Structure Preserving Embedding

Blake Shaw BLAKE@CS.COLUMBIA.EDU
Tony Jebara JEBARA@CS.COLUMBIA.EDU
Department of Computer Science, Columbia University, 1214 Amsterdam Ave, New York, NY 10027

Abstract

Structure Preserving Embedding (SPE) is an algorithm for embedding graphs in Euclidean space such that the embedding is low-dimensional and preserves the global topological properties of the input graph. Topology is preserved if a connectivity algorithm, such as k-nearest neighbors, can easily recover the edges of the input graph from only the coordinates of the nodes after embedding. SPE is formulated as a semidefinite program that learns a low-rank kernel matrix constrained by a set of linear inequalities which captures the connectivity structure of the input graph. Traditional graph embedding algorithms do not preserve structure according to our definition, and thus the resulting visualizations can be misleading or less informative. SPE provides significant improvements in terms of visualization and lossless compression of graphs, outperforming popular methods such as spectral embedding and Laplacian eigenmaps. We find that many classical graphs and networks can be properly embedded using only a few dimensions. Furthermore, introducing structure preserving constraints into dimensionality reduction algorithms produces more accurate representations of high-dimensional data.

1. Introduction

Graphs are essential for encoding information, and graph and network data is increasingly abundant in fields ranging from computational biology to computer vision. Given a graph where vertices and edges represent pairwise interactions between entities (such as links between websites, friendships in social networks, or bonds between atoms in molecules), we hope to recover a low-dimensional set of coordinates for each vertex that implicitly encodes the graph's binary connectivity.

Graph embedding algorithms place nodes at points on some surface (e.g. Euclidean space) and connect points with an arc if the nodes have an edge between them. One traditional objective of graph embedding is to place points on a surface such that arcs never cross. In this setting, it is well known that any graph can be embedded in 3-dimensional Euclidean space and planar graphs can be embedded in 2-dimensional Euclidean space. However, there are many uses for graph embedding that do not relate to arc crossing and thus there exists a suite of embedding algorithms with different goals (Chung, 1997; Battista et al., 1999). One motivation for embedding graphs is to solve computationally hard problems geometrically. For example, using hyperplanes to separate points after graph embedding is useful for efficiently approximating the NP-hard sparsest cut problem (Arora et al., 2004). This article will focus on graph embedding for visualization and compression. Given only the connectivity of a graph, can we efficiently recover low-dimensional point coordinates for each node such that these points can easily be used to reconstruct the original structure of the network? We are interested in reversible graph embeddings (from a graph to points back to a graph). If the graph can be easily reconstructed from the points and these require low-dimensionality (and storage), the method can be useful for both visualization and compression.

Many embedding algorithms find compact coordinates for nodes in a graph. Given an unweighted graph consisting of N nodes and $|E|$ edges represented as a symmetric adjacency matrix $A \in \{0,1\}^{N \times N}$ specifying which pairs of nodes are connected, spectral embedding finds a set of coordinates for each node $\vec{y_i} \in \mathbb{R}^d$ for $i = 1, \ldots, N$ by applying singular value decomposition (SVD) or principal component analysis (PCA) to the adjacency matrix A and using the d eigenvectors of A with the largest eigenvalues as the coordinates. Similarly, Laplacian eigenmaps (Belkin

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

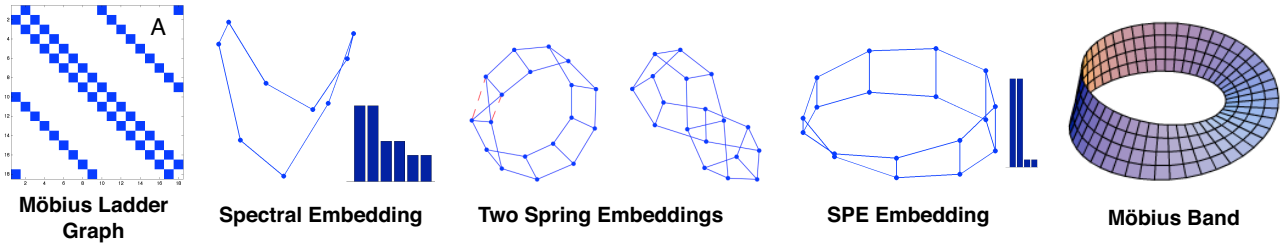

Figure 1. Embedding the classical Mobius Ladder Graph. Given the adjacency matrix (left), the visualizations produced by spectral embedding and spring embedding (middle) do not accurately capture the graph topology. The SPE embedding is compact and topologically correct.

