GRASPEL: GRAPH SPECTRAL LEARNING AT SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning meaningful graphs from data plays important roles in many data mining and machine learning tasks, such as data representation and analysis, dimension reduction, data clustering, and visualization, etc. In this work, we present a scalable spectral approach to graph learning from data. By limiting the precision matrix to be a graph Laplacian, our approach aims to estimate ultra-sparse weighted graphs and has a clear connection with the prior graphical Lasso method. By interleaving nearly-linear time spectral graph sparsification, coarsening and embedding procedures, ultra-sparse yet spectrally-stable graphs can be iteratively constructed in a highly-scalable manner. Compared with prior graph learning approaches that do not scale to large problems, our approach is highly-scalable for constructing graphs that can immediately lead to substantially improved computing efficiency and solution quality for a variety of data mining and machine learning applications, such as spectral clustering (SC), and t-Distributed Stochastic Neighbor Embedding (t-SNE).

1 INTRODUCTION

Graph construction is playing increasingly important roles in many machine learning and data mining applications. For example, a key step of many existing machine learning methods requires converting potentially high-dimensional data sets into graph representations: it is a common practice to represent each (high-dimensional) data point as a node, and assign each edge a weight to encode the similarity between the two nodes (data points). The constructed graphs can be efficiently leveraged to represent the underlying structure of a data set or the relationship between data points (Jebara et al., 2009; Maier et al., 2009; Liu et al., 2018). However, how to learn meaningful graphs from large data set at scale still remains a challenging problem.

In the past decades, considerable effort has been devoted to the development of graph construction methods. For example, constructing k-nearest-neighbor (kNN) graphs requires each node to be connected with its top-k nearest neighbors, while in construction of the ϵ-neighborhood graphs all the neighbors within the range of distance ϵ will be connected; to improve the capability of kNN graph in handling multi-scale data, (Zelnik-Manor & Perona, 2005) introduced a self-tuning technique to adjust the local scaling parameter for similarity measurement; to find meaningful similarity measures between nodes, (Bach & Jordan, 2006) propose to learn the similarities from feature vectors in a supervised setting; (Zhu et al., 2014) adopted an information-theoretic definition of data similarity to capture subtle similarity information; (Jebara & Shchogolev, 2006) proposed to remove spurious edges from kNN graph via b-matching; (Pavan & Pelillo, 2007) introduced a method for removing noisy edges by selecting the maximum cliques; (Premachandran & Kakarala, 2013) proposed to leverage collected consensus information form various neighborhoods to improve the robustness of the kNN graph; (Nie et al., 2014) proposed to learn the adjacency graph by adaptively assigning neighbors. However, the aforementioned nearest-neighbor (NN) based graph construction methods can only capture local manifold information and may not be able to truthfully reveal the global structure of a given data set (Nie et al., 2016; Liu et al., 2018; Guo, 2015), which can result in over complicated (with too many edges) or sometimes misleading graph representations. For example, choosing different numbers of nearest neighbors for constructing kNN graphs may lead to drastically different classification performance in spectral clustering tasks (Chen et al., 2018).

Several recent graph learning methods based on Laplacian estimation are based on emerging graph signal processing (GSP) and have shown very promising results (Dong et al., 2016; Egilmez et al., 2017). For example, (Egilmez et al., 2017) proposed to address the graph learning problem by maximizing a posterior estimation of Gaussian Markov Random Field (GMRF) and restricting the precision matrix to be a graph Laplacian, while an $l1$-regularization term is used to promote graph sparsity; (Rabbat, 2017) provides an error analysis for inferring sparse graphs from smooth signals.

However, even the state-of-the-art Laplacian estimation methods for graph learning do not scale well for large data set due to their extremely high algorithm complexity. For example, solving the optimization problem for Laplacian estimation in (Dong et al., 2015; 2016; Kalofolias, 2016; Egilmez et al., 2017; Dong et al., 2018) requires $O(N^2)$ time complexity per iteration for N data entities and nontrivial parameters tuning for controlling graph sparsity which limits their applications to only very small data sets (e. g. with up to a few thousands of data points).

In this work, we introduce a spectral method for learning ultra-sparse yet spectrally-robust graphs at scale. There is a clear connection between our approach and the GSP-based Laplacian estimation methods (Dong et al., 2015; 2016; Kalofolias, 2016; Egilmez et al., 2017; Dong et al., 2018), as well as the classical graphical Lasso framework (Friedman et al., 2008). Specifically, by treating p-dimensional data points as p graph signals, our approach aims to learn a graph Laplacian by maximizing its first few eigenvalues as well as the smoothness of graph signals across edges, subject to a graph sparsity constraint. By iteratively interleaving recent scalable spectral graph sparsification, coarsening and embedding procedures (Feng, 2016; 2017; Zhao et al., 2018), our graph learning approach enjoys a nearly-linear runtime and space complexity. We show through extensive experiments that our approach can learn high-quality graphs without sacrificing the sparsity: the learned graphs can be immediately leveraged to significantly improve the efficiency and accuracy of spectral clustering tasks; we also show how to use such graphs to significantly improve the runtime of the well-known t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithms (Maaten & Hinton, 2008; Van Der Maaten, 2014).

