1/4$, *and batch-size* $B = 1$, *let* $\bar{\mathbf{M}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{M}_t$ *be the average* $\mathbf{M}$ *so far. Then, with probability of at least* $1 - \delta$,

$$f(\bar{\mathbf{M}}) - \min_{\mathbf{M}} f(\mathbf{M}) \leq \frac{84R^2 \ln(T/\delta)}{\rho T}.$$

Consequently, the number of iterations necessary to reach an optimization error of ϵ *is* $\tilde{O}(\frac{1}{\rho\epsilon})$.

Proof The theorem is proven by realizing that Algorithm 8 is an instance of PEGASOS without a projection step on one-class data, since Corollary 2 in Shalev-Shwartz et al. [To appear] proves this same bound for traditional SVM input, also without a projection step. The input to the SVM is the set of all $d \times d$ matrices $XE^{(i,j,k)}X^\top$ for each triplet $(i,j,k) \in S$. ■

Note that the large size of set S plays no role in the running time; each iteration requires $O(d^2)$ work. Assuming the node feature vectors are of bounded norm, the radius of the input data R is constant with respect to n, since each is constructed using the feature vectors of three nodes. In practice, as in the PEGASOS algorithm, we propose using $\mathbf{M}_T$ as the output instead of the average, as doing so performs better on real data, but an averaging version is easily implemented by storing a running sum of $\mathbf{M}$ matrices and dividing by T before returning.

Figure 6.2(b) shows the training and testing prediction performance on one of the experiments described in detail in Section 6.3 as stochastic SPML converges. The area under the receiver operator characteristic (ROC) curve is measured, which is related to the structure preserving hinge loss, and the plot clearly shows fast convergence and quickly diminishing returns at higher iteration counts.

6.2.4 Variations

While stochastic SPML does not scale with the size of the input graph, evaluating distances using a full $\mathbf{M}$ matrix requires $O(d^2)$ work. Thus, for high-dimensional data, one approach is to use *principal component analysis* or *random projections* to first reduce dimensionality. It has been shown that n points can be mapped into a space of dimensionality $O(\log n/\varepsilon^2)$ such that distances are distorted by no more than a factor of $(1 \pm \varepsilon)$ Dasgupta and Gupta [2003], Johnson and Lindenstrauss [1984]. Another approach is to to limit $\mathbf{M}$ to be nonzero only along the diagonal. Diagonalizing $\mathbf{M}$ reduces the amount of work to $O(d)$.

If modeling cross-feature interactions is necessary, another option for reducing the computational cost is to perform SPML using a low-rank factorization of $\mathbf{M}$. In this case, all references to $\mathbf{M}$ can be replaced with $\mathbf{L}^\top\mathbf{L}$, thus inducing a true metric without projection. The updated gradient with respect to $\mathbf{L}$ is simply

$$\nabla_t \leftarrow 2\mathbf{X}\mathbf{E}\mathbf{X}^\top\mathbf{L}^\top + \rho\mathbf{L}_t.$$

Using a factorization also allows replacing the regularizer with the Frobenius norm of the $\mathbf{L}$ matrix, which is equivalent to the *nuclear norm* of $\mathbf{M}$ [Rennie and Srebro, 2005]. Unfortunately, using this formulation causes the objective to no longer be convex, but seems to work well in practice. Finally, when predicting links of new nodes, SPML does not know how many connections to predict. To address this uncertainty, we propose a variant to SPML called *degree distributional metric learning* (DDML), which simultaneously learns the metric as well as parameters for the connectivity algorithm. Details on DDML and low-rank SPML are provided in Sections 6.4 and 6.5 respectively.

6.3 Experiments

We present a variety of synthetic and real-world experiments that elucidate the behavior of SPML. First we show how SPML performs on a simple synthetic dataset that is easily visualized in two dimensions and which we believe mimics many traditional network datasets. We then demonstrate favorable performance for SPML in predicting links of the Wikipedia document network and the Facebook social network.

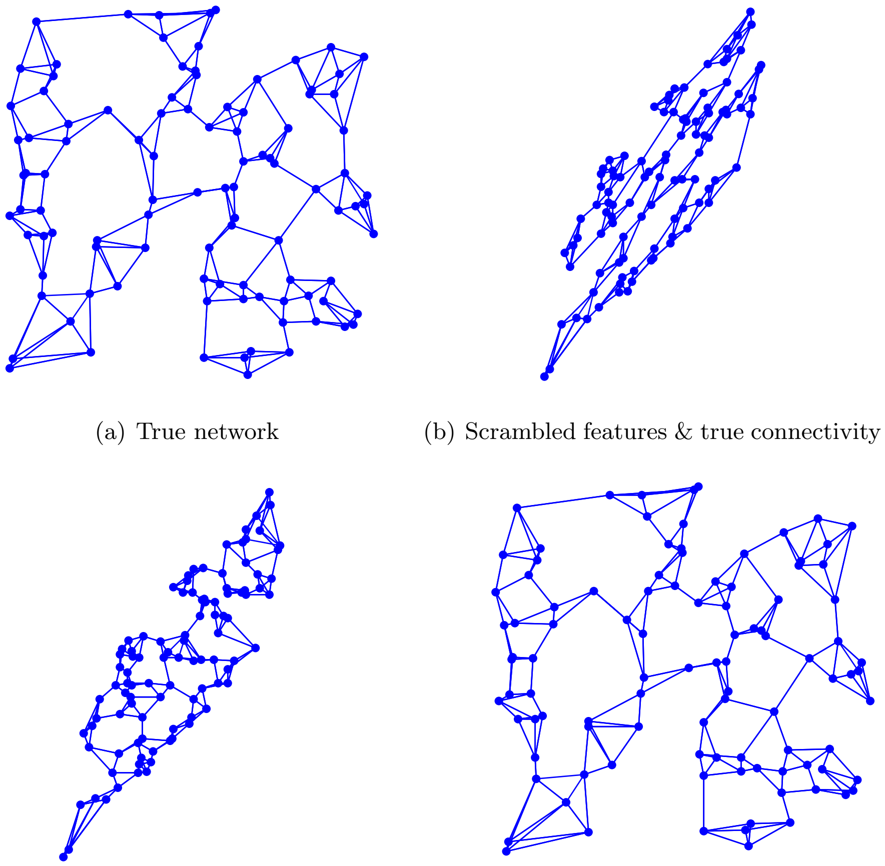
6.3.1 Synthetic example

To better understand the behavior of SPML, consider the following synthetic experiment. First n points are sampled from a d-dimensional uniform distribution. These vectors represent the true features for the n nodes $\mathbf{X} \in \mathbb{R}^{d \times n}$. We then compute an adjacency matrix by performing a minimum-distance b-matching on $\mathbf{X}$. Next, the true features are scrambled by applying a random linear transformation: $\mathbf{RX}$ where $\mathbf{R} \in \mathbb{R}^{d \times d}$. Given $\mathbf{RX}$ and $\mathbf{A}$, the goal of SPML is to learn a metric $\mathbf{M}$ that undoes the linear scrambling, so that when b-matching is applied to $\mathbf{RX}$ using the learned distance metric, it produces the input adjacency matrix.