& Niyogi, 2002) employ a spectral decomposition of the graph Laplacian $L = D - A$, or normalized graph Laplacian $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where $D = \text{diag}(A\mathbf{1})$ and use the d eigenvectors of L with the smallest non-zero eigenvalues. Unfortunately, in practice there are often many eligible eigenvectors of A or L, and furthermore the resulting coordinates do not preserve the topology of the input graph exactly. We propose learning a positive semi-definite kernel matrix $K \in \mathbb{R}^{N\times N}$ whose spectral decomposition yields a small set of eigenvectors which preserve the topology of the input graph. Specifically, given a connectivity algorithm $\mathcal{G}$ (such as k-nearest neighbors, b-matching, or maximum weight spanning tree) which accepts as input a kernel K specifying an embedding and returns an adjacency matrix, we call an embedding *structure preserving* if the application of $\mathcal{G}$ to K exactly reproduces the input graph: $\mathcal{G}(K) = A$. This article proposes SPE, an efficient convex optimization based on semidefinite programming for finding an embedding K such that K is both low-rank and structure preserving.

Traditional graph embedding algorithms such as spectral embedding and spring embedding do not explicitly preserve structure according to our definition and thus in practice produce poor visualizations of many simple classical graphs. In Figure 1 we see the classical Möbius ladder graph and the resulting visualizations from the two methods. The spectral embedding looks degenerate and does not resemble the Möbius band in any regard. The eigenspectrum indicates that the embedding is 6-dimensional when we expect to be able to embed this graph using fewer dimensions. Two spring embeddings are shown. The left spring embedding is a good diagram of what the graph should look like; however, we see that the twist of the Möbius strip is not accurately captured. Given the coordinates in Euclidean space produced by this method, any simple neighbor-finding algorithm $\mathcal{G}$ would connect nodes along the red dotted lines, not the blue ones specified by the connectivity matrix. Thus, the inherent connectivity of the embedding disagrees with the actual connectivity of the graph. The spring embedding on the right shows a typical result when a poor random initialization is used. Due to local minima in spring embedding algorithms, the graph is no longer visually recognizable or accurate. We are motivated to find a simple tool for properly visualizing graphs such as the Möbius ladder, as well as large complex network datasets. The tool should be accurate and should circumvent local minima issues.

Structure preserving constraints can also benefit dimensionality reduction algorithms. These methods similarly find compact coordinates that preserve certain properties of the input data. Multidimensional scaling preserves distances between data points (Cox & M.Cox, 1994). Nonlinear manifold learning algorithms preserve local distances described by a graph on the data (Tenenbaum et al., 2000; Roweis & Saul, 2000; Weinberger et al., 2005). For these algorithms the input consists of high-dimensional points as well as binary connectivity. Many of these manifold learning techniques preserve local distances but not graph topology. We show that adding explicit topological constraints to these existing algorithms is crucial for preventing folding and collapsing problems that occur in dimensionality reduction.

The rest of the article is organized as follows. In Section 2, we introduce the concept of structure preserving constraints and formulate these constraints as a set of linear inequalities for a variety of different connectivity algorithms. We then derive an objective function in Section 3 which favors low-dimensional embeddings close to the spectral embedding solution. In Section 4, we combine the structure preserving constraints and the objective function into a convex optimization and propose a semidefinite program for solving it efficiently. We present a variety of experiments on real and synthetic graphs in Section 5 and show improvements in terms of visualization quality and level of compression. We then briefly explore using SPE to improve dimensionality reduction algorithms before concluding in Section 7.

2. Preserving Graph Structure

We assume we are given an input graph defined by both a connectivity matrix A as well as an algorithm $\mathcal{G}$ which accepts as input a kernel matrix K, and outputs a connectivity matrix $\tilde{A} = \mathcal{G}(K)$, such that $\tilde{A}$ is close to the original graph A. In this article, we consider several choices for $\mathcal{G}$ including k-nearest neighbors, epsilon-balls, maximum weight spanning trees, maximum weight generalized matching and other maximum weight subgraphs. We can evaluate how well the embedding produced by K preserves graph structure by determining how much the input connectivity differs from the connectivity computed directly from the learned embedding. A simple metric to capture this difference is the normalized number of pairwise errors $\triangle(\tilde{A}, A) = \frac{1}{N^2}\sum_{ij}|\tilde{A}_{ij} - A_{ij}|$. For a variety of different classes of graphs and algorithms $\mathcal{G}$, it is possible to enumerate *linear* constraints on K to ensure that this error or difference is zero. The next subsections describe several choices for algorithm $\mathcal{G}$ and the linear constraints they entail on the embedding K.

2.1. Nearest Neighbor Graphs

The k-nearest neighbor algorithm ($k$nn) greedily connects each node to the k neighbors to which the node has shortest distance, where k is an input parameter. Preserving the structure of $k$nn graphs requires enumerating a small set of linear constraints on K.

