Manifold Spanning Graphs

CJ Carey and **Sridhar Mahadevan**
School of Computer Science
University of Massachusetts, Amherst
Amherst, Massachusetts, 01003
{ccarey,mahadeva}@cs.umass.edu

Abstract

Graph construction is the essential first step for nearly all manifold learning algorithms. While many applications assume that a simple k-nearest or ϵ-close neighbors graph will accurately model the topology of the underlying manifold, these methods often require expert tuning and may not produce high quality graphs. In this paper, the hyperparameter sensitivity of existing graph construction methods is demonstrated. We then present a new algorithm for unsupervised graph construction, based on minimal assumptions about the input data and its manifold structure.

Introduction

Manifold learning is a well-established set of techniques for understanding and manipulating the high dimensional data inherent in state-of-the-art machine learning applications. A variety of popular algorithms exist for manifold-based dimension reduction (Belkin and Niyogi 2001; Coifman and Lafon 2006; Roweis and Saul 2000; Tenenbaum, Silva, and Langford 2000; Zhang and Zha 2002), alignment (Ham, Lee, and Saul 2005; Wang and Mahadevan 2009), and clustering (Shi and Malik 2000), and each of these algorithms shares a common first step: the construction of a discrete approximation of the manifold in the form of a graph.

This first step is critically important for the performance of the algorithm as a whole, yet the area of manifold graph construction has received a relative lack of attention from the manifold learning community. Many papers relegate the task to the default k-nearest or ϵ-close neighborhood-based construction algorithms, even though these have been shown to be sensitive to noise and scaling issues (Balasubramanian and Schwartz 2002). Other approaches have drawn inspiration from neural networks (Kohonen 1990; Bishop, Svensén, and Williams 1998), but often fail to represent complex manifold structures accurately due to their restrictive global models (Tenenbaum 1998).

Several algorithms have attempted to side-step the graph construction problem altogether, by explicitly representing the manifold as a combination of piecewise linear components (Kambhatla and Leen 1997; Tipping and Bishop 1999; Hinton, Dayan, and Revow 1997). These approaches only provide coarse approximations, however, and require specialized algorithms for the varied downstream applications that graph-based representations enable naturally, like out-of-sample extension and cross-manifold alignment.

More recently, attempts have been made to blend the process of graph construction with dimensionality reduction (Yang and Chen 2010; Zhang, Chen, and Qiao 2012). These methods aim to build a graph and compute an embedding simultaneously, which offers potential benefits to graph quality at the cost of limiting the generality of the algorithm. In this same fashion, many graph construction algorithms have been proposed to exploit additional information in specific problem domains (Johns and Mahadevan 2007; Rohban and Rabiee 2012).

In this paper, we introduce a novel algorithm for construction of manifold topology preserving graphs, based on the parameter-free forest-joining framework of Kruskal's minimum spanning tree algorithm (Kruskal 1956) and the structural information conveyed by locally linear subspace fitting (Hinton, Dayan, and Revow 1997; Kambhatla and Leen 1997; Tipping and Bishop 1999; Zhang and Zha 2002). This combined approach contructs graphs that respect both inter-point distances and edge-to-manifold angles, without requiring the use of any expert-tuned hyperparameters.

Manifolds and Graph Construction

It is commonly held that while most information we encounter in the real world is high dimensional and noisy, the true structure of the data lies on a lower-dimensional manifold. These globally nonlinear manifolds are assumed to be locally linear given neighborhoods of a specific, and typically unknown, size. Understanding the manifold structure in data is an important tool when combating the curse of dimensionality, as this local linearity property enables manifold embedding to spaces with simple distance metrics and compact feature representations (Tenenbaum 1998).