Figure 6.1 illustrates the results of the above experiment for $d = 2$, $n = 50$, and $b = 4$. In Figure 6.1(a), we see an embedding of the graph using the true features for each node as coordinates, and connectivity generated from b-matching. In Figure 6.1(b), the random linear transformation has been applied. We posit that many real-world datasets resemble plot 6.1(b), with seemingly incongruous feature and connectivity information. If we apply b-matching to the scrambled data, it produces connections shown in Figure 6.1(c). Finally, by learning $\mathbf{M}$ via SPML (Algorithm 7) and computing $\mathbf{L}$ by Cholesky decomposition of $\mathbf{M}$, we can recover features $\mathbf{LRX}$ (Figure 6.1(d)) that respect the structure in the target adjacency matrix and thus more closely resemble the true features used to generate the data.

6.3.2 Link prediction

We compare SPML to a variety of methods for predicting links from node features: Euclidean distances, *relational topic models* (RTM) , and traditional *support vector machines* (SVM). A simple baseline for comparison is how well the Euclidean distance metric performs at ranking possible connections. Relational topic models learn a link probability function in addition to latent topic mixtures describing each node Chang and Blei [2010]. For the SVM, we construct training examples consisting of the pairwise differences between



(a) True network

(b) Scrambled features & true connectivity

(c) Scrambled features & implied connectivity

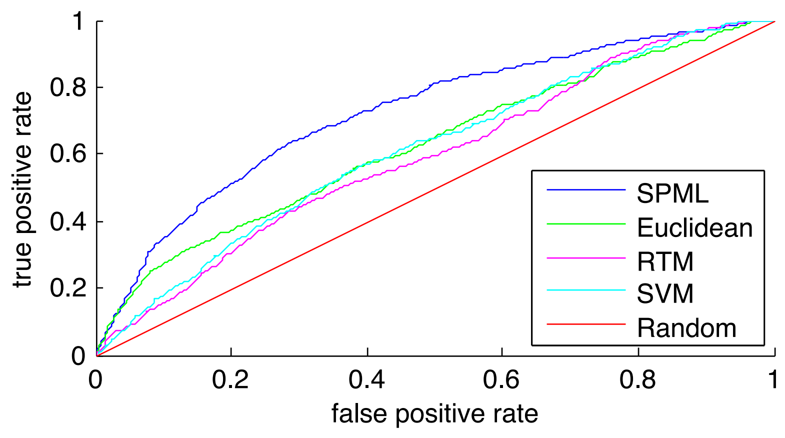
(d) Recovered features & true connectivity

Figure 6.1: In this synthetic experiment, SPML finds a metric that inverts the random transformation applied to the features (b), such that under the learned metric (d) the implied connectivity is identical to the original connectivity (a) as opposed to inducing a different connectivity (c).

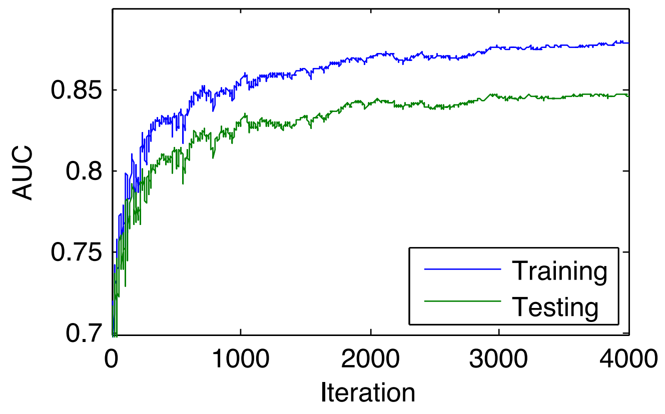
node features. Training examples are labeled positive if there exists an edge between the corresponding pair of nodes, and negative if there is no edge. Because there are potentially $O(n^2)$ possible examples, and the graphs are sparse, we subsample the negative examples so that we include a randomly chosen equal number of negative examples as positive edges. Without subsampling, the SVM is unable to run our experiments in a reasonable time. We use the SVMPerf implementation for our SVM Joachims [2006], and the authors' code for RTM Chang and Blei [2010].

Interestingly, an SVM with these inputs can be interpreted as an instance of SPML using diagonal $\mathbf{M}$ and the ϵ-neighborhood connectivity algorithm, which connects points based on their distance, completely independently of the rest of the graph structure. We thus expect to see better performance using SPML in cases where the structure is important. The RTM approach is appropriate for data that consists of counts, and is a generative model which recovers a set of topics in addition to link predictions. Despite the generality of the model, RTM does not seem to perform as well as discriminative methods in our experiments, especially in the Facebook experiment where the data is quite different from bag-of-words features. For SPML, we run the stochastic algorithm with batch size 10. We skip the PSD projection step, since these experiments are only concerned with prediction, and obtaining a true metric is not necessary. SPML is implemented in MATLAB and requires only a few minutes to converge for each of the experiments below.

Wikipedia articles We apply SPML to predicting links on Wikipedia pages. Imagine the scenario where an author writes a new Wikipedia entry and then, by analyzing the word counts on the newly written page, an algorithm is able to suggest which other Wikipedia pages it should link to. We first create a few subnetworks consisting of all the pages in a given category, their bag-of-words features, and their connections. We choose three categories: "graph theory topics", "philosophy concepts", and "search engines". We use a word dictionary of common words with stop-words removed. For each network, we split the data 80/20 for training and testing, where 20% of the nodes are held out for evaluation. On the remaining 80% we cross-validate (five folds) over the parameters for each algorithm (RTM, SVM, SPML), and train a model using the best-scoring regularization parameter.



(a) Average ROC curve for Wikipedia Experiment: "graph theory topics"



(b) Convergence behavior of SPML optimized via SGD on Facebook Data

Figure 6.2: Average ROC performance for the "graph theory topics" Wikipedia experiment (top) shows a strong lift for SPML over competing methods. We see that SPML converges quickly with diminishing returns after many iterations (bottom).

For SPML, we use the diagonal variant of Algorithm 8, since the high-dimensionality of the input features reduces the benefit of cross-feature weights. On the held-out nodes, we task each algorithm to rank the unknown edges according to distance (or another measure of link likelihood), and compare the accuracy of the rankings using *receiver operator characteristic* (ROC) curves. Table 6.1 lists the statistics of each category and the average area under the curve (AUC) over three train/test splits for each algorithm. A ROC curve for the "graph theory" category is shown in Figure 6.2(a). For "graph theory" and "search engines", SPML provides a distinct advantage over other methods, while no method has a particular advantage on "philosophy concepts". One possible explanation for why the SVM is unable to gain performance over Euclidean distance is that the wide range of degrees for nodes in these graphs makes it difficult to find a single threshold that separates edges from non-edges. In particular, the "search engines" category had an extremely skewed degree distribution, and is where SPML shows the greatest improvement.

We also apply SPML to a larger subset of the Wikipedia network, by collecting word counts and connections of 100,000 articles in a breadth-first search rooted at the article "Philosophy". The experimental setup is the same as previous experiments, but we use a 0.5% sample of the nodes for testing. The final training algorithm ran for 50,000 iterations, taking approximately ten minutes on a desktop computer. The resulting AUC on the edges of the held-out nodes is listed in Table 6.1 as the "Philosophy Crawl" dataset. The SVM and RTM do not scale to data of this size, whereas SPML offers a clear advantage over using Euclidean distance for predicting links.