Definition 1. *Define the distance between a pair of points (i, j) with respect to a given positive semidefinite kernel matrix K, as $D_{ij} = K_{ii} + K_{jj} - 2K_{ij}$.*

Note that the matrix D constructed from elements D_{ij} is a linear function of K. 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. This results in the linear constraints on K: $D_{ij} > (1 - A_{ij})\max_m(A_{im}D_{im})$. Another common connectivity scheme is to connect each node to all neighbors which lie within an epsilon ball of radius ϵ. Preserving this ϵ-neighborhood graph can be achieved with the linear constraints on K: $D_{ij}(A_{ij} - \frac{1}{2}) \leq \epsilon(A_{ij} - \frac{1}{2})$.

It is trivial to show that if for each node the connected distances are less than the unconnected distances (or some ϵ), a greedy algorithm will find the exact same neighbors for that node, and thus the connectivity computed from K is exactly A, and $\triangle(\mathcal{G}(K), A) = 0$.

2.2. Maximum Weight Subgraphs

Maximum weight subgraph algorithms select edges from a weighted graph to produce a subgraph which has maximal weight (Fremuth-Paeger & Jungnickel, 1999).

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

Once again, the matrix W composed of elements W_{ij} is simply a linear function of K. Given W, a maximum weight subgraph produces a graph with binary adjacency matrix $\tilde{A}$ which maximizes $\sum_{ij}\tilde{A}_{ij}W_{ij}$. For example when $\mathcal{G}$ is the generalized b-matching algorithm, $\mathcal{G}$ finds the connectivity matrix which has maximum weight while enforcing a set of degree constraints b_i for $i = 1...N$ as follows: $\mathcal{G}(K) = \arg\max_{\tilde{A}}\sum_{ij}W_{ij}\tilde{A}_{ij}$ s.t. $\sum_j \tilde{A}_{ij} = b_i$, $\tilde{A}_{ij} = \tilde{A}_{ji}$, $\tilde{A}_{ii} = 0$, $\tilde{A}_{ij} \in \{0, 1\}$. Similarly, a maximum weight spanning tree algorithm $\mathcal{G}$ returns the maximum weight subgraph $\mathcal{G}(K)$ such that $\mathcal{G}(K) \in \mathcal{T}$, where $\mathcal{T}$ is the set of all tree graphs: $\mathcal{G}(K) = \arg\max_{\tilde{A}}\sum_{ij}W_{ij}\tilde{A}_{ij}$ s.t. $\tilde{A} \in \mathcal{T}$. Unfortunately, for both these algorithms the linear constraints on K (or W) to preserve the structure in the original adjacency matrix A such that $\tilde{A} = A$ cannot be enumerated with a small finite set of linear inequalities; in fact, there can be an exponential number of constraints of the form: $\sum_{ij}W_{ij}A_{ij} \geq \sum_{ij}W_{ij}\tilde{A}_{ij}$. However, in Section 4 we present a cutting plane approach such that the exponential enumeration is avoided and the most violated inequalities are introduced sequentially. It has been shown that cutting-plane optimizations such as this converge and perform well in practice (Finley & Joachims, 2008).

3. A Low-Rank Objective

The previous section showed that it is possible to force an embedding to preserve the graph structure in a given adjaceny matrix A by introducing a set of linear constraints on the embedding Gram matrix K. To choose a unique K from the admissible set in the convex hull generated by these linear constraints, we propose an objective function which favors low-dimensional embeddings close to the spectral embedding solution. Spectral embedding is the canonical way of mapping nodes of a graph into points with coordinates. Given a symmetric adjacency matrix A, it uses the eigenvectors of $A = V\Lambda V^T$ with the largest k eigenvalues as coordinates of the points. Spectral embedding produces embeddings with few dimensions, since it uses the most dominant eigenvectors first. Similarly, we are interested in recovering an embedding, or equivalently, a positive semidefinite kernel matrix $K \succeq 0$ which is low-rank. Consider the following objective function $\max_{K\succeq 0}\mathrm{tr}(KA)$ and limit the trace