2 Background of Graph learning via Laplacian estimation

Given M observations on N data entities stored in a data matrix $X \in \mathbb{R}^{N \times M}$, each column of X can be considered as a signal on a graph. The recent graph learning method (Dong et al., 2016) aims to estimate a graph Laplacian from X while achieving following desired characteristics:

1) Smoothness of signals on the graph. The graph signals corresponding to the real-world data should be sufficiently smooth on the learned graph structure: the signal values will only change gradually across connected neighboring nodes. The smoothness of a signal x over a graph G can be measured by the following Laplacian quadratic form

$$x^T L_G x = \sum_{(p,q) \in E} w_{p,q} (x\,(p) - x\,(q))^2, \tag{1}$$

where $L_G = D_G - W_G$ denotes the Laplacian matrix of graph G with D_G and W_G denoting the degree and the weighted adjacency matrices of G. $w_{p,q}$ denotes the weight for edge (p, q), while $x(p)$ and $x(q)$ denote the graph signal values on nodes p and q. The smaller value of quadratic form indicates the smoother signals across the graph. It is also possible to quantify the smoothness (Q) of a set of signals X over graph G using the following matrix trace (Kalofolias, 2016):

$$Q(X, L_G) = Tr(X^T L_G X), \tag{2}$$

where Tr denotes the matrix trace. Defining $Z \in \mathbb{R}^{N \times N}$ to be the pairwise distance matrix for the N data entities, where each entry encodes the $l2$ distance between data vectors p and q:

$$Z(p, q) = z_{p,q}^{data} = \|x_p - x_q\|_2^2. \tag{3}$$

We can rewrite the trace as follows:

$$Tr(X^T L_G X) = \frac{1}{2} Tr(W_G Z) = \frac{1}{2} \|W_G \circ Z\|_{1,1}, \tag{4}$$

where $\|\|_{1,1}$ denotes the element-wise matrix norm, and $\circ$ is the Hadamard product (Kalofolias, 2016).

2) Sparsity of the estimated graph (Laplacian). Graph sparsity is another critical consideration in graph learning. One of the most important motivations of learning a graph is to use it for downstream data mining or machine learning tasks. Therefore, desired graph learning algorithms should allow better capturing and understanding the global structure (manifold) of the data set, while producing sufficiently sparse graphs that can be easily stored and efficiently manipulated in the downstream algorithms, such as graph clustering, partitioning, dimension reduction, data visualization, etc. To this end, the graphical Lasso algorithm (Friedman et al., 2008) has been proposed to learn the structure in an undirected Gaussian graphical model using $l1$ regularization to control the sparsity of the precision

matrix. Given a sample covariance matrix S and a regularization parameter R, graphical Lasso maximizes the following objective function:

$$\log\det(\Theta) - Tr(\Theta S) - R\|\Theta\|_1, \tag{5}$$

over all non-negative definite precision matrices Θ. The first two terms together can be interpreted as the log-likelihood under a GMRF. $R\|\Theta\|_1$ is the sparsity promoting regularization term. This model tries to learn the graph structure by maximizing the penalized log-likelihood. However, the log-determinant problems are very computationally expensive.

The emerging GSP-based methods infer the graph by adopting the criterion of signal smoothness. Recent methods enforce the sparsity and smoothness of signals by formulating the problem as the following optimization problem (Kalofolias, 2016; Dong et al., 2015; 2016; Egilmez et al., 2017):

$$\min_{W\in\mathcal{W}} \ \|W \circ Z\|_{1,1} + S(W), \tag{6}$$

where $\mathcal{W}$ denotes the set of valid adjacency matrices. It is obvious that the first term enforces the smoothness of the observed signals on the learned graph, while the second term is for imposing sparsity of the inferred graph. In recent years, different forms of $S(W)$ have been proposed. For example, the following form has been introduced in (Dong et al., 2016)

$$S(W) = c\|W \cdot \mathbf{1}\|^2 + c\|W\|_F^2, \tag{7}$$

whereas (Kalofolias, 2016) proposed the form as follows

$$S(W) = c\mathbf{1}^T log(W \cdot \mathbf{1}) + d\|W\|_F^2. \tag{8}$$

(Dong et al., 2019) has shown that the optimization problem (6) can be considered as a generalization of the graphical Lasso model when the precision matrix is chosen to be a graph Laplacian. The smoothness property associated with a multivariate Gaussian distribution is also behind the idea of graphical Lasso (Egilmez et al., 2017).

Although the aforementioned formulations are theoretically sound and can assure the quality of the learned graph structure, their extremely high complexities do not allow for learning large-scale graphs involving hundred thousands or even millions of nodes. For example, solving the optimization problem in (6) can be very expensive for large data sets; the state-of-the-art methods have a $O(N^2)$ time complexity per iteration for N data entities (Dong et al., 2015; 2016; Kalofolias, 2016). Furthermore, these methods usually require nontrivial parameters tuning for controlling graph sparsity.

3 GRASPEL: A SPECTRAL APPROACH TO GRAPH LEARNING FROM DATA

3.1 PROBLEM FORMULATION

At high level, our approach for graph learning gains insight from recent GSP-based Laplacian estimation methods (Dong et al., 2019), aiming to solve the following optimization problem that is similar to the graphical Lasso problem (Friedman et al., 2008):

$$\textbf{maximize:}_{L\in\mathcal{L}} \ \log\det(L) - \alpha Tr(X^T L X) - \beta\|L\|_1, \tag{9}$$

where $\mathcal{L}$ denotes the set of valid Laplacian matrices. It can be shown that the three terms in (9) are corresponding to $\log\det(\Theta)$, $Tr(\Theta S)$ and $R\|\Theta\|_1$ in (5), respectively. When the precision matrix Θ is restricted to be a graph Laplacian, and each column vector in the original data matrix X is treated as a graph signal vector, there is a close connection between our formulation and the graphical Lasso problem. Since graph Laplacians are symmetric and positive definite (PSD) matrices (or M matrices) with non-positive off-diagonal entries, this formulation will lead to the estimation of attractive GMRFs (Dong et al., 2019).