Facebook social networks Applying SPML to social network data allows us to more accurately predict who will become friends based on the profile information for those users. We use Facebook data [Traud et al., 2011], where we have a small subset of anonymized profile information for each student of a university, as well as friendship information. The profile information consists of gender, status (meaning student, staff, or faculty), dorm, major, and class year. Similarly to the Wikipedia experiments in the previous section, we compared SPML to Euclidean, RTM, and SVM. For SPML, we learn a full $\mathbf{M}$ via Algorithm 8. For each person, we construct a sparse feature vector where there is one

Table 6.1: Wikipedia (top), Facebook (bottom) dataset and experiment information. Shown below: number of nodes n, number of edges m, dimensionality d, and AUC performance.

	n	m	d	Euclidean	RTM	SVM	SPML
Graph Theory	223	917	6695	0.624	0.591	0.610	**0.722**
Philosophy Concepts	303	921	6695	0.705	0.571	**0.708**	0.707
Search Engines	269	332	6695	0.662	0.487	0.611	**0.742**
Philosophy Crawl	100,000	4,489,166	7702	0.547	–	–	**0.601**
Harvard	1937	48,980	193	0.764	0.562	0.839	**0.854**
MIT	2128	95,322	173	0.702	0.494	0.784	**0.801**
Stanford	3014	147,516	270	0.718	0.532	0.784	**0.808**
Columbia	3050	118,838	251	0.717	0.519	0.796	**0.818**

feature corresponding to every possible dorm, major, etc. for each feature type. We select only people who have indicated all five feature types on their profiles. Table 6.1 shows details of the Facebook networks for the four schools we consider: Harvard, MIT, Stanford, and Columbia. We perform a separate experiment for each school, randomly splitting the data 80/20 for training and testing. We use the training data to select parameters via five-fold cross validation, and train a model. The AUC performance on the held-out edges are also listed in Table 6.1. It is clear from the quantitative results that structural information is contributing to higher performance for SPML as compared to other methods.

By looking at the weight of the diagonal values in $\mathbf{M}$ normalized by the total weight, we can determine which feature differences are most important for determining connectivity. Figure 6.3 shows the normalized weights averaged by feature types for Facebook data. Here we see the feature types compared across four schools. For all schools except MIT, the graduating year is most important for determining distance between people. For MIT, dorms are the most important features. A possible explanation for this difference is that MIT is the only school in the list that makes it easy for students to stay in a residence for all four years of their undergraduate program, and therefore which dorm one lives in may

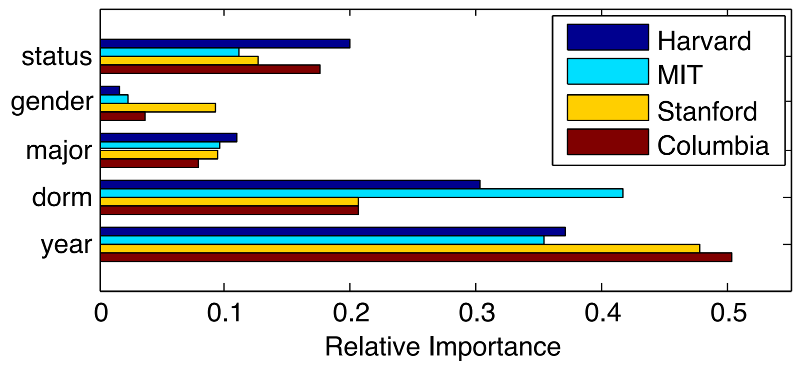


Figure 6.3: Comparison of Facebook social networks from four schools in terms of feature importance computed from the learned structure preserving metric.

affect more strongly the people they connect to.

6.4 Degree Distributional Metric Learning

While SPML using k-nearest neighbors learns a structure preserving metric, one of its limitations is in predicting full graphs in an out-of-sample setting. On the training data, the degree of each node is known, so the connectivity algorithm connects the exact number of neighbors as necessary to reconstruct the input graph. On a new set of nodes, however, the target degree is unknown. One method to address this is to learn a non-stationary *degree preference function* over node features that relates the features of a node to its target degree.

As one possible variant to *structure preserving metric learning* (SPML), *degree distributional metric learning* (DDML) simultaneously learns a metric while also learning a parameterized, non-stationary degree preference function used to compute the connectivity of nodes. This extension can be understood as SPML with an adaptive connectivity algorithm, rather than the default k-nearest neighbors.

The connectivity algorithm uses a degree preference function g, which takes a node's feature vector $\mathbf{x}$ and a target degree k, and is parameterized by matrix $\mathbf{S} \in \mathbb{R}^{d \times n}$. The score is then computed via

$$g(k|\mathbf{x};\mathbf{S}) = \sum_{k'=1}^{k} \mathbf{x}^\top \mathbf{s}_{\mathbf{k'}}.$$

The score of a graph A is then the sum of all edge distances and the degree preference functions for each node

$$F(\mathbf{A}|\mathbf{X};\mathbf{M},\mathbf{S}) = \sum_{ij} A_{ij} D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) - \sum_i g\left(\sum_j A_{ij}|\mathbf{x}_i;\mathbf{S}\right).$$

The objective for DDML is otherwise analogous to that of SPML:

$$f(\mathbf{M}) = \frac{\rho}{2}||\mathbf{M}||^2 - \sum_{\tilde{\mathbf{A}}\in\mathbb{B}^{n\times n}} \max(F(\mathbf{A}|\mathbf{X};\mathbf{M},\mathbf{S}) - F(\tilde{\mathbf{A}}|\mathbf{X};\mathbf{M},\mathbf{S}) + \Delta(\mathbf{A},\tilde{\mathbf{A}}),0),$$

where Δ denotes Hamming distance. This objective is solvable via the cutting-plane style optimization by iteratively finding the worst-violating $\tilde{\mathbf{A}}$ and adding it to a constraint set. For concave degree preference functions, the worst-violated constraint can be found by converting the problem to a maximum weight b-matching on an augmented graph [Huang and Jebara, 2009], thus an additional concavity constraint on g is added to the optimization.

A similar approach to the stochastic SPML algorithm is also possible to perform DDML much faster, and, by parameterizing the degree preference function only up to a fixed maximum degree, also eliminates the dependence of the running time on the size of the graph. As in stochastic SPML, a DDML objective can be written in terms of triplets of nodes i, neighbor j, disconnected node triplets k. Let $\mathbf{A}^{(i,j,k)}$ denote the false graph produced by toggling the edge between nodes i and j and the edge between nodes i and k. The DDML objective using the triplet-style constraints is

$$f^{\text{deg}}(\mathbf{M}) = \frac{\rho}{2}||\mathbf{M}||^2 - \frac{1}{|S|}\sum_{(i,j,k)\in S} \max(F(\mathbf{A}|\mathbf{X};\mathbf{M},\mathbf{S}) - F(\mathbf{A}^{(i,j,k)}|\mathbf{X};\mathbf{M},\mathbf{S}) + 1,0).$$

The difference in scores decomposes into four scalar values, since the only differences changing $\mathbf{A}$ to $\mathbf{A}^{(i,j,k)}$ are that $\mathbf{A}^{(i,j,k)}$ is missing edge (i,j), gains edge (i,k), the degree of node j decreases by one and the degree of node k increases by one. Thus, the difference can be computed by evaluating the distance from node i to node j, the distance from node i to node k, the change in degree preference score from the degree of node j to its degree minus one, and the change in degree preference from the degree of node k from its degree plus one. Let the degrees of all nodes be stored in array c, such that the degree of node j is $c[j]$. The difference is then computable as

$$F(\mathbf{A}|\mathbf{X};\mathbf{M},\mathbf{S})-F(\mathbf{A}^{(i,j,k)}|\mathbf{X};\mathbf{M},\mathbf{S}) = D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)-D_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k)+\mathbf{x}_j^\top \mathbf{s}_{(\mathbf{c[j]-1})}-\mathbf{x}_\mathbf{k}^\top \mathbf{s}_{(\mathbf{c[k]+1})}.$$

This formulation eliminates the need for the expensive separation oracle and allows stochastic optimization. The gradient update for the metric parameter $\mathbf{M}$ is the same as in SPML. The gradient with respect to $\mathbf{s}_{(\mathbf{c[j]-1})}$ is $\mathbf{x}_j$ and the gradient with respect to $\mathbf{s}_{(\mathbf{c[k]+1})}$ is $(-\mathbf{x}_k)$.