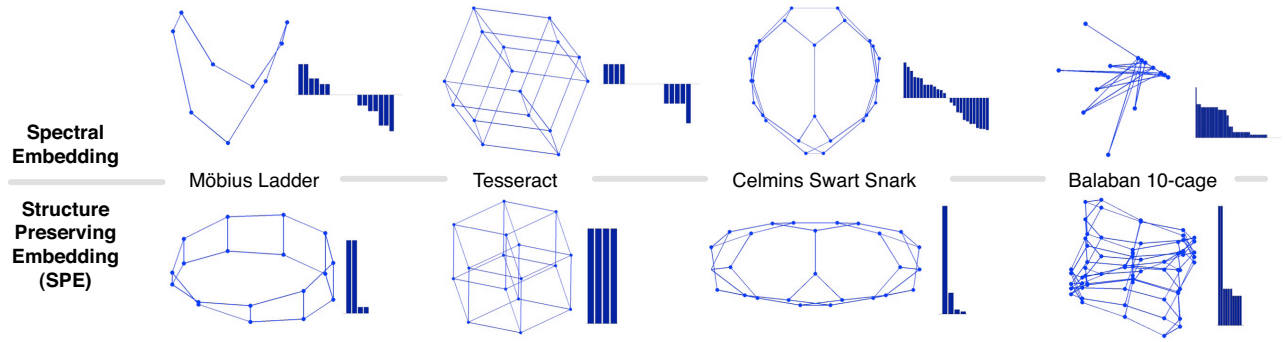

Figure 2. Classical graphs embedded with spectral embedding (above), and SPE w/ kNN (below). Eigenspectra are shown to the right. SPE finds a small number of dimensions that highlight many of the symmetries of these graphs.

norm of K to avoid the objective function from growing unboundedly. We claim that this objective function attempts to recover a low-rank version of spectral embedding.

Lemma 1. *The objective function* $\max_{K\succeq 0} \mathrm{tr}(KA)$ *subject to* $\mathrm{tr}(K) \leq 1$ *recovers a low-rank version of spectral embedding.*

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

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

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:

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

Here λ is a vector containing the diagonal entries of Λ in decreasing order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and $\tilde{\lambda}$ is a vector containing the diagonal entries of $\tilde{\Lambda}$ 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

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

where we also have to satisfy $\lambda^T\mathbf{1} \leq 1$ since $\mathrm{tr}(K) \leq 1$. To maximize the objective function $\lambda^T\tilde{\lambda}$ we 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 the spectral embedding

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

Thus, the maximization problem reproduces spectral embedding while pushing all the spectrum weight into the top eigenvalue. The (rank 1) solution must be $K = vv^T$ where v is the leading eigenvector of A. If there are ties in A for its top eigenvalues, K is a conic combination of the top eigenvectors of A which is a low rank solution and has rank at most equal to the multiplicity of the top eigenvalue of A. □

In summary, this objective function will attempt to mimic the traditional spectral embedding of a graph. By combining this objective with the linear constraints from the previous section, it will be possible to also correct the embedding such that the graph structure in A is preserved and can be reconstructed from the embedding by using a connectivity algorithm $\mathcal{G}$.

4. Algorithm

In this section, the convex objective function is combined with the linear constraints implied by the connectivity algorithm $\mathcal{G}$, be it $k$nn, maximum weight b-matching or a maximum weight spanning tree. All share the same objective function and some common constraints $\mathcal{K} = \{K \succeq 0, \mathrm{tr(K)} \leq 1, \sum_{\mathrm{ij}} \mathrm{K_{ij}} = 0, \xi \geq 0\}$ to limit the trace norm of K and to center K such that the embeddings are centered on the origin. The objective function becomes $\max_{K\in\mathcal{K}} \ \mathrm{tr(KA)} - \mathrm{C}\xi$. Note that the function involves an additional term $C\xi$ which uses a manually specified parameter C. This allows some violations of the constraints on K given by the choice of algorithm $\mathcal{G}$, as shown in Figure 3. All linear inequalities encountered from structure preserving constraints are slackened with a large C weight to allow a few violations if necessary. The use of a finite C assures a solution to the SDP is always possible and helps avoid numerical problems and also encourages faster convergence of cutting plane methods as discussed in (Finley & Joachims, 2008).

For graphs created from greedy algorithms such as k-nearest neighbors, Table 1 outlines the corresponding SPE procedure.

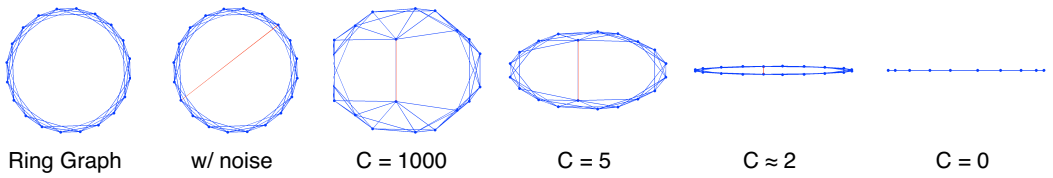

Figure 3. By adjusting the input paramter C, SPE is able to handle noisy graphs. From left to right, we see a perfect ring graph embedded by SPE, a noisy line added to the graph at random , and then the results of using SPE on the noisy graph with C roughly set to 1000, 5, 2, and 1. Note when C is small, SPE reproduces the rank-1 spectral embedding.