3.2 OVERVIEW OF OUR APPROACH

To achieve good efficiency in graph learning that may involve millions of nodes, instead of solving (9) directly, we propose a spectral approach for solving (9) implicitly. We define the *spectrally-critical edges* to be the ones that can most effectively perturb the graph spectral properties, such as the first few Laplacian eigenvalues and eigenvectors. Our approach aims to iteratively add the most

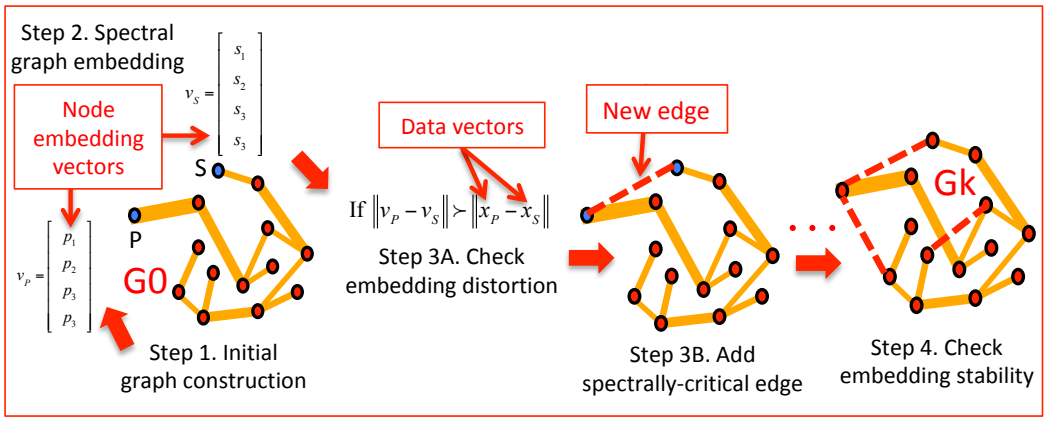

Figure 1: The overview of the proposed GRASPEL framework.

spectrally-critical edges into the latest graph until no such edges can be found, which consists of the following key steps as illustrated in Figure 1:

Step (1): Initial graph construction. We start with constructing an approximate kNN graph where each edge weight encodes the similarity (e.g. Gaussian kernel or cosine similarity) of two nodes (data entities); next, we convert the kNN graph into an ultra-sparse nearest-neighbor (uNN) graph with $O(N \log N)$ edges by leveraging a nearly-linear-time spectral sparsification algorithm (Feng, 2017).

Step (2): Spectral graph embedding. We apply a spectral embedding procedure to the current graph so that each node will be associated with a low-dimensional embedding vector (e.g. v_s for node S in Figure 1), where the embedding dimension (number of eigenvectors) can be determined based on the largest gaps of the first few (e.g. 100) Laplacian eigenvalues (Peng et al., 2015).

Step (3): Spectrally-critical edge identification. We quickly check the *embedding distortion* for each candidate edge (node pair) defined as $\eta = \frac{z^{emb}}{z^{data}}$, where z^{emb} (z^{data}) denotes the distance in the embedding (data) vector space. The edges with the largest η are considered as the most spectrally-critical edges and will be added into the latest graph.

Step (4): Spectral stability checking. We will return the final graph once the overall embedding distortion becomes sufficiently small or stable [1] after repeating the Steps (2)-(3) multiple times for adding new edges.

We note that the spectral graph reduction (coarsening) algorithm and the multilevel Laplacian eigensolver proposed in (Zhao et al., 2018) have been leveraged for achieving nearly-linear runtime scalability in Steps (2)-(3). More detailed descriptions can be found in the following sections.

3.2.1 Initial graph construction

As aforementioned, (approximate) kNN graphs can be used to construct the initial graphs in Step (1), since they can be created in $O(N \log N)$ time (Muja & Lowe, 2009), while being able to approximate the local data proximity (Roweis & Saul, 2000). However, traditional kNN graphs have the following drawbacks: a) the kNN graphs with large k (the number of nearest neighbors) has the tendency of increasing the cut-ratio (Qian et al., 2012); b) the optimal k value is usually problem dependent and can be very difficult to find. In this work, we will start creating an approximate kNN graph with a relatively small k value (e.g. $k = 5$), and strive to significantly improve the graph quality by adding extra spectrally-critical edges through implicitly solving the proposed optimization problem in (9). In the last, a spectral sparsification algorithm (GRASS) [2] has been applied to further simplify the kNN graph into one with only $O(N \log N)$ edges (Feng, 2017).

[1] If the first few Laplacian eigenvalues do not change much over iterations of adding extra edges, the graph spectra is considered stable or robust since adding more edges does not significantly perturb the key (smallest) eigenvalues.

[2] GRASS can be downloaded at https://sites.google.com/mtu.edu/zhuofeng-graphspar

3.2.2 SPECTRAL GRAPH EMBEDDING

Spectral graph embedding directly leverages the first few nontrivial eigenvectors for mapping nodes onto low-dimensional space (Belkin & Niyogi, 2003). The eigenvalue decomposition of Laplacian matrix is usually the computational bottleneck in spectral graph embedding, especially for large graphs (Shi & Malik, 2000; Von Luxburg, 2007; Chen et al., 2011). To achieve good scalability, we exploit multilevel spectrally-reduced graphs that allow for much faster eigenvector (eigenvalue) computations without loss of accuracy (Zhao et al., 2018). Specifically, the multilevel method first spectrally coarsens the fine-level graph into much smaller ones with preservation of key spectral properties, and then map the eigenvectors obtained on the coarse-level graphs back to the fine-level graph; multilevel eigenvector refinements (smoothing) and orthogonalization steps also can be applied to further improve the approximation accuracy (Zhao et al., 2018).