To retain coherence between the different degree functions, we add a requirement that the resulting degree preference function for each node is concave. One way to enforce concavity is by stochastically sampling a node i per iteration, and projecting $\mathbf{S}$ such that entries in $\mathbf{x}_i^\top \mathbf{S}$ are in decreasing order.

The pseudocode for stochastic DDML is in Algorithm 9.

6.4.1 Experiments

Using DDML on the same Wikipedia experiments from Table 6.1, we score comparable AUC to SPML. On “graph theory”, “philosophy concepts”, and “search engines”, DDML scores AUCs of 0.691, 0.746, and 0.725. While these scores are quite close to those of SPML, the DDML variant provides a tradeoff between running time and model richness. In the case of the Wikipedia category “philosophy concepts”, DDML even provides a performance improvement, which may indicate a clear signal in degree preference learnable from the word counts.

6.5 Low-rank structure preserving metric learning

In this section, we present the low-rank variant of SPML first introduced in Section 2.4. The low-rank variant computes all distances using a factorization $\mathbf{L} \in \mathbb{R}^{r\times d}$ of $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$, eliminating the need to compute a $d \times d$ matrix. Existing metric learning algorithms use similar low-rank factorizations Weinberger and Saul [2009]. Low-rank SPML has an additional parameter r, which limits the rank of $\mathbf{M}$ by explicitly determining the size of $\mathbf{L}$. The

Algorithm 9 Stochastic degree distributional metric learning

```
Input: $\mathbf{A} \in \mathbb{B}^{n \times n}$, $\mathbf{X} \in \mathbb{R}^{d \times n}$, and parameters $\rho, T, B$
 1: $\mathbf{M}_1 \leftarrow \mathbf{I}_d$, $\mathbf{S}_1 \leftarrow \mathbf{0}_{d,n}$
 2: Compute degree array $c$ s.t. $c[i] = \sum_j A_{ij}, \forall i$
 3: for $t$ from 1 to $T-1$ do
 4:     $\eta_t \leftarrow \frac{1}{\rho t}$
 5:     $\mathbf{E} \leftarrow \mathbf{0}_{n,n}$
 6:     $\mathbf{S}' \leftarrow \rho \mathbf{S}$
 7:     for $b$ from 1 to $B$ do
 8:         $(i,j,k) \leftarrow$ Sample random triplet from $S = \{(i,j,k) \mid A_{ij} = 1, A_{ik} = 0\}$
 9:         if $F(\mathbf{A}|\mathbf{X}; \mathbf{M}_t, \mathbf{S}_t) - F(\mathbf{A}^{(i,j,k)}|\mathbf{X}; \mathbf{M}_t, \mathbf{S}_t) + 1 > 0$ then
10:             $\mathbf{E}_{jj} \leftarrow \mathbf{E}_{jj} + 1$, $\mathbf{E}_{ik} \leftarrow \mathbf{E}_{ik} + 1$, $\mathbf{E}_{ki} \leftarrow \mathbf{E}_{ki} + 1$
11:             $\mathbf{E}_{ij} \leftarrow \mathbf{E}_{ij} - 1$, $\mathbf{E}_{ji} \leftarrow \mathbf{E}_{ji} - 1$, $\mathbf{E}_{kk} \leftarrow \mathbf{E}_{kk} - 1$
12:             $\mathbf{s}'_{\mathbf{c}[\mathbf{j}]} \leftarrow \mathbf{s}'_{\mathbf{c}[\mathbf{j}]} + \mathbf{x_j}$
13:             $\mathbf{s}'_{\mathbf{c}[\mathbf{k}]} \leftarrow \mathbf{s}'_{\mathbf{c}[\mathbf{k}]} - \mathbf{x_k}$
14:         end if
15:     end for
16:     $\nabla_t \leftarrow \mathbf{X}\mathbf{E}\mathbf{X}^\top + \rho \mathbf{M}_t$
17:     $\mathbf{M}_{t+1} \leftarrow \mathbf{M}_t - \eta_t \nabla_t$
18:     $\mathbf{S}_{t+1} \leftarrow \mathbf{S}_t - \eta_t \mathbf{S}'$
19:     $i \leftarrow$ Sample random index
20:     Project $\mathbf{S}$ so $\mathbf{x}_i^\top \mathbf{S}$ is monotonically nonincreasing
21:     Optional: $\mathbf{M}_{t+1} \leftarrow [\mathbf{M}_{t+1}]^+$ {Project onto the PSD cone}
22: end for
23: return $\mathbf{M}_T$
```

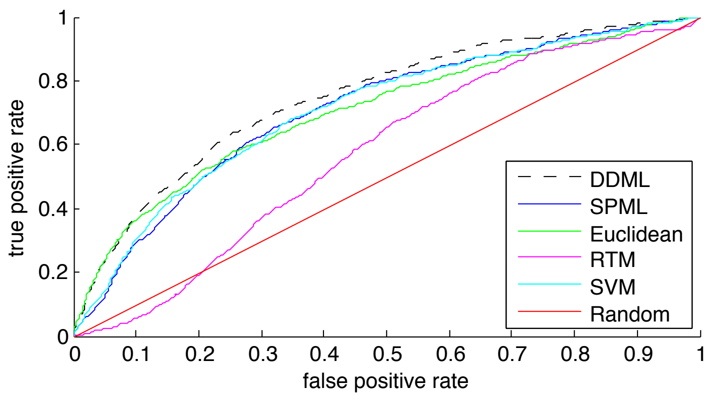


Figure 6.4: ROC curve for various algorithms on the "philosophy concepts" category.

optional projection onto the PSD cone is no longer necessary because $\mathbf{L}^\top\mathbf{L}$ always forms a valid metric by construction. This optimization not convex, but initial experimental results seem to show that the stochastic optimization avoids local minima in practice. Algorithm 10 details the steps of low-rank SPML.