Manifold Representations

In order to represent a manifold computationally, it is often useful to create a discrete approximation in the form of a graph (Martinetz and Schulten 1994). Vertices of the graph are sampled from the input data (and thus lie on the manifold), and edges are chosen such that no point along the

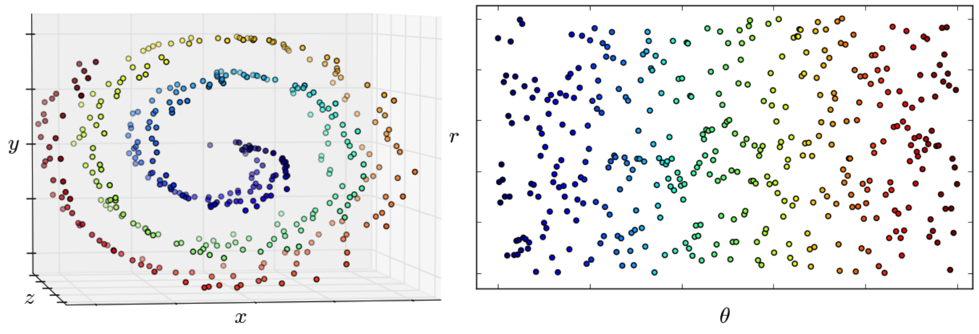


Figure 1: θ-parametrized Swiss Roll, in three-dimensional input space (left) and r-θ manifold space (right). In both plots, points are colored by their θ value.

edge lies outside the manifold. The graph construction problem can be framed as follows: given a set of points in an arbitrarily high-dimensional space, produce a set of edges that satisfy the "along-manifold" constraint. Furthermore, algorithms should produce enough edges that each manifold present in the input data can be represented by exactly one connected component.

Some methods of manifold representation eschew the graph approximation, however, in favor of computing a piecewise-linear representation. This family of methods includes charting (Brand 2002), as well as several local PCA algorithms (Kambhatla and Leen 1997; Tipping and Bishop 1999). This latter group takes a data-driven approach by fitting linear subspaces that lie tangent to the manifold surface. This concept has proven useful in the Local Tangent Space Alignment embedding algorithm (Zhang and Zha 2002), which, given an existing graph representation, computes subspaces on local neighborhoods to recover embeddings with less distortion.

Graph Construction Algorithms

The most commonly used algorithm for graph construction is k-nearest neighbors, in which distances between all pairs of points are computed, and edges are added between a point, p, and the k points with the smallest distance from p. While computationally efficient and simple, this algorithm does not necessarily create a symmetric graph, as some of p's k nearest neighbors may not include p among their k nearest neighbors. Symmetry is an important property for downstream algorithms such as Spectral Embedding (Belkin and Niyogi 2001), so a k-nearest adjacency matrix W is often symmetrized as $W_{sym} = \frac{W+W^\top}{2}$.

The other canonical graph construction method is the ϵ-close algorithm: edges are added between each point p and all other points with distance less than a threshold ϵ. While this algorithm always constructs a symmetric graph, it also tends to over-connect, especially when the input data is not uniformly distributed or poorly scaled. There is also no guarantee on the maximum degree of vertices in the generated graph.

Others have proposed alternative graph construction algorithms, such as b-matching (Jebara, Wang, and Chang 2009). Given an affinity matrix, this algorithm produces a balanced graph that minimizes the sum of edge weights, under the constraint that each vertex has degree b.

The Manifold Spanning Graph Algorithm

This paper presents a novel, bottom-up algorithm for graph construction. Inputs are X, an $N \times D$ matrix of D-dimension points, d, an estimate of the intrinsic dimension of the underlying manifold, and m, the desired number of connected components in the final graph. The high-level procedure can be divided into three stages:

1. Divide X into many small connected components of size at least $d+1$, then compute a d-dimension linear subspace for each component via PCA.
2. Add edges between components that minimize both Euclidean distance and the maximum angle created by the new edge and the two subspaces it joins. Continue adding edges until m connected components remain.
3. Add more edges to fill out gaps in the graph structure, using the learned subspaces from step 1 as well as distance and angle thresholds learned in step 2.

The following sections examine each stage in detail.

Connectivity Forest and Subspace Construction

For a manifold with intrinsic dimension d, a locally linear patch can be accurately described with at least $d+1$ points, assuming general position. Therefore, the first task is to partition the input data into groups of at least $d+1$ points, composing many small, locally linear neighborhoods: a "connectivity forest". Note that this does not imply that each vertex in the graph should have degree d, as each neighborhood need not form a complete graph.