3.2.3 SPECTRALLY-CRITICAL EDGE IDENTIFICATION

Once Laplacian eigenvectors are available for the current graph, we can identify spectrally-critical edges by looking at each candidate edge's embedding distortion (spectral criticality). To this end, we exploit the following first-order spectral perturbation analysis to quantitatively evaluate each candidate edge's impact on the first few eigenvalues. Denote the edge weight by $w_{p,q}$ and the Laplacian eigenvector corresponding to the eigenvalue λ_i by u_i. We define $e_p \in \mathbb{R}^n$ to be a standard basis vector with all zero entries except for the p-th being 1, and $e_{p,q} = e_p - e_q$. The following theorem will allow us to identify the most spectrally-critical edges using first few Laplacian eigenvectors.

Theorem 1 *The spectral criticality $c_{p,q}$ or embedding distortion $\eta_{p,q}$ of a candidate edge (p,q) on the Laplacian eigenvalue λ_i can be properly estimated by $c_{p,q} = w_{p,q}\left(u_i^T e_{p,q}\right)^2 \propto \eta_{p,q} = \frac{z_{p,q}^{emb}}{z_{p,q}^{data}}$.*

Proof: See the appendix.

Edge identification with Fiedler vectors. The idea for identifying spectrally-critical edges is to sort nodes according to the Fiedler vector. Only the nodes with large embedding distances will be examined as candidate edges. Therefore, we are able to limit the search within the candidate edge connections between the top and bottom few nodes in 1D sorted node vector. Only the candidate edges with the top spectral criticality or embedding distortion values will be added into the latest graph. We also avoid adding redundant edges into the graph by checking their spectral embedding results: if two candidate edges have similar spectral embedding results, only one of them will be added. This scheme has also been briefly described in Algorithm 1 in Appendix.

Edge identification with multiple eigenvectors. With k eigenvectors for spectral embedding, we can first project the graph nodes onto a k-dimensional space and perform spectral clustering to group the nodes into k clusters. Next, we only have to examine the candidate edges that connect nodes between two distant clusters in the embedding space, and sort them based on their embedding distortions. To further reduce the computation cost, we first perform spectral graph coarsening and then search for high-distortion candidate edges on the coarse-level graph. Once a small set of top spectrally-critical edges are identified, we will find their corresponding candidate edges in the fine-level graph. Since each coarse-level candidate edge may correspond to multiple candidate edges in the fine-level graph, we will sort them based on their embedding distortions, and only add the ones with largest distortions into the latest graph. The algorithm for spectrally-critical edge identification using multiple eigenvectors has been briefly described in Algorithm 2 in Appendix.

3.2.4 SPECTRAL STABILITY CHECKING

Our method employs an iterative procedure to repeatedly add new edges into the graph and thereby improving the approximation quality of the graph. Specifically, at each iteration new spectrally-critical edges are identified and added to the current graph; we will terminate the iterations when the graph spectra becomes stable: that is when the first few eigenvalues and eigenvectors are no longer changing. Alternatively, we can check if the embedding distortion still keeps improving; if not, the iterations can be terminated, indicating that no edges can be found to significantly perturb the eigenvectors/eigenvalues. In practice, we found that the number of spectrally-critical edges decreases rapidly within very few iterations. In this work, we introduce the following scheme for checking the spectral stability of each graph learning iteration: 1) in each iteration, we compute and record the several smallest eigenvalues of the latest graph Laplacian according to the largest gap between eigenvalues Peng et al. (2015): for example, the first (smallest nonzero) k eigenvalues

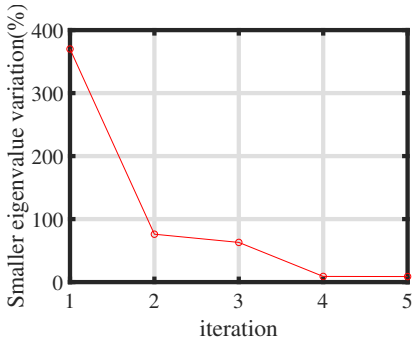

Figure 2: Variation ratio of bottom eigenvalues with increasing number of iterations.

that are critical for spectral clustering tasks will stored; 2) we check whether a sufficiently stable spectra has been reached for graph learning by comparing them with the eigenvalues computed in the previous iteration: if the change is significant, more iterations are needed. To this end, we record the first k Laplacian eigenvalues computed in the previous (current) iteration into vector v_p (v_{p+1}), and calculate the spectral variation ratio by:

$$ratio_{var} = \frac{\|(v_p - v_{p+1})\|}{\|(v_p)\|}. \tag{10}$$

A greater spectral variation ratio indicates less stable eigenvalues within the latest graph, and thus justifies another iteration for adding more "spectrally-critical" edges into the graph. The spectral stability checking results for the USPS data set (see Appendix for details) has been shown in Figure 2.

3.2.5 A Scheme for Eigenvalue Stability Checking.

4 Experiments

In this section, several experiments have been conducted to evaluate the performance of the proposed graph learning method for a variety of public domain data sets.

4.1 Graph learning for spectral clustering

As shown in Algorithm 3 in Appendix, the the classical spectral clustering (SC) algorithm first constructs a graph where each edge weight encodes similarities between different data points (entities); then SC calculates the eigenvectors of the graph Laplacian matrix and embeds data points into low-dimensional space (Belkin & Niyogi, 2003); in the last, k-means algorithms are used to partition the data points into multiple clusters. The performance of spectral clustering strongly depends on the quality of the underlying graph (Guo, 2015). Compared with conventional graph construction (learning) methods (e.g. kNN graphs) may not be effective for handling many complex real-world data sets, and can lead to rather low clustering performance (Nie et al., 2009). In this section, we apply the proposed spectral methods for graph construction, and show the learned graphs can lead to drastically improved efficiency and accuracy in SC tasks. The detailed description of our evaluation metrics, data sets and experiment setup has been provided in Appendix.