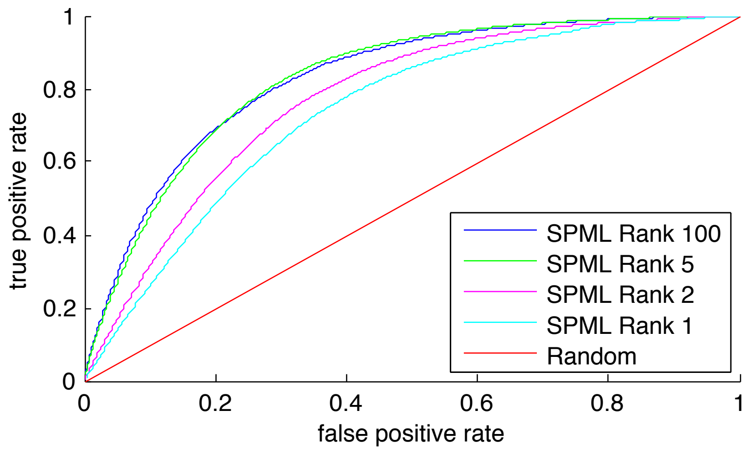
We run low-rank SPML on the Harvard Facebook data, fixing $\rho = 1\text{e}-5$ and varying the rank parameter r. The ROC curves and AUC scores using training data for different ranks are in Figure 6.5. With greater rank, SPML has more flexibility to construct a metric that fits the training data, but lower rank provides a tradeoff between efficiency and reconstruction quality. It is clear from this dataset that a rank of $r = 5$ is sufficient to represent the structure preserving metric, while reducing the number of parameters from d^2 =37,249 to $d \times r = 965$. Training fewer parameters requires less time, and allows low-rank SPML to handle large-scale networks with many nodes and high-dimensional features.

6.6 Discussion

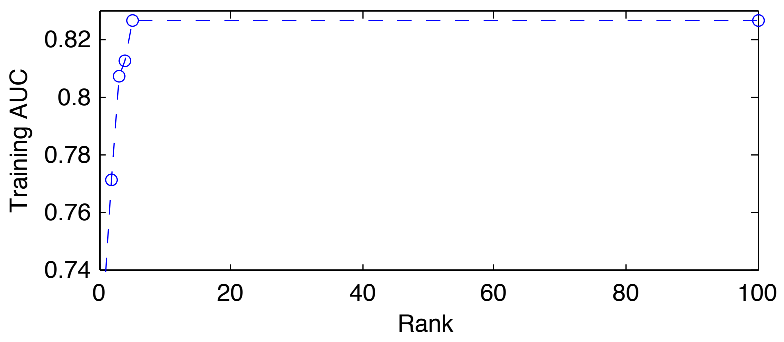
We have demonstrated a fast convex optimization for learning a distance metric from a network such that the distances are tied to the network's inherent topological structure. The structure preserving distance metrics introduced in this article allow us to better model and predict the behavior of large real-world networks. Furthermore, these metrics are

Algorithm 10 Low-rank structure preserving metric learning with nearest neighbor constraints and optimization with projected stochastic subgradient descent

```
Input: $\mathbf{A} \in \mathbb{B}^{n \times n}$, $\mathbf{X} \in \mathbb{R}^{d \times n}$, and parameters $\rho, T, B, r$
 1: $\mathbf{L}_1 \leftarrow \mathrm{rand}(r, d)$ {Initialize $\mathbf{L}$}
 2: for $t$ from 1 to $T-1$ do
 3:   $\eta_t \leftarrow \frac{1}{\rho t}$
 4:   $\mathbf{E} \leftarrow \mathbf{0}_{n,n}$
 5:   for  $b$ from 1 to $B$ do
 6:     $(i, j, k) \leftarrow$ Sample random triplet from $S = \{(i, j, k) \mid A_{ij} = 1, A_{ik} = 0\}$
 7:     if  $||\mathbf{L}_t\mathbf{x}_i - \mathbf{L}_t\mathbf{x}_j||^2 - ||\mathbf{L}_t\mathbf{x}_i - \mathbf{L}_t\mathbf{x}_k||^2 + 1 > 0$ then
 8:       $\mathbf{E}_{jj} \leftarrow \mathbf{E}_{jj} + 1$, $\mathbf{E}_{ik} \leftarrow \mathbf{E}_{ik} + 1$, $\mathbf{E}_{ki} \leftarrow \mathbf{E}_{ki} + 1$
 9:       $\mathbf{E}_{ij} \leftarrow \mathbf{E}_{ij} - 1$, $\mathbf{E}_{ji} \leftarrow \mathbf{E}_{ji} - 1$, $\mathbf{E}_{kk} \leftarrow \mathbf{E}_{kk} - 1$
10:     end if
11:   end for
12:   $\nabla_t \leftarrow 2\mathbf{X}\mathbf{E}\mathbf{X}^\top\mathbf{L}_t^\top + \rho\mathbf{L}_t$
13:   $\mathbf{L}_{t+1} \leftarrow \mathbf{L}_t - \eta_t\nabla_t$
14: end for
15: return  $\mathbf{L}_T$
```



(a) Training ROC curve for different ranks



(b) Training AUC as a function of rank

Figure 6.5: Performance of low-rank SPML on training data varying the rank parameter, run on a single Facebook school. The results imply that a significantly smaller rank than the true feature dimensionality is sufficient to fit the training data.

as lightweight as independent pairwise models, but capture structural dependency from features making them easy to use in practice for link-prediction.

Chapter 7

Conclusion

This thesis proposes a set of novel graph embedding and dimensionality reduction algorithms: MVE, SPE, and SPML. By building on the foundation of spectral methods such as PCA, but relaxing the constraint of preserving all pairwise distances to only preserving local distances and graph topology, these algorithms are able to better capture the inherent structure in datasets which have non-linear variations. In this thesis we highlight two important aspects of graph embedding: creating low-dimensional embeddings, and preserving graph structure. We present a variety of objectives for enforcing low-dimensional embeddings, and schemes for preserving different kinds of graph structure using linear constraints. We further demonstrate methods for efficiently implementing these algorithms using semidefinite programs and stochastic gradient descent, and show the utility of these graph embedding algorithms for applications to visualization, dimensionality reduction, and modeling problems. The key contributions of this thesis are the following:

- The "spectral fidelity" measure was derived and shown to be a natural objective for dimensionality reduction
- Minimum Volume Embedding (MVE) was developed as a tool for nonlinear dimensionality reduction and weighted graph embedding. MVE was shown to provide a natural extension to PCA for recovering nonlinear manifolds and capturing more variance in fewer dimensions than many state-of-the-art algorithms on a variety of synthetic and real-world experiments.

- Structure Preserving Embedding (SPE) was developed for embedding unweighted undirected graphs, and applied to visualizing many synthetic and real-world graphs and networks.
- The general concept of preserving graph topology using linear constraints was developed and applied to graph embedding (SPE), dimensionality reduction (MVE+SP), and metric learning (SPML).
- Structure Preserving Metric Learning (SPML) was developed and shown to produce favorable results in a variety of link prediction tasks.
- A fast optimization scheme for SPE and SPML was proposed based on projected stochastic subgradient descent that allows these methods to scale to graphs with hundreds of thousands of nodes.

There exist several possible directions for future work. One obvious extension of SPML is to make the algorithm work in a truly online fashion. It should be possible to exploit SPML's lack of dependence on graph size to learn a structure preserving metric on massive-scale graphs, e.g., the entire Wikipedia site. Since each iteration requires only sampling a random node, following a link to a neighbor, and sampling a non-neighbor, this can all be done in an online fashion as the algorithm crawls a network such as the worldwide web, learning a metric that may gradually change over time. Another particular area of interest is incorporating structure preserving constraints into collaborative filtering algorithms. Many collaborative filtering algorithms such as maximum margin matrix factorization (MMMF) [Rennie and Srebro, 2005] seek to find low-rank representations of users and items that can be used for recommendation. These matrix factorizations can be viewed as a non-symmetric case of many of the spectral decompositions used in this thesis. Preserving local distances and graph topology allows us to better capture non-linearities in the data when using PCA; perhaps a similar technique can be applied to an SVD where we are embedding two distinct sets of objects. Another interesting area to explore is the connections between large-scale SPE and spring embedding techniques. Where SPE is a convex optimization that exactly preserves graph topology and therefore exists in a different realm than many popular spring embedding techniques for visualizing graphs, the large-scale version of SPE which uses stochastic gradient descent is approximate and no longer convex, and in fact the gradient update rule is similar to some existing spring embedding methods [Battista et al., 1999].