Table 1. Structure Preserving Embedding algorithm for k-nearest neighbor constraints.

Input	$A \in \mathbb{B}^{N\times N}$, connectivity algorithm $\mathcal{G}$, and parameter C.
Step 1	Solve SDP $\tilde{K} = \arg\max_{K\in\mathcal{K}} \mathrm{tr}(\mathrm{KA}) - \mathrm{C}\xi$ s.t. $D_{ij} > (1 - A_{ij}) \max_m (A_{im} D_{im}) - \xi$
Step 2	Apply SVD to $\tilde{K}$ and use the top eigenvectors as embedding coordinates

For maximum weight subgraph constraints, there often exists an exponential number of constraints of the form:

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

where $\triangle(\tilde{A}, A) = \frac{1}{N^2} \sum_{ij} |\tilde{A}_{ij} - A_{ij}|$, and $\tilde{A} \in \mathcal{G}$ states that $\tilde{A}$ is in the family of graphs formed by a connectivity algorithm $\mathcal{G}$, such as b-matchings or trees. To avoid enumerating the exponential number of constraints, we start by running the optimization without any structure preserving constraints and then add the most violated constraint at each iteration. Given a learned kernel $\tilde{K}$ from the previous iteration, we find the most violated constraint by computing the connectivity $\tilde{A}$ that maximizes $\mathrm{tr}(\tilde{\mathrm{W}}\tilde{\mathrm{A}})$ s.t. $\tilde{A} \in \mathcal{G}$ using a maximum weight subgraph method. We then add the constraint to our optimization $(\mathrm{tr}(\mathrm{WA}) - \mathrm{tr}(\mathrm{W}\tilde{\mathrm{A}})) \geq \triangle(\tilde{\mathrm{A}}, \mathrm{A}) - \xi$. The first iteration yields a rank-1 solution, which typically violates many constraints, but after several iterations, the algorithm converges when $|\mathrm{tr}(\tilde{\mathrm{W}}\tilde{\mathrm{A}}) - \mathrm{tr}(\tilde{\mathrm{W}}\mathrm{A})| \leq \epsilon$, where ϵ is an input parameter. Table 2 summarizes the cutting-plane version of SPE.

SPE is implemented in MATLAB as a Semidefinite Program using CSDP and SDP-LR (Burer & Monteiro, 2003) and has complexity similar to other dimensionality reduction SDPs such as Semidefinite Embedding. The complexity is $O(N^3 + \mathcal{C}^3)$ (Weinberger et al., 2005) where $\mathcal{C}$ denotes the number of constraints (for the k-nearest neighbor constraints we typically have $\mathcal{C} \propto |E|$). However, in practice many constraints are inactive and working set methods (now common practice for SDPs) perform well. Furthermore, SDP-LR directly exploits the low-rank properties of our objective function. We have run SPE on graphs with over thousands of nodes and tens of thousands of edges. For the cutting plane formulation, we add constraints iteratively, and it can be shown that the algorithm will converge in polynomial time for quadratic programs with linear constraints (Finley & Joachims, 2008). Since we have a semidefinite program, these guarantees do not carry over immediately, although in practice the cutting plane algorithm works well and has also been successfully deployed in settings beyond structured prediction and quadratic programming (Sontag & Jaakkola, 2008).

Table 2. Structure Preserving Embedding algorithm with cutting-plane constraints.

Input	$A \in \mathbb{B}^{N\times N}$, connectivity algorithm $\mathcal{G}$, and parameters C, ϵ.
Step 1	Solve SDP $\tilde{K} = \arg\max_{K\in\mathcal{K}} \mathrm{tr}(\mathrm{KA}) - \mathrm{C}\xi$.
Step 2	Use $\mathcal{G}$, $\tilde{K}$ to find biggest violator $\tilde{A} = \arg\max_A \mathrm{tr}(\tilde{\mathrm{W}}\mathrm{A})$.
Step 3	If $\lvert\mathrm{tr}(\tilde{\mathrm{W}}\tilde{\mathrm{A}}) - \mathrm{tr}(\tilde{\mathrm{W}}\mathrm{A})\rvert > \epsilon$, add constraint $\mathrm{tr}(\mathrm{WA}) - \mathrm{tr}(\mathrm{W}\tilde{\mathrm{A}}) \geq \triangle(\tilde{\mathrm{A}}, \mathrm{A}) - \xi$ and go to Step 1
Step 4	Apply SVD to $\tilde{K}$ and use the top eigenvectors as embedding coordinates