Table 1 shows the ACC and NMI results of spectral clustering with graphs constructed by different methods. Graph construction time are also reported in Table 1. The very high computational cost and memory usage of recent graph learning methods, such as GL-SigRep (Dong et al., 2016), GL-Logdet (Dong et al., 2016) and GLSC (Egilmez et al., 2017) do not allow for data sets with more than a few thousands of entities, thus can not be applied for real-world SC tasks.

We can see that the proposed GRASPEL method leads to significant improvement in the performance of SC: GRASPEL beats all the competitors in the aspect of ACC on all data sets; GRASPEL achieves more than 20% accuracy gain on USPS and 10% accuracy gain on COIL20 over the second best method; for MNIST our method also achieves nearly 10% accuracy gain over the standard kNN graph and more than $6X$ speedup in graph construction time; we note that several graph construction methods are not applicable to some of our data sets due to the very high computational (memory) cost. The superior performance of our method is due to the following reasons:

1) in traditional kNN graphs, all the nodes have same neighborhood degrees; as a result, the clustering may strongly favor balanced cut so that some cuts can improperly occur in high-density regions of the graph. In contrast, our method only includes the edges that have largest impact to the graph spectral properties, resulting in ultra-sparse (tree-like) graphs with much lower node degrees. Consequently, the resultant cuts will always occur in proper regions of the graph, which enables to handle even unbalanced data.

2) Our approach is less susceptible to noise in similarity calculations, since our method only connects two nodes according to the edge spectral criticality. Based on matrix perturbation theory (Stewart & Sun, 1990), eigenvectors corresponding to small eigenvalues are more affected by noise in similarities, which can cause problems when the number of desired clusters is not small (Hennig et al., 2015). Although consensus kNN method attempts to use consensus information from kNN graph for discarding noisy edges, the improvement is limited. This is probably due to the fact that it is hard to extract useful consensus information from a given kNN graph when the neighborhood size is small; when large amount of potential noisy edges exist in the graph, the capability of consensus information from a given kNN in circumventing noise can be very limited. The random forest-based (RF-Bi) method leads to degraded performance of spectral clustering on all the four data sets. The potential cause is that the features learned from clustering trees may not be accurate and reliable, especially for high-dimensional data sets.

Table 1: Spectral Clustering Results

	ACC(%)/ NMI/ time(seconds)			
Data Set	standard KNN	ConskNN	GRASPEL(Fiedler)	GRASPEL(Multi)
COIL-20	78.80/ 0.86/ 0.36	79.86/ 0.86/ 2.12	**90.27/ 0.96/** 0.40	85.39/ 0.88/ 0.85
PenDigits	81.12/ 0.80/ 1.25	84.17/ **0.81/** 105.92	**85.96/** 0.80/ 4.51	84.12/ 0.80/6.55
USPS	68.22/ 0.77/ 2.66	72.54/ 0.78/ 396.83	**92.59/ 0.87/** 5.19	90.22/ 0.85/ 8.06
MNIST	71.95/ 0.72/ 242.38	-	**81.67/ 0.75/ 59.27**	79.05/ 0.74/ 75.38

- indicates that the method is not capable for handling data sets of this scale.

4.2 Results for Graph recovery

We also quantitatively evaluate graph recovery performance of the proposed method and compare with recent Laplacian-estimation based graph learning methods, by comparing the graphs learned from observations to the ground-truth graphs. Four widely adopted evaluation metrics in information retrieval are used: Precision, Recall, F-measure and Normalized Mutual Information (Dong et al., 2016). The Precision measures the percentage of correct edges (the edges that are present in the ground-truth graph) in the learned graph. The Recall measures the percentage of the edges in the ground-truth graph that are also in the learned graph. F-measure measures the overall quality by taking both Precision and Recall into account. The results in Table 2 demonstrate the effectiveness of the proposed method in learning graphs that are always very close to the ground-truth graphs. Compared with other graph learning methods that can only handle up to a few hundreds or thousands of data entities, our approach has much better (nearly-linear runtime and space) scalability and thus will be more efficient for handling large data sets.

Table 2: Graph Recovery Results

	The Gaussian graph				The ER graph			
Algorithm	F-measure	Precision	Recall	NMI	F-measure	Precision	Recall	NMI
GL-SigRep	0.8310	0.8120	0.8826	0.5272	0.7243	0.6912	0.8389	0.3600
GL-LogDet	0.8178	0.8193	0.8521	0.4701	**0.7378**	0.6983	0.8030	**0.4012**
GLSC	0.7203	0.6901	0.9000	0.3208	0.6609	0.5427	0.8224	0.3379
GRASPEL	**0.8499**	0.8394	0.8812	**0.5397**	**0.7256**	0.6990	0.8132	**0.3607**

4.3 Applications in t-SNE

The t-Distributed Stochastic Neighbor Embedding (t-SNE) has become the most popular visualization tool for many high-dimensional data analytic tasks (Maaten & Hinton, 2008; Linderman & Steinerberger, 2017). However, its high computational cost ($O(N^2)$ where N is the number of data points in the data set) limits its applicability to large scale problems. (Zhao et al., 2018) proposed a multilevel

t-SNE algorithm to reduce the computational burden by leveraging spectral graph coarsening as a pre-processing step such that a much smaller set of representative data points can be selected for t-SNE visualization. However, to perform spectral graph coarsening, a graph encoding the manifold of the data set has to be constructed first, which can also be very time consuming. In this paper, we apply the proposed spectral graph learning method to accelerate multilevel t-SNE algorithm according to (Zhao et al., 2018).