Chapter 8

Appendix

8.1 Proofs

8.1.1 Maximizing over Arbitrary Orthonormal Vectors

Theorem 4 *For any* $\mathbf{K} \in \mathbb{R}^{n\times n}$ *such that* $\mathbf{K} \succeq 0$, $\mathbf{K}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}, \lambda_i \geq \lambda_{i+1}$, *and* $\alpha_i \geq \alpha_{i+1}$ *for* $i = 1...n$*:*

$$\sum_{i=1}^{n} \alpha_i \lambda_i = \max_{\substack{\mathbf{u}_1, \ldots, \mathbf{u}_n \\ \mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}}} \operatorname{tr}\left(\mathbf{K} \sum_{i=1}^{n} \alpha_i \mathbf{u}_i \mathbf{u}_i^\top\right).$$

Proof First manipulate the left-hand side algebraically, maintaining the additional constraints:

$$\begin{aligned}
\sum_{i=1}^{n} \alpha_i \lambda_i &= \sum_{i=1}^{n} \operatorname{tr}(\alpha_i \lambda_i \mathbf{v}_i \mathbf{v}_i^\top) \\
&= \sum_{i=1}^{n} \operatorname{tr}(\alpha_i \mathbf{K} \mathbf{v}_i \mathbf{v}_i^\top) \\
&= \operatorname{tr}\left(\mathbf{K} \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \mathbf{v}_i^\top\right).
\end{aligned}$$

Then rewrite the quantity on the right-hand side as the product of two eigendecompositions of positive semidefinite matrices $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ and $\mathbf{Z} = \sum_{i=1}^{n} \alpha_i \mathbf{u}_i \mathbf{u}_i^\top = \mathbf{U}\mathbf{\Omega}\mathbf{U}^\top$ where $\mathbf{\Omega} =$

diag$(\alpha_1 \ldots \alpha_n)$:

$$\mathrm{tr}\left(\mathbf{K}\sum_{i=1}^{n}\alpha_i\mathbf{u}_i\mathbf{u}_i^\top\right) = \mathrm{tr}\left(\mathbf{KZ}\right) = \mathrm{tr}\left(\mathbf{V\Lambda V}^\top\mathbf{U\Omega U}^\top\right).$$

Instead of optimizing over an arbitrary set of vectors, we can equivalently maximize $\mathbf{U}$ over the set of orthonormal matrices $\mathcal{O}$, also known as the Stiefel manifold. Since tr$(\mathbf{MN})$ = tr$(\mathbf{NM})$, we can rewrite the above as:

$$\begin{aligned}\max_{\substack{\mathbf{u}_1,\ldots,\mathbf{u}_n\\ \mathbf{u}_i^\top\mathbf{u}_j=\delta_{ij}}} \mathrm{tr}\left(\mathbf{K}\sum_{i=1}^{n}\alpha_i\mathbf{u}_i\mathbf{u}_i^\top\right) &= \max_{U\in\mathcal{O}}\mathrm{tr}\left(\mathbf{V\Lambda V}^\top\mathbf{U\Omega U}^\top\right)\\ &= \max_{U\in\mathcal{O}}\mathrm{tr}((\mathbf{U}^\top\mathbf{V})\mathbf{\Lambda}(\mathbf{U}^\top\mathbf{V})^\top\mathbf{\Omega})\\ &= \max_{R\in\mathcal{O}}\mathrm{tr}(\mathbf{R\Lambda R}^\top\mathbf{\Omega}).\end{aligned}$$

This optimization over $\mathbf{R}$ is known as a homogenous quadratically constrained quadratic program (QQP) [Anstreicher and Wolkowicz, 2001] and has the solution $\mathbf{R}^*$ as the product of the diagonalizers of $\mathbf{\Lambda}$ and $\mathbf{\Omega}$. Because both $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ are already diagonal matrices, $\mathbf{R}^*$ must be a permutation matrix, and if the diagonal entries of $\mathbf{\Lambda}$ and $\mathbf{\Omega}$ are arranged in decreasing order, then $\mathbf{R}^* = \mathbf{I}$ and the optimal eigenvectors of $\mathbf{Z}$ must satisfy $\mathbf{U} = \mathbf{V}$. Thus, maximizing g over arbitrary orthonormal vectors

$$\max_{\substack{\mathbf{u}_1,\ldots,\mathbf{u}_n\\ \mathbf{u}_i^\top\mathbf{u}_j=\delta_{ij}}} \mathrm{tr}\left(\mathbf{K}\sum_{i=1}^{n}\alpha_i\mathbf{u}_i\mathbf{u}_i^\top\right)$$

is *equivalent* to setting those vectors to

$$\mathrm{tr}\left(\mathbf{K}\sum_{i=1}^{n}\alpha_i\mathbf{v}_i\mathbf{v}_i^\top\right) \quad \text{s.t. } \mathbf{K}\in\mathcal{K},\ \mathbf{K}\mathbf{v}_i=\lambda_i\mathbf{v}_i,\ \mathbf{v}_i^\top\mathbf{v}_j=\delta_{ij},\ \lambda_i\geq\lambda_{i+1},\ \alpha_i\geq\alpha_{i+1},\ \forall i,j$$

under the more rigid set of constraints. In other words, maximization over arbitrary orthonormal vectors is equivalent to setting them to the eigenvectors of the matrix $\mathbf{K}$. ■

8.1.2 Objective Function of MVE Monotonically Increases

Theorem 5 *The iterative MVE algorithm is guaranteed to monotonically increase the objective function* $F_\beta(\mathbf{K}, d) = \beta \sum_{i=1}^{d} \lambda_i - \sum_{i=1}^{n} \lambda_i$ *where* $\mathbf{K} \in \mathbb{R}^{n \times n}, \mathbf{K} \succeq 0, \mathbf{K}v_i = \lambda_i v_i, v_i^\top v_j = \delta_{ij}, \lambda_i \geq \lambda_{i+1}$ *and* $\beta \geq 0$.