5. Experiments

We present visualization results on a variety of synthetic and real-world datasets, highlighting the improvements of SPE over purely spectral methods. Figure 2 shows a variety of classical graphs visualized by spectral embedding and SPE. Note that spectral embedding typically finds many eigenvectors with dominant eigenvalues, and thus needs many more coordinates for accurate visualization, as compared to SPE which finds compact and accurate embeddings. Figure 4 shows an embedding of two organic compounds. The true physical embedding in 3D space is shown on the left. Given only connectivity information SPE is able to produce coordinates for each atom that better resemble the true physical coordinates. Figure 5 shows a visualization of 981 political blogs (Adamic & Glance, 2005). The eigenspectrum shown next to each embedding reveals that both spectral embedding and

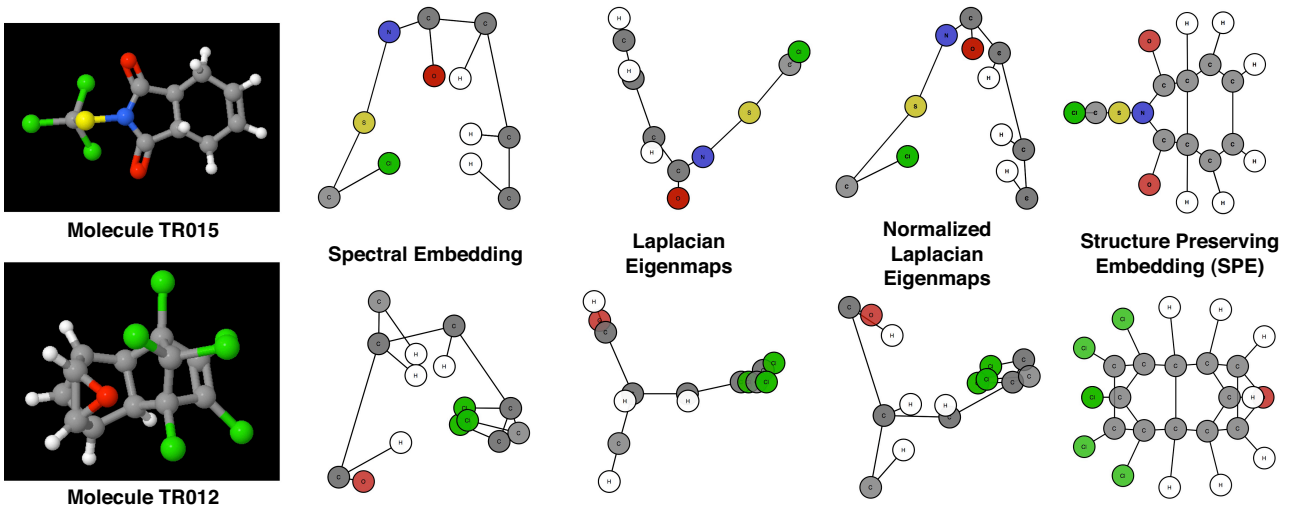

Figure 4. Two comparisons of molecule embeddings (top row and bottom). The SPE w/kNN embedding (right) more closely resembles the true physical embedding of the molecule (left), despite being given only connectivity information.

Laplacian eigenmaps find many possible dimensions for visualization, whereas SPE requires far fewer dimensions to accurately represent the structure of the data. Also, note that the embedding that uses the graph Laplacian is dominated by the degree distribution of the network. Nodes with high degree require neighbors to be very far away. The SPE embedding was created using b-matching constraints and represents the network quite compactly, finding 2 dominant dimensions which produce a beautiful visualization of the link structure as well as yield the lowest error on the adjacency matrix created from the low-dimensional embedding.

6. Dimensionality Reduction

SPE is not explicitly intended as a dimensionality reduction tool, but rather a tool for embedding graphs in few dimensions. However, these two goals are closely related; many nonlinear dimensionality reduction algorithms, such as Locally Linear Embedding (LLE), Maximum Variance Unfolding (MVU), and Minimum Volume Embedding (MVE) begin by finding a sparse connectivity matrix A that describes local pairwise distances (Roweis & Saul, 2000; Weinberger et al., 2005; Shaw & Jebara, 2007). These methods assume that the data lies on a low-dimensional nonlinear manifold embedded in a high-dimensional ambient space. To recover the manifold, these methods preserve local distances along edges specified by A in the hope that they mimic true geodesic distances measured along the manifold as opposed to measured through arbitrary directions in high-dimensional space. These pairwise distances and the connectivity matrix describe a weighted graph. However, preserving distances *does not* explicitly preserve the structure of this graph. Algorithms such as LLE, MVU, and MVE, produce embeddings whose resulting connectivity no longer matches the inherent connectivity of the data. The key problem arises from unconnected nodes. The distances between nodes that are connected are preserved; however, distances between unconnected nodes are free to vary, and thus can drastically change the graph's topology. Manifold unfolding algorithms may actually collapse parts of the manifold onto each other inadvertently and break the original connectivity structure, as illustrated in the toy problem in Figure 6. Therein, the distances are preserved yet MVU folds the graph and moves points in such a way that they are closer to other points that were originally not neighbors. By incorporating SPE's constraints, these algorithms can globally preserve the input graph exactly, not just its local properties.