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

Once the first-nearest neighbors are connected, we can compute the set of strongly connected components $C^{(0)}$

(Tarjan 1972). For any components $C_i \in C^{(0)}$ where $|C_i| < d+1$, additional edges are added by merging C_i with C_j such that:

$$C_j = \arg\min_{j \neq i} \min_{p \in C_i, q \in C_j} \|p - q\|_2 \tag{1}$$

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

When all components in $C^{(0)}$ satisfy the cardinality condition in Equation 1, we compute the set of linear subspaces:

$$P = \left\{ \text{PCA}_d \left(C_i \right) \mid C_i \in C^{(0)} \right\} \tag{2}$$

Each P_i is represented by the first d principal components (Jolliffe 2002) of the vertices in C_i. This procedure at this stage is reminiscent of the Local PCA algorithm (Kambhatla and Leen 1997), albeit with a different selection criterion for the sets of points C_i. The primary difference, however, is that the subspaces P are not the final representation of the manifold, but only an intermediate set of constraints on edge connections. In this way, P may be viewed as an implicit set of charts (Brand 2002; Lin and Zha 2008), with the inter-chart connections yet undefined.

Component Joining

Given a desired number of connected components m, we continue to add edges until $\left|C^{(t)}\right| = m$. For this task, we consider only edges between "adjacent" connected components, that is, (C_i, C_j) pairs satisfying Equation1. In addition to this requirement, a candidate edge $(p \in C_i, q \in C_j)$ must satisfy:

$$\|\bar{e}\|_2 \leq \epsilon_{dist} \tag{3}$$

$$\max\left(\|P_i - \bar{e}\|_F, \|P_j - \bar{e}\|_F \right) \leq \epsilon_{angle} \tag{4}$$

where $\bar{e} \equiv p - q$. In Equation 4, $\|\cdot\|_F$ refers to the projection F-norm (Gruber and Theis 2006), in this case measuring the angle between the edge vector and each adjoining subspace.

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

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

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

Note that as edges are added and the cardinality of the remaining components in $C^{(t)}$ increase, each C_i is likely to lose its linearity property. For this reason, we refrain from recomputing PCA subspaces as in Equation 2, but instead maintain a mapping for elements from each $C^{(t)}$ to their original components in $C^{(0)}$, and thus P.

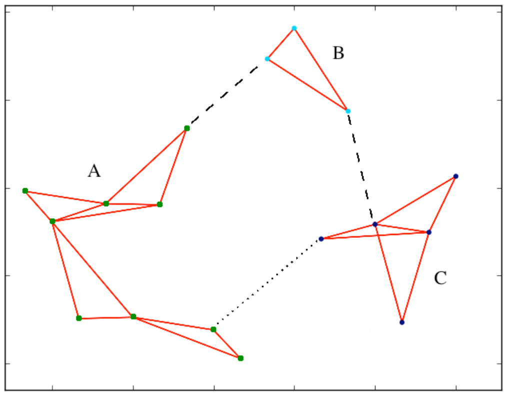


Figure 2: Limitation of only connecting adjacent components. Components A and C should connect along the dotted edge, but no edges will be added between them after they merge via the addition of the dashed A-B and B-C edges.

Edge Addition Post-processing

One limitation of the proposed algorithm is that only edges between "adjacent" connected components are added, which may result in a lack of edges that convey important structural information, even after the desired number of connected components is reached. Figure 2 illustrates this issue.

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

$$\left|\text{ShortestPath}\left(p, q\right)\right| \geq h \tag{5}$$

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

Algorithm Analysis

If no estimate for d is available, the algorithm can infer a reasonable d from the explained variance of each PCA subspace, à la Isomap's "elbow" heuristic. Alternatively, the intrinsic dimension of the underlying manifold can be estimated using any of several existing techniques (Kégl 2002; Levina and Bickel 2004).