Proof For any $\mathbf{K} \in \mathcal{K}$, recall that $F_\beta(\mathbf{K}, d) =$

$$\max_{\substack{\mathbf{v}_1, \ldots, \mathbf{v}_n \\ \mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}}} \operatorname{tr}\left[K\left(\beta \sum_{i=1}^{d} \mathbf{v}_i \mathbf{v}_i^\top - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^\top\right)\right].$$

Define the function:

$$g(\mathbf{K}, \mathbf{v}_1, \ldots, \mathbf{v}_n) = \operatorname{tr}\left[\mathbf{K}\left(\beta \sum_{i=1}^{d} \mathbf{v}_i \mathbf{v}_i^\top - \sum_{i=1}^{n} \mathbf{v}_i \mathbf{v}_i^\top\right)\right].$$

Thus, $F_\beta(\mathbf{K}, d)$ is the maximization of $g(\mathbf{K}, \mathbf{v}_1, \ldots, \mathbf{v}_n)$ over eigenvectors. Assume we have a current setting of $\mathbf{K}$ denoted $\mathbf{K}^t$ and a current setting of the eigenvectors $\mathbf{v}_1^t, \ldots, \mathbf{v}_n^t$. In general, we must have:

$$F_\beta(\mathbf{K}^t, d) \quad \geq \quad g(\mathbf{K}^t, \mathbf{v}_1^t, \ldots, \mathbf{v}_n^t).$$

After Step 5 of the MVE algorithm we obtain:

$$F_\beta(\mathbf{K}^t, d) \quad = \quad g(\mathbf{K}^t, \mathbf{v}_1^{t+1}, \ldots, \mathbf{v}_n^{t+1}).$$

After Step 7 of the MVE algorithm, the SDP ensures that:

$$g(\mathbf{K}^{t+1}, \mathbf{v}_1^{t+1}, \ldots, \mathbf{v}_n^{t+1}) \quad \geq \quad g(\mathbf{K}^t, \mathbf{v}_1^{t+1}, \ldots, \mathbf{v}_n^{t+1}).$$

Since $F_\beta(\mathbf{K}, d)$ is a maximization of $g(\mathbf{K}, \mathbf{v}_1, \ldots, \mathbf{v}_n)$,

$$\begin{aligned} F_\beta(\mathbf{K}^{t+1}, d) &\geq g(\mathbf{K}^{t+1}, \mathbf{v}_1^{t+1}, \ldots, \mathbf{v}_n^{t+1}) \\ &\geq g(\mathbf{K}^t, \mathbf{v}_1^{t+1}, \ldots, \mathbf{v}_n^{t+1}) \\ &\geq F_\beta(\mathbf{K}^t, d) \end{aligned}$$

showing that $F_\beta(\mathbf{K}^t, d) \leq F_\beta(\mathbf{K}^{t+1}, d)$ after each loop of the MVE algorithm and $F_\beta(\mathbf{K}, d)$ increases monotonically. ■

8.1.3 Low-rank Objective Function for SPE

Theorem 9 *For any symmetric binary matrix* $\mathbf{A}$*, such that* $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$*,* $\mathbf{v}_i^\top \mathbf{v}_j = \delta_{ij}$*,* $\lambda_i \geq \lambda_{i+1}$*,* $A_{ii} = 0$ *for* $i = 1...n$*, if* $\lambda_1 > \lambda_i$ *for* $i = 2...n$*, the objective function* $\max_{\mathbf{K}\succeq 0} \mathrm{tr}(\mathbf{KA})$ *subject to* $\mathrm{tr}(\mathbf{K}) \leq 1$ *is optimized by setting* $\mathbf{K} = \mathbf{v}_1\mathbf{v}_1^\top$*. If there exists an* i *such that* $\lambda_1 = \lambda_i$*, the objective function will recover* $\mathbf{K} = \sum_{i|\lambda_i=\lambda_1} \beta_i \mathbf{v}_i \mathbf{v}_i^\top$ *where* $\beta_i \geq 0$ *for* $i = 1...n$*.*

Proof Rewrite the matrices in terms of the eigendecomposition of the positive semidefinite matrix $\mathbf{K} = \mathbf{U\Lambda U}^\top$ and the symmetric matrix $\mathbf{A} = \mathbf{V}\tilde{\mathbf{\Lambda}}\mathbf{V}^\top$, and insert into the objective function:

$$\max_{\mathbf{K}\succeq 0} \mathrm{tr}(\mathbf{KA}) \quad = \quad \max_{\mathbf{\Lambda}\in\mathcal{L},\mathbf{U}\in\mathcal{O}} \mathrm{tr}(\mathbf{U\Lambda U}^\top \mathbf{V}\tilde{\mathbf{\Lambda}}\mathbf{V}^\top)$$

where $\mathcal{L}$ is the set of positive semidefinite diagonal matrices and $\mathcal{O}$ is the set of orthonormal matrices also known as the Stiefel manifold. By Von Neumann's lemma, we have:

$$\max_{\mathbf{U}\in\mathcal{O}} \mathrm{tr}(\mathbf{U\Lambda U}^\top \mathbf{V}\tilde{\mathbf{\Lambda}}\mathbf{V}^\top) \quad = \quad \boldsymbol{\lambda}^\top \tilde{\boldsymbol{\lambda}}.$$

Here $\boldsymbol{\lambda}$ is a vector containing the diagonal entries of $\mathbf{\Lambda}$ in decreasing order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and $\tilde{\boldsymbol{\lambda}}$ is a vector containing the diagonal entries of $\tilde{\mathbf{\Lambda}}$ in decreasing order, i.e. $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \ldots \geq \tilde{\lambda}_n$. Therefore, the full optimization problem can be rewritten in terms of an optimization over non-negative eigenvalues

$$\max_{K\succeq 0, \mathrm{tr}(K)\leq 1} \mathrm{tr}(\mathbf{KA}) \quad = \quad \max_{\boldsymbol{\lambda}\geq 0, \boldsymbol{\lambda}^\top \mathbf{1} \leq 1} \boldsymbol{\lambda}^\top \tilde{\boldsymbol{\lambda}}$$

where we also have to satisfy $\boldsymbol{\lambda}^\top \mathbf{1} \leq 1$ since $\mathrm{tr}(\mathbf{K}) \leq 1$. To maximize the objective function $\boldsymbol{\lambda}^\top \tilde{\boldsymbol{\lambda}}$ simply set $\lambda_1 = 1$ and the remaining $\lambda_i = 0$ for $i = 2, \ldots, n$ which produces the maximum $\tilde{\lambda}_1$, the top eigenvalue of $\mathbf{A}$.

$$\max_{K\succeq 0, \mathrm{tr}(K)\leq 1} \mathrm{tr}(\mathbf{KA}) \quad = \quad \tilde{\lambda}_1.$$

Thus, the solution to the maximization problem over $\mathbf{K}$ must be $\mathbf{K} = \mathbf{v}_1\mathbf{v}_1^\top$ (up to a scaling factor) where $\mathbf{v}_1$ is the leading eigenvector of $\mathbf{A}$. If there are ties in $\mathbf{A}$ for its top eigenvalues, the weight of $\boldsymbol{\lambda}$ can be spread arbitrarily among the leading eigenvalues of $\mathbf{A}$ to maximize $\boldsymbol{\lambda}^\top \tilde{\boldsymbol{\lambda}}$ and thus $\mathbf{K}$ is a conic combination of the outer products of the top eigenvectors of

A. In this case when there are multiple top eigenvalues, **K** has rank at most equal to the multiplicity of the top eigenvalue of **A**. ■

Bibliography

M. Jordan A. Kleiner, A. Rahimi. Random conic pursuit for semidefinite programming. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*. 2010.

L. A. Adamic and N. Glance. The political blogosphere and the 2004 US election. In *WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

E. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. *JMLR*, 9:1981–2014, 2008.

K. Anstreicher and H. Wolkowicz. On Lagrangian relaxation of quadratic matrix constraints. *SIAM Journal on Matrix Analysis and Applications*, 22(1):41–55, 2001.