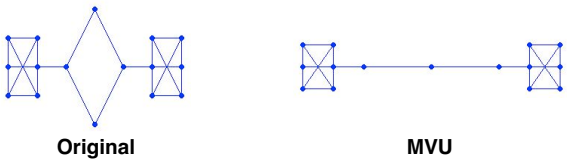

Figure 6. An example of maximum variance unfolding (MVU) folding the input graph. Because MVU does not explicitly preserve structure, MVU collapses the diamond shape in the middle to a single point. Explicit structure preserving constraints would keep the original solution.

We present an adaptation of MVU called MVU+SP, that simply adds the kNN structure preserving constraints to the MVU semidefinite program. Similarly, we present an adaptation of the MVE algorithm called MVE+SP that adds the kNN structure preserving constraints to the MVE semidefinite program (Shaw & Jebara, 2007). Exactly preserving a k-nearest neighbor graph on the data during embedding means a k-nearest

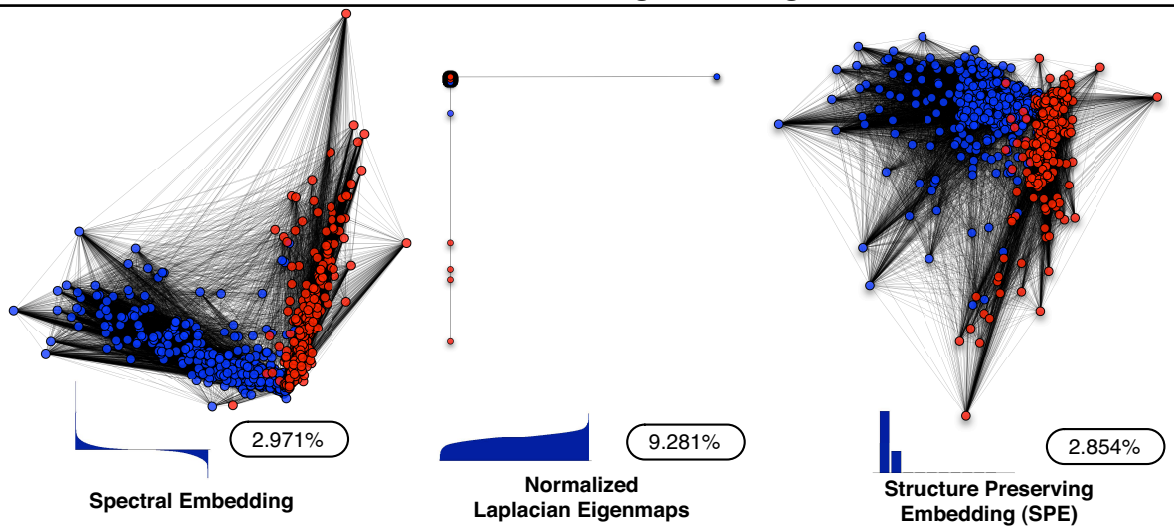

Figure 5. The link structure of 981 political blogs. Conservative are labeled red, and liberal blue. Also shown is the % error of the adjacency matrix created from the 2D embedding, and the resulting eigenspectrum from each method. Not shown here is un-normalized Laplacian eigenmaps, whose embedding is similar to the normalized case and reconstruction error is 55.775%. Note that SPE is able to achieve the lowest error rate, representing the data using only 2 dimensions.

neighbor classifier will perform equally well on the recovered low-dimensional embedding K as it did on the original high-dimensional dataset. Because structure is not explicitly preserved with many existing dimensionality reduction techniques, the neighborhood of each point can drastically change, thus potentially reducing the accuracy of the $k$nn classifier. These errors indicate that the manifold has been folded onto itself during dimensionality reduction. Table 3 shows that MVE+SP offers a significant advantage over other methods in terms of accuracy of using a 1-nearest neighbor classifier on the resulting 2D embeddings.