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

1. At least one edge satisfies both constraints 3 and 4. All candidate edges connect disjoint connected components, so when the edge is added it merges at least two components and decreases $\left|C^{(t+1)}\right|$ by at least one.
2. No edge satisfies constraint 3, but at least one edge satisfies constraint 4. ϵ_{dist} is increased to the length of the

minimum-angle edge satisfying constraint4, and the first condition now applies.

3. No edge satisfies constraint 4, but at least one edge satisfies constraint 3. ϵ_{angle} is increased to the angle of the minimum-length edge satisfying constraint 3, and the first condition now applies.
4. No edge satisfies either constraint 3 or 4. ϵ_{dist} is increased to the minimum length of all candidate edges, ϵ_{angle} is increased to the chosen edge's angle, and the first condition now applies.

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

Experimental Results

The preceding algorithm was implemented in Python using open-source libraries (Oliphant 2007; Pedregosa et al. 2011). Source code will be made available publicly on the author's website following publication.

Parametric Swiss Roll

We consider the classic "swiss roll" dataset, shown in Figure 1. Points on the roll are randomly distributed along the z-axis, and their other coordinates are parameterized by r and θ, which control the distance from and angle with the origin, respectively:

$$\begin{aligned} x &= r \sin\theta \qquad (6)\\ y &= r\cos\theta \\ z &= \text{Uniform}\,(0,1) \end{aligned}$$

Under this parametrization, it is trivial to recover the optimal embedding by plotting the points in (θ, z) space. We can also define an objective measure of graph quality, marking an edge as incorrect if it connects points p and q, and $|\theta_p - \theta_q| > \epsilon$. For the following experiments, we used the threshold $\epsilon = 0.1$. One measure of overall graph quality is thus the "edge error ratio", defined as the ratio of incorrect edges to total edges in the graph.

We first calculate this error ratio for the standard k-nearest and ϵ-close graphs, over the set of reasonable hyperparameters. The results in Figure 3 show that for our particular data set, there is no value of k which produces a graph with a low error ratio that also forms a single connected component. This replicates previous results (Balasubramanian and Schwartz 2002) which demonstrated that simple k-nearest neighbor algorithms are sensitive to noise. While there are some values of ϵ that avoid introducing erroneous edges while forming one connected component, these tend to overconnect the graph (see Figure 3), and begin to fail dramatically once a certain ideal threshold has been met.

By contrast, the Manifold Spanning Graph algorithm presented in this paper produces a connected graph with minimal incorrect edges, without requiring any hyperparameter tuning. Figure 4 illustrates the effect of incorrect edges on an Isomap embedding of the swiss roll data.

To evaluate the average performance of each algorithm, the number of incorrect edges and total edges were calculated over 200 randomly-generated swiss rolls (following

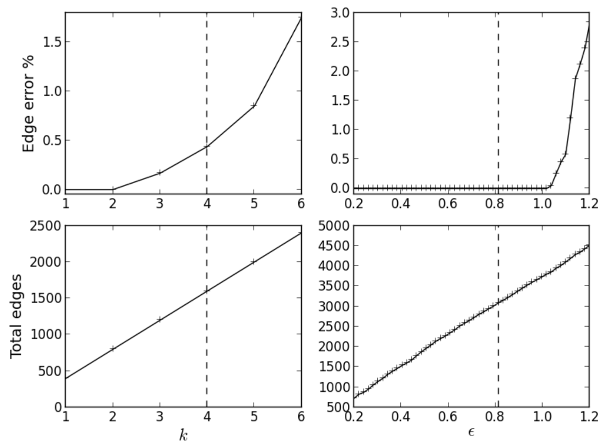


Figure 3: Sensitivity of k-nearest (left) and ϵ-close (right) algorithms. The dashed vertical lines represent the first hyperparameter value producing one connected component.