S. Arora, S. Rao, and U.V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Symposium on Theory of Computing*, 2004.

L. Asimov and B. Roth. The rigidity of graphs. *Trans. Amer. Math. Soc.*, 1978.

G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing: algorithms for the visualization of graphs.* Prentice Hall, 1999. ISBN 0-13-301615-3.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2002.

B. Borchers. CSDP, a C library for semidefinite programming, 1997.

Léon Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, pages 146–168, 2003.

M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. *ACM International Conference Proceeding Series*, 119:65–72, 2005.

Stephen Boyd. Convex optimization of graph laplacian eigenvalues. In *in International Congress of Mathematicians*, pages 1311–1319.

S. Burer and R. D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (series B)*, 95(2):329–357, 2003.

J. Chang and D. Blei. Hierarchical relational models for document networks. *Annals of Applied Statistics*, 4:124–150, 2010. doi: 10.1214/09-AOAS309.

G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *J. Mach. Learn. Res.*, 11:1109–1135, March 2010.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, February 1997. ISBN 0821803158.

T. Cox and M.Cox. *Multidimensional Scaling*. Chapman & Hall, 1994.

S. Dasgupta and A. Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22:60–65, January 2003. ISSN 1042-9832. doi: 10.1002/rsa.10073. URL `http://portal.acm.org/citation.cfm?id=639790.639795`.

T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *Proc. of 25^{th} International Conference on Machine Learning*, pages 304–311, 2008.

C. Fremuth-Paeger and D. Jungnickel. Balanced network flows, a unifying framework for design and analysis of matching algorithms. *Networks*, 33(1):1–28, 1999.

M. X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

B. Huang and T. Jebara. Loopy belief propagation for bipartite maximum weight b-matching. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of JMLR: W&CP, pages 195–202, 2007.

B. Huang and T. Jebara. Exact graph structure estimation with degree priors. In *ICMLA*, pages 111–118, 2009.

B. Huang and T. Jebara. Fast b-matching via sufficient selection belief propagation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

A. Javanmard and A. Montanari. Localization from incomplete noisy distance measurements. March 2011.

T. Joachims. Training linear SVMs in linear time. In *ACM SIG International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 217 – 226, 2006.

T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

W. Johnson and J. Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, (26):189–206, 1984.

I. T. Jolliffe. *Principal Components Analysis.* Springer-Verlag, 1986.

C. C. Kemp, T. L. Griffiths, S. Stromsten, and J. B. Tenenbaum. Semi-supervised learning with trees. In *In Advances in Neural Information Processing Systems*, page 2004. MIT Press, 2003.

Jon Kleinberg and Michel X. Goemans. The lovasz theta function and a semidefinite programming relaxation of vertex cover. *SIAM Journal on Discrete Mathematics*, 11:196–204, 1995.

Yehuda Koren. Drawing graphs by eigenvectors: Theory and practice. *Computers and Mathematics with Applications*, 49:2005, 2005.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. *ACM WWW*, 2008. URL `http://scholar.google.de/scholar.bib?q=info:dvYmn_qj6NQJ:scholar.google.com/&output=citation&hl=de&as_sdt=2000&ct=citation&cd=0`.

J. Leskovec, J Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, March 2002.

M. Middendorf, E. Ziv, C. Adams, J. Hom, R. Koytcheff, C. Levovitz, and G. Woods. Discriminative topological features reveal biological network mechanisms. *BMC Bioinformatics*, 5:1471–2105, 2004.

G. Namata, H. Sharara, and L. Getoor. A survey of link mining tasks for analyzing noisy and incomplete networks. In *Link Mining: Models, Algorithms, and Applications*. Springer, 2010.

Angelia Nedic and Dimitri P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optimization*, 12:109–138, January 2001. ISSN 1052-6234. doi: http://dx.doi.org/10.1137/S1052623499362111. URL `http://dx.doi.org/10.1137/S1052623499362111`.

M. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167–256, 2003.

M. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131, Nov 2004. doi: 10.1103/PhysRevE.70.056131.

M. Newman. Finding community structure in networks using the eigenvectors of matrices. *Arxiv preprint physics/0605087*, 2006.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Neural Information Processing Systems 14*, 2001.

M. L. Overton and R. S. Womersley. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Mathematical Programming*, 62(1):321–357, 1993.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web, 1999.

R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

R. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7: 179–188, 1936.

J. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the Twenty-Second International Conference*, volume 119 of *ACM International Conference Proceeding Series*, pages 713–719. ACM, 2005. ISBN 1-59593-180-5.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

G. Sadetsky. Aol search data mirrors, May 2006. URL `http://www.gregsadetsky.com/aol-data/`.

B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Compuation*, 10, 1998.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 807–814, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: http://doi.acm.org/10.1145/1273496.1273598. URL `http://doi.acm.org/10.1145/1273496.1273598`.

S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, To appear.

B. Shaw and T. Jebara. Minimum volume embedding. In Marina Meila and Xiaotong Shen, editors, *Proc. of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2 of JMLR: W&CP, pages 460–467, March 2007.

B. Shaw and T. Jebara. Minimum Volume Embedding (MVE) Matlab Code. http://www.metablake.com/mve, 2008.

B. Shaw and T. Jebara. Structure preserving embedding. In *Proc. of the 26th International Conference on Machine Learning*, 2009.

B. Shaw and T. Jebara. Nonlinear dimensionality reduction and graph embedding. *Journal of Machine Learning Research (In submission)*, 2011a.

B. Shaw and T. Jebara. Visualizing social networks with structure preserving embedding. *Interdisciplinary Workshop on Information and Decision in Social Networks*, 2011b.

Amit Singer and Mihai Cucuringu. Uniqueness of low-rank matrix completion by rigidity theory. *CoRR*, abs/0902.3846, 2009.

L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, 1987.

L. Song, A. Smola, K. Borgwardt, and A. Gretton. Colored maximum variance unfolding. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, pages 1385–1392. MIT Press, Cambridge, MA, 2008.

D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1393–1400, 2008.

Nathan Srebro. Learning with matrix factorizations. *PhD Thesis, Massachusetts Institute of Technology*, 2004.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

A. Traud, P. Mucha, and M. Porter. Social structure of facebook networks. *CoRR*, abs/1102.2166, 2011.

L.J.P van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009. ISSN 1532-4435.

K. Weinberger, F. Sha, Q. Zhu, and L. Saul. Graph laplacian methods for large-scale semidefinite programming, with an application to sensor localization. In B. Schölkopf, J. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21st International Conference on Machine Learning*, pages 839–846, Banff, Canada, 2004.

K. Q. Weinberger, B. D. Packer, and L. K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proc. of the of the 10th International Workshop on Artificial Intelligence and Statistics*, pages 381–388, 2005.

Christpher K.I. Williams. On a connection between kernel pca and metric multidimensional scaling. In *Advances in Neural Information Processing Systems 13*, pages 675–681. MIT Press, 2001.

J. Xu and Y. Li. Discovering disease-genes by topological features in human protein-protein interaction network. *Bioinformatics*, 22(22):2800–2805, 2006. doi: 10.1093/bioinformatics/btl467. URL `http://bioinformatics.oxfordjournals.org/content/22/22/2800.abstract`.

T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 927–936, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: http://doi.acm.org/10.1145/1557019.1557120. URL `http://doi.acm.org/10.1145/1557019.1557120`.

J. Zhuang, I. W. Tsang, and S. C. H. Hoi. Simplenpkl: simple non-parametric kernel learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, New York, NY, USA, 2009. ACM.