Fig 3 and Figs 4 show the visualization and runtime results of the standard t-SNE and the multilevel t-SNE algorithm based on the graph constructed by the proposed method. Our method truthfully shows the 10 clusters for both data sets. The runtime for the multi-level t-SNE method includes both the graph construction time and t-SNE time. It can be seen that the run time for visualizing the data sets can be dramatically reduced (12.8X and 7X for MNIST and USPS respectively) without loss of visualization quality.

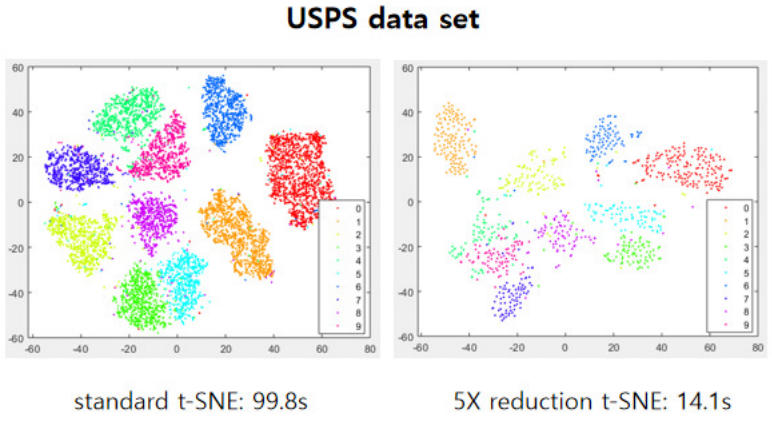

Figure 3: USPS visualization results.

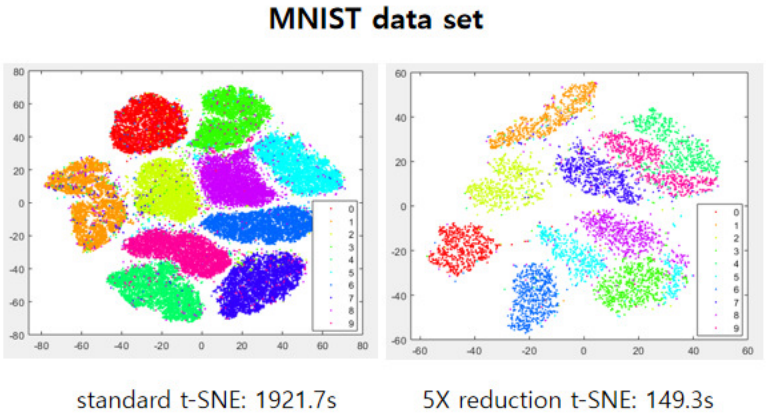

Figure 4: MNIST visualization results.

5 Conclusion

In this work, we present a scalable spectral approach to graph learning from data. By limiting the precision matrix to be a graph Laplacian, our approach aims to estimate ultra-sparse weighted graphs and has a clear connection with the prior graphical Lasso method. Compared with prior graph learning approaches that do not scale to large problems, our approach is highly-scalable for constructing graphs that can immediately lead to substantially improved computing efficiency and solution quality for a variety of data mining and machine learning applications, such as spectral clustering (SC), and t-Distributed Stochastic Neighbor Embedding (t-SNE).

REFERENCES

Francis R Bach and Michael I Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7(Oct):1963–2001, 2006.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

George H Chen, Devavrat Shah, et al. Explaining the success of nearest neighbor methods in prediction. *Foundations and Trends® in Machine Learning*, 10(5-6):337–588, 2018.

Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):568–586, 2011.

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Laplacian matrix learning for smooth graph signal representation. In *2015 IEEE international conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3736–3740. IEEE, 2015.

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23): 6160–6173, 2016.

Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *arXiv preprint arXiv:1806.00848*, 2018.

Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Processing Magazine*, 36(3):44–63, 2019.

Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

Zhuo Feng. Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis. In *Proceedings of the 53rd Annual Design Automation Conference*, pp. 57. ACM, 2016.

Zhuo Feng. Similarity-aware spectral sparsification by edge filtering. *arXiv preprint arXiv:1711.05135*, 2017.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

Xiaojie Guo. Robust subspace segmentation by simultaneously learning data representations and their affinity matrix. In *IJCAI*, pp. 3547–3553, 2015.

Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci. *Handbook of cluster analysis*. CRC Press, 2015.

Tony Jebara and Vlad Shchogolev. B-matching for spectral clustering. In *European conference on machine learning*, pp. 679–686. Springer, 2006.

Tony Jebara, Jun Wang, and Shih-Fu Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 441–448. ACM, 2009.

Vassilis Kalofolias. How to learn a graph from smooth signals. In *Artificial Intelligence and Statistics*, pp. 920–929, 2016.

George C Linderman and Stefan Steinerberger. Clustering with t-sne, provably. *arXiv preprint arXiv:1706.02582*, 2017.

Yang Liu, Quanxue Gao, Zhaohua Yang, and Shujian Wang. Learning with adaptive neighbors for image clustering. In *IJCAI*, pp. 2483–2489, 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Markus Maier, Ulrike V Luxburg, and Matthias Hein. Influence of graph construction on graph-based clustering measures. In *Advances in neural information processing systems*, pp. 1025–1032, 2009.

Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

Feiping Nie, Dong Xu, Ivor W Tsang, and Changshui Zhang. Spectral embedded clustering. In *IJCAI*, pp. 1181–1186, 2009.

Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering with adaptive neighbors. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 977–986. ACM, 2014.

Feiping Nie, Xiaoqian Wang, Michael I Jordan, and Heng Huang. The constrained laplacian rank algorithm for graph-based clustering. In *AAAI*, pp. 1969–1976, 2016.

Christos H Papadimitrou and Kenneth Steiglitz. Combinatorial optimization: algorithms and complexity. 1982.

Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):167–172, 2007.

Yuru Pei, Tae-Kyun Kim, and Hongbin Zha. Unsupervised random forest manifold alignment for lipreading. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 129–136, 2013.

Richard Peng, He Sun, and Luca Zanetti. Partitioning well-clustered graphs: Spectral clustering works. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, pp. 1423–1455, 2015.

Vittal Premachandran and Ramakrishna Kakarala. Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1594–1601, 2013.

Jing Qian, Venkatesh Saligrama, and Manqi Zhao. Graph-based learning with unbalanced clusters. *arXiv preprint arXiv:1205.1496*, 2012.

Michael G Rabbat. Inferring sparse graphs from smooth signals with theoretical guarantees. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6533–6537. IEEE, 2017.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

GW Stewart and JG Sun. Matrix perturbation theoryacademic press. *San Diego*, 1990.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pp. 1601–1608, 2005.

Zhiqiang Zhao, Yongyu Wang, and Zhuo Feng. Nearly-linear time spectral graph reduction for scalable graph partitioning and data visualization. *arXiv preprint arXiv:1812.08942*, 2018.

Xiatian Zhu, Chen Change Loy, and Shaogang Gong. Constructing robust affinity graphs for spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1450–1457, 2014.

.1 PROOF OF THEOREM 1

Let L_P denote the Laplacian matrix of an undirected graph P, and u_i denote the i-th eigenvector of L_P corresponding to the i-th eigenvalue λ_i that satisfies:

$$L_p u_i = \lambda_i u_i, \tag{11}$$

then we have the following eigenvalue perturbation analysis:

$$\left(L_P + \delta L_P\right)\left(u_i + \delta u_i\right) = \left(\lambda_i + \delta\lambda_i\right)\left(u_i + \delta u_i\right), \tag{12}$$

where a perturbation δL_P that includes a new edge connection is applied to L_P, resulting in perturbed eigenvalues and eigenvectors $\lambda_i + \delta\lambda_i$ and $u_i + \delta u_i$ for $i = 1, ..., n$, respectively.

Keeping only the first-order terms leads to:

$$L_P \delta u_i + \delta L_P u_i = \lambda_i \delta u_i + \delta\lambda_i u_i. \tag{13}$$

Write δu_i in terms of the original eigenvectors u_i for for $i = 1, ..., n$:

$$\delta u_i = \sum_{i=1}^{n} \alpha_i u_i. \tag{14}$$

Substituting (14) into (13) leads to:

$$L_P \sum_{i=1}^{n} \alpha_i u_i + \delta L_P u_i = \lambda_i \sum_{i=1}^{n} \alpha_i u_i + \delta\lambda_i u_i. \tag{15}$$

Multiplying u_i^T to both sides of (15) results in:

$$u_i^T L_P \sum_{i=1}^{n} \alpha_i u_i + u_i^T \delta L_P u_i = \lambda_i u_i^T \sum_{i=1}^{n} \alpha_i u_i + \delta\lambda_i u_i^T u_i. \tag{16}$$

Since u_i for for $i = 1, ..., n$ are unit-length, mutually-orthogonal eigenvectors, we have:

$$u_i^T L_P \sum_{i=1}^{n} \alpha_i u_i = \alpha_i u_i^T L_P u_i, \qquad \lambda_i u_i^T \sum_{i=1}^{n} \alpha_i u_i = \alpha_i u_i^T \lambda_i u_i. \tag{17}$$

Substituting (11) into (17), we have:

$$\alpha_i u_i^T L_P u_i = \alpha_i u_i^T \lambda_i u_i. \tag{18}$$

According to (17), we have:

$$u_i^T L_P \sum_{i=1}^{n} \alpha_i u_i = \lambda_i u_i^T \sum_{i=1}^{n} \alpha_i u_i. \tag{19}$$

Substituting (19) into (16) leads to:

$$u_i^T \delta L_P u_i = \delta\lambda_i u_i^T u_i = \delta\lambda_i. \tag{20}$$

Then the eigenvalue perturbation due to δL_P is given by:

$$\delta\lambda_i = w_{p,q} \left(u_i^T e_{p,q}\right)^2. \tag{21}$$

If each edge weight $w_{p,q}$ encodes the similarity of data vectors x_p and x_q at nodes p and q, it can be shown that $w_{p,q} \propto \frac{1}{z^{data}}$, where z^{data} denotes the distance between x_p and x_q; on the other hand, $\left(u_i^T e_{p,q}\right)^2 \propto z^{emb}$. Therefore, as long as we can find an edge with large $w_{p,q} \left(u_i^T e_{p,q}\right)^2$ or $\eta_{p,q} = \frac{z_{p,q}^{emb}}{z_{p,q}^{data}}$, including this edge into the current graph will significantly perturb the Laplacian eigenvalue λ_i and eigenvector u_i.

.2 ALGORITHMS

Algorithm 1 Spectrally-critical edge identification with Fiedler vectors

Input: a data set D with n data points $x_1, ...x_n \in \mathbb{R}^d$, window size η, edge selection ratio ζ.
Output: updated graph.