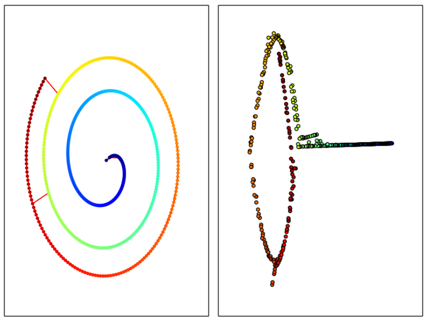
Equation 6). Hyperparameters k, ϵ, and b were optimized to produce one connected component for the first swiss roll, then the same parameters were used for all future examples. No hyperparameter tuning is required for the Manifold Spanning Graph algorithm. Figure 5 demonstrates that the MSG algorithm consistently produces near-zero incorrect edges, while still generating a reasonable number of total edges.

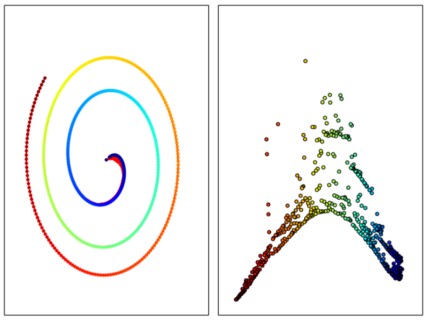
MNIST digit clustering

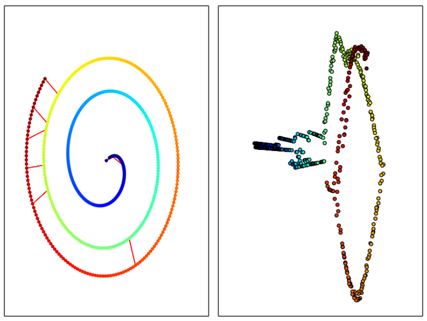
Real world data often makes evaluating graph quality difficult, as the optimal low-dimensional embedding is typically unknown. However, the MNIST dataset of $10,000$ handwritten digits (LeCun and Cortes 1998) allows for a simple "edge error" metric: the ratio of between-class to within-class edges. Each 28×28 grayscale image is represented as a 784-dimensional vector of pixel values. This test demonstrates the Manifold Spanning Graph algorithm's ability to scale up to high-dimensional data, as well as exercising the ability to specify m, the number of desired connected components.

Figure 6 demonstrates the applicability of the MSG algorithm on high-dimensional data sets, again revealing a lower error rate without hyperparameter tuning. The k and b values were tuned by choosing the first k and b such that the number of connected components in the resulting graph was ≤ 10, the number of classes in the corpus. The same experiment with the ϵ-close algorithm produced too many incorrect edges for any reasonable number of desired connected components, so we omit further results here.

As a final evaluation of the created graphs from Figure 6, a simple digit classification task was performed. 20 unique images were selected at random from the $10,000$-image corpus to act as labeled examples, while ensuring that each of the 10 digits had at least one example image. For each of the k-nearest, b-matching, and Manifold Spanning graphs, a simple label propagation algorithm (Zhu and Ghahramani 2002) was used to classify the remaining $9,980$ images. The


(a) k-nearest


(b) ϵ-close


(c) b-matching

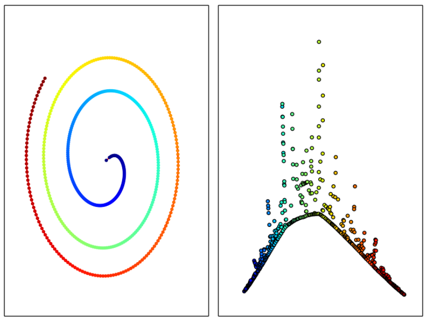
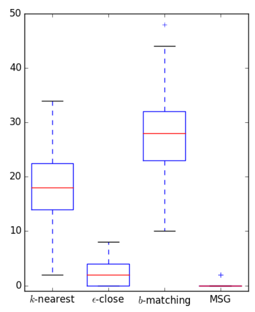
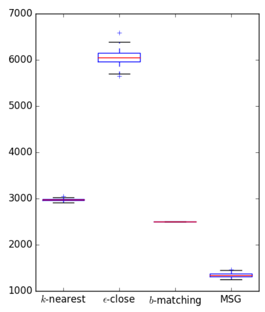

(d) Manifold Spanning Graph

Figure 4: Side on view of Swiss Roll graphs (left) and the resulting Isomap embeddings (right). For the algorithms that require hyperparameter tuning, the parameters k, ϵ, and b were set by choosing the smallest value that produced a single connected component.