The results in Table 3 were obtained by sampling a variety of small graphs from UCI datasets (Asuncion & Newman, 2007) by randomly selecting 120 nodes from the two largest classes of each dataset. Then, the data was embedded with the the variety of algorithms above and the resulting performance of a 1NN classifier was measured. The data was split 60/20/20 into training/cross validation/testing groups. The accuracies shown in Table 3 show the results of averaging 50 of these folds. Cross-validation was used to find the optimal value of k for each algorithm and dataset, where k specifies the number of nearest neighbors each node connects to. The All-Dimension column on the right shows the corresponding accuracy of a 1-nearest-neighor classifier using all of the features of the data. MVE+SP performs better than all other low-dimensional methods, and for the datasets Ionosphere, Ecoli, and OptDigits, MVE+SP is more accurate than using all the features of the data even though MVE+SP uses only 2 features per datapoint. It appears that the manifold recovered by MVE+SP denoises the data, placing the key modes of variation in the top few dimensions, while disregarding other noise that reduces accuracy for the all-dimensions case.

Clearly MVE+SP is not intended to compete with supervised algorithms for classification since it operates agnostically without any knowledge of the labels for the points. However, Table 3 highlights a common deficiency with many dimensionality reduction algorithms. Often, if dimensionality reduction is used in an unsupervised setting as a preprocessing step, it can reduce performance and a classifier formed from the low-dimensional data can perform worse. Simply preserving distances during dimensionality reduction is not enough to preserve accuracy for a subsequent classification problem. This is not the case for MVE+SP which is maintaining (or even helping) classification rates, further reinforcing the strength of the structure preserving constraints to create accurate low-dimensional representations of data (without any knowledge of labels or a classification task). Furthermore, these results confirm a recent finding regarding the finite-sample risk for a k-nearest neighbor classifer (Snapp & Venkatesh, 1998) which show that the finite-sample risk can be expressed asymptotically as: $R_m = R_\infty + \sum_{j=2}^{k} c_j n^{-j/d} + O(n^{-(k+1)/d})$, $n \to \infty$, where n is the number of finite samples, d is the dimensionality of the data, k is the number of neighbors in the nearest neighbor classifier and $c_2, ..., c_k$ are scalar coefficients specified in (Snapp & Venkatesh, 1998). The main theorem of (Snapp & Venkatesh, 1998) suggests that this expansion validates the *curse of dimensionality* and, in order, to maintain $|R_n - R_\infty| < \epsilon$ for some $\epsilon > 0$ the sample size must scale with ϵ according to $n \propto \epsilon^{-d/2}$. Thus, if the k-nearest neighbor structure of the data is preserved exactly while dimension-

Table 3. Average classification accuracy of a 1-nearest neighbor classifier on UCI datasets. 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.

	KPCA	MVU	MVE	MVE+SP	All-Dimensions
Ionosphere	66.0%	85.0%	81.2%	**87.1**%	78.8%
Cars	66.1%	70.1%	71.6%	**78.1**%	79.3%
Dermatology	58.8%	63.6%	64.8%	**66.3**%	76.3%
Ecoli	94.9%	95.6%	94.8%	**96.0**%	95.6%
Wine	68.0%	68.5%	68.3%	**69.7**%	71.5%
OptDigits 4 vs. 9	94.4%	99.2%	99.6%	**99.8**%	98.6%

ality d is reduced (for instance using the approach of MVE+SP), the finite-sample risk should more quickly approach the infinite sample risk. This makes it is possible to more accurately use training performance on low-dimensional k-nearest neighbor graphs to estimate test performance.

7. Conclusion

This article suggests that preserving local distances or using spectral methods is insufficient for faithfully embedding graphs in low-dimensional space, and demonstrates the improvements of SPE over current graph embedding algorithms in terms of both the quality of the resulting visualizations as well as the amount of information compression. SPE allows us to accurately visualize many interesting network structures ranging from classical graphs, to organic compounds, to the link structure between websites, using relatively few dimensions. Furthermore, by incorporating structure preserving constraints into existing nonlinear dimensionality reduction algorithms, these methods can explicitly preserve graph topology *in addition* to local distances, and produce more accurate low-dimensional embeddings.

References

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

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

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Battista, G. D., Eades, P., Tamassia, R., & Tollis, I. (1999). *Graph drawing: algorithms for the visualization of graphs.* Prentice Hall.

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

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

Chung, F. R. K. (1997). *Spectral graph theory.* American Mathematical Society.

Cox, T., & M.Cox (1994). *Multidimensional scaling.* Chapman & Hall.

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

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

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

Shaw, B., & Jebara, T. (2007). Minimum volume embedding. *Proc. of the 11^{th} International Conference on Artificial Intelligence and Statistics* (pp. 460–467).

Snapp, R., & Venkatesh, S. (1998). Asymptotic expansions of the k nearest neighbor risk. *The Annals of Statistics*, *26*, 850–878.

Sontag, D., & Jaakkola, T. (2008). New outer bounds on the marginal polytope. *Advances in Neural Information Processing Systems 20* (pp. 1393–1400).

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

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