1: Construct an initial graph G_i as in (Chen et al., 2011).
2: Initialize: $Terminate$=0;
3: **while** $Terminate$==0 **do**
4: Embed G_i with the Fiedler vector and sort the data points (nodes).
5: Evaluate the embedding distortions of candidate edges connected between the top π and bottom π sorted nodes.
6: Select top ζ N edges based on the evaluation result and add them to G_i.
7: Check the spectral stability and update $Terminate$.
8: **end while**

Algorithm 2 Spectrally-critical edge identification with multiple eigenvectors

Input: r selected edges in reduced graph, original graph G_{orig}, desired number of edges to be added in this iteration l .
Output: updated graph.

1: **for** each selected edges in reduced graph **do**
2: **for** each of its two nodes **do**
3: Find its corresponding set of nodes in G_{orig}
4: **end for**
5: Form the edge set E_{orig} between the two set of nodes
6: **if** $|E_{orig}| \leq \frac{l}{r}$ **then**
7: Add all edges $\in E_{orig}$ to the graph G_{orig}
8: **else**
9: **for** each edge $\in E_{orig}$ **do**
10: Evaluate it
11: **end for**
12: Sort edge $\in E_{orig}$ based on the evaluation results and add the top $\frac{l}{r}$ ones to the graph G_{orig}
13: **end if**
14: **end for**

Algorithm 3 Spectral Clustering Algorithm

Input: A graph $G = (V, E, w)$ and the number of clusters k.
Output: Clusters $C_1...C_k$.

1: Compute the adjacency matrix A_G, and diagonal matrix D_G;
2: Obtain the unnormalized Laplacian matrix L_G=D_G-A_G;
3: Compute the eigenvectors u_1,...u_k that correspond to the bottom k nonzero eigenvalues of L_G;
4: Construct $U \in \mathbb{R}^{n \times k}$, with k eigenvectors of L_G stored as columns;
5: Perform k-means algorithm to partition the rows of U into k clusters and return the result.

.3 DATA SETS DESCRIPTION

COIL20: A data set contains $1,440$ gray-scale images of 20 objects, and each object on a turntable has 72 normalized gray-scale images taken from different degrees. The image size is 32x 32 pixels.

PenDigits: A data set consists of 7,494 images of handwritten digits from 44 writers, using the sampled coordination information. Each digit is represented by 16 attributes.

USPS: A data set includes $9,298$ scanned hand-written digits on the envelops from U.S. Postal Service with 256 attributes.

MNIST: A data set consists of 70,000 images of handwritten digits. Each image has 28-by-28 pixels in size. This database can be found from Prof.Yann LeCun's website (http://yann.lecun.com/exdb/mnist/).

.4 COMPARED ALGORITHMS

Standard kNN : the most widely used affinity graph construction method. Each node is connected to its k nearest neighbors.

Consensus of kNN (cons-kNN) (Premachandran & Kakarala, 2013) : adopts the state-of-the-art neighborhood selection methods to construct the affinity graphs. It selects strong neighborhoods to improve the robustness of the graph by using the consensus information from different neighborhoods in a given kNN graph.

Random forest-based Method (RF-Bi)(Pei et al., 2013; Zhu et al., 2014): constructs affinity graphs by exploiting discriminative features during the stage of training the clustering forests.

GL-SigRep (Dong et al., 2016): construct a graph from signals that are assumed to be smooth with respect to the corresponding graph.

GL-LogDet (Dong et al., 2016): encodes the information about the partial correlations between the variables without the constraint to form a valid Laplacian.

Graph Learning under structural constraints(GLSC) (Egilmez et al., 2017): formulated the problem as to maximum a posterior estimation of GaussianMarkov Random Field (GMRF) when the precision matrix is chosen to be a graph Laplacian.

.5 EVALUATION METRIC

1)

$$ACC = \frac{\sum_{j=1}^{n} \delta(y_i, map(c_i))}{n}, \tag{22}$$

where n is the number of samples in the data set, y_i is the ground-truth label provided by the data sets, and c_i is clustering result obtained from the algorithm. $\delta(x, y)$ is a delta function defined as: $\delta(x, y)$=1 for $x = y$, and $\delta(x, y)$=0, otherwise. $map(\bullet)$ is a permutation function that maps each cluster index c_i to a ground truth label, which can be realized using the Hungarian algorithm (Papadimitrou & Steiglitz, 1982). ACC measures the agreement between the clustering results generated by clustering algorithms and the ground-truth labels. A higher value of ACC indicates better clustering quality.

2)

For two random variables P and Q, normalized mutual information is defined as (Strehl & Ghosh, 2002):

$$NMI = \frac{I(P, Q)}{\sqrt{H(P)H(Q)}}, \tag{23}$$

where $I(P, Q)$ denotes the mutual information between P and Q, while $H(P)$ and $H(Q)$ are entropies of P and Q. In practice, the NMI metric can be calculated as follows (Strehl & Ghosh, 2002):

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} n_{i,j} \log(\frac{n \cdot n_{i,j}}{n_i \cdot n_j})}{\sqrt{(\sum_{i=1}^{k} n_i \log \frac{n_i}{n})(\sum_{j=1}^{k} n_j \log \frac{n_j}{n})}}, \tag{24}$$

where n is the number of data points in the data set, k is the number of clusters, n_i is the number of data points in cluster C_i according to the clustering result generated by algorithm, n_j is the number of data points in class C_j according to the ground truth labels provided by the data set, and $n_{i,j}$ is the number of data points in cluster C_i according to the clustering result as well as in class C_j according to the ground truth labels. The NMI value is in the range of [0, 1], while a higher NMI value indicates a better matching between the algorithm generated result and ground truth result.