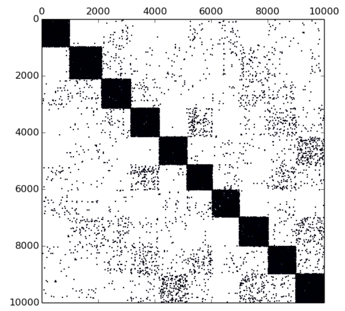


(a) Incorrect edges

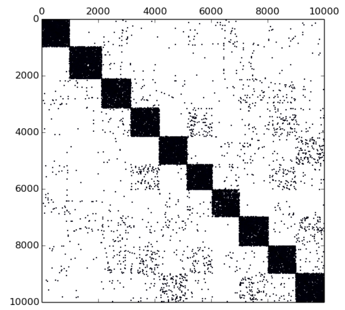


(b) Total edges

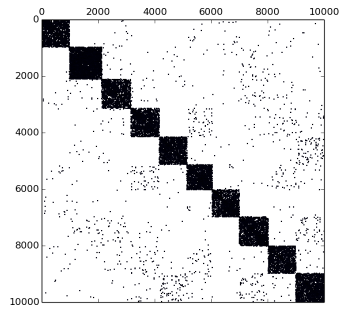
Figure 5: Summarized performance over 200 random swiss rolls, each with 500 points.



(a) k-nearest neighbor edges

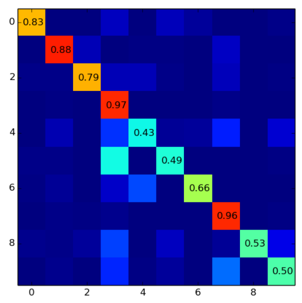


(b) b-matching edges

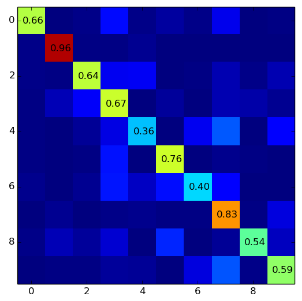


(c) MSG edges

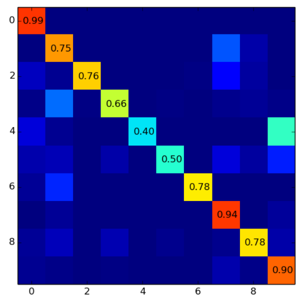
Figure 6: MNIST experiment results. k-nearest with $k = 3$ had $3256/45426 \approx 7.2\%$ bad edges. b-matching with $b = 3$ had $1989/30000 \approx 6.6\%$ bad edges. MSG with $m = 2$ and $d = 2$ had $1118/23042 \approx 4.9\%$ bad edges.



(a) k-nearest



(b) b-matching



(c) MSG

Figure 7: Confusion matrices from the MNIST digit classification task. Out of $10,000$ images, k-nearest, b-matching, and MSG correctly classified 7091, 6463, and 7489, respectively.

results (Figure 7) demonstrate once again that the MSG algorithm produces high-quality graphs without the use of parameter tuning.

Discussion and Future Work

Manifold learning encompasses a powerful set of techniques which typically rely on the construction of a graph that accurately represents the input data's underlying manifold structure. This graph-based approximation is of critical importance, as even a single incorrect edge could potentially distort the dimensionality and structure of the discretized manifold, rendering many downstream computations useless. This problem is especially pernicious in high-dimensional, real-world data, whose structure is unknown a priori.

This paper presents a novel algorithm for graph construction based on a bottom-up approach à la Kruskal's minimum spanning tree algorithm. The proposed algorithm demonstrates improved accuracy over traditional methods without requiring any expert-tuned hyperparameters. The learned graphs approximate the true manifold structure of the data, relying only on the smoothness and local linearity properties of Riemannian manifolds.

While the assumption that points and their first nearest neighbor are adjacent in manifold space is reasonable for completely unstructured data, many corpuses carry additional signal in the form of spatial and temporal correlations. Future work will explore specialized algorithms for producing connectivity forests from these semi-structured inputs, which may then be joined and filled via the standard Manifold Spanning Graph procedure.

References

Balasubramanian, M., and Schwartz, E. L. 2002. The isomap algorithm and topological stability. *Science* 295(5552):7–7.

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

Bishop, C. M.; Svensén, M.; and Williams, C. K. 1998. Gtm: The generative topographic mapping. *Neural computation* 10(1):215–234.

Brand, M. 2002. Charting a manifold. In *Advances in neural information processing systems*, 961–968.

Coifman, R. R., and Lafon, S. 2006. Diffusion maps. *Appl. Comput. Harmon. Anal* 21:5–30.

Gruber, P., and Theis, F. J. 2006. Grassmann clustering. *Proc. EUSIPCO 2006*.

Ham, J.; Lee, D.; and Saul, L. 2005. Semisupervised alignment of manifolds. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence, Z. Ghahramani and R. Cowell, Eds*, volume 10, 120–127.

Hinton, G. E.; Dayan, P.; and Revow, M. 1997. Modeling the manifolds of images of handwritten digits. *Neural Networks, IEEE Transactions on* 8(1):65–74.

Jebara, T.; Wang, J.; and Chang, S.-F. 2009. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 441–448. ACM.

Johns, J., and Mahadevan, S. 2007. Constructing basis functions from directed graphs for value function approximation. In *Proceedings of the 24th international conference on Machine learning*, 385–392. ACM.

Jolliffe, I. 2002. *Principal component analysis*. Springer series in statistics. Springer-Verlag.

Kambhatla, N., and Leen, T. K. 1997. Dimension reduction by local principal component analysis. *Neural Computation* 9(7):1493–1516.

Kégl, B. 2002. Intrinsic dimension estimation using packing numbers. In *Advances in neural information processing systems*, 681–688.

Kohonen, T. 1990. The self-organizing map. *Proceedings of the IEEE* 78(9):1464–1480.

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

LeCun, Y., and Cortes, C. 1998. The mnist database of handwritten digits.

Levina, E., and Bickel, P. J. 2004. Maximum likelihood estimation of intrinsic dimension. In *Advances in neural information processing systems*, 777–784.

Lin, T., and Zha, H. 2008. Riemannian manifold learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30(5):796–809.

Martinetz, T., and Schulten, K. 1994. Topology representing networks. *Neural Networks* 7(3):507–522.

Oliphant, T. E. 2007. Python for scientific computing. *Computing in Science & Engineering* 9(3):10–20.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research* 12:2825–2830.

Rohban, M. H., and Rabiee, H. R. 2012. Supervised neighborhood graph construction for semi-supervised classification. *Pattern Recognition* 45(4):1363–1372.

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

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22(8):888–905.

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

Tenenbaum, J. B.; Silva, V. D.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science*.

Tenenbaum, J. B. 1998. Mapping a manifold of perceptual observations. *Advances in neural information processing systems* 682–688.

Tipping, M. E., and Bishop, C. M. 1999. Mixtures of probabilistic principal component analyzers. *Neural computation* 11(2):443–482.

Wang, C., and Mahadevan, S. 2009. A general framework for manifold alignment. In *AAAI Fall Symposium on Manifold Learning and its Applications*, 53–58.

Yang, B., and Chen, S. 2010. Sample-dependent graph construction with application to dimensionality reduction. *Neurocomputing* 74(1):301–314.

Zhang, Z., and Zha, H. 2002. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *arXiv preprint cs/0212008*.

Zhang, L.; Chen, S.; and Qiao, L. 2012. Graph optimization for dimensionality reduction with sparsity constraints. *Pattern Recognition* 45(3):1205–1210.

Zhu, X., and Ghahramani, Z. 2002. Learning from labeled and unlabeled data with label propagation.