Large scale graph inference from smooth signals

Aakanksha Choudhary
Department of Electrical Engineering
IIT Kanpur
aakanksh@iitk.ac.in
150002

Vishal Srivastava
Department of Electrical Engineering
IIT Kanpur
vishalsr@iitk.ac.in
150820

Type: Application based

***Abstract*—Graph signal processing is an effective framework to study correlations among unstructured data, and is widely used in machine learning. They provide systematic approaches for clustering, manifold learning, dimensionality reduction, sampling etc. Graphs can capture inherent underlying structure from large datasets, but often the ground truth relating the actual graph is unknown. We study several approaches to learn a graph $\mathcal{G}$ from a set of smooth signals assumed to be residing on $\mathcal{G}$. The earlier methods though achieved great quality of recovery, they were computationally expensive rising to $\mathcal{O}(m^2)$, hence the recent methods have tried to overcome this curse: still achieving similar performance as the existing state of the art.**

***Index Terms*—graph signal processing, graph learning, large scale optimization, laplacian smoothness, network topology inference**

I. Introduction

Graphs are a convenient tool to describe and capture several complex relations among huge amount of data. A weighted graph is utilised to represent the pairwise similarity/dissimilarity between nodes, and the nodes represent samples. Many examples where such representations make good sense are social network graph, wireless sensor networks, fMRI signals from brain, and even images which seems like having an underlying 2D grid graph. The recent tools developed in graph signal processing [1], [2] have advanced our understanding of graph operations. Once we know the graph, it opens way to numerous data analysis techniques such as manifold learning, clustering, semi supervised learning, PCA, matrix completion, random walks etc.

However, in the scenario when the underlying graph structure is not known, it becomes a challenging problem to learn actual graph from observed data. There are huge number of combinations possible ($\mathcal{O}(m^2)$) and usually actual graphs are sparse in nature. Thus we are left with a learning task: Given a set of graph signals, estimate the underlying graph with a good confidence.

A. Preliminaries

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has set of m vertices and $\|\mathcal{E}\|$ edges. It has its associated adjacency matrix W and a Laplacian $\mathcal{L} = D - W$, where D is a diagonal degree matrix. Laplacian $\mathcal{L}$ is decomposed as $U\Lambda U^T$. The eigenvalues constitute the spectrum of the graph.

Smooth Signals: We consider a martrix $X \in R^{m\times n} = [\ x_1\ \ x_2\ \ ...\ \ x_m]^{\ T}$ where each row $x_i \in R^n$ resides on one node of a graph. A general assumption about the datapoints (or the graph signal) is that they are smooth, i.e,, their manifold is regularised. A quantitative measure for this is defined as $tr(X^T LX) = \frac{1}{2}\sum_{i,j} W_{ij}\|x_i - x_j\|^2 = \frac{1}{2}\|W \circ Z\|_{1,1}$, (where Z is the distance matrix), which should be small for the signal to be smooth. Upholding this assumption, it becomes possible to infer some properties about the Laplacian L.

Earlier approaches to graph learning were based on k-NN method [Zhu et al. [8], Belkin et al. [9] where the number of neighbours remained fixed across all nodes. Wang and Zhang [10] learn a graph with normalized degrees by minimizing the objective $\sum_i \|x_i - \sum_j w_{ij}x_j\|^2$, but they assume a fixed k-NN edge pattern. Daitch, Kelner, and Spielman [2009] considered the similar objective $\|LX\|_2^F$ and they obtained suboptimal solutions.

However a universal approach has been proposed by Kalofolias [2016] which poses the graph learning problem as an optimization of objective of following form:

$$\min_{L\in\mathcal{L}} tr(X^T LX) + f(L) \tag{1}$$

This was a benchmark among some recent ones because: 1) It enables learning the laplacian directly, 2) Most of the earlier problems can be formulated as a special case of this model.

II. Problem Setup and Perspectives

The problem of graph learning under smoothness assumption of signals can be formulated in general as (1) and solved for some regularising function $f(.)$ However there are several caveats that need attention before such an optimization problem can framed. The Laplacian has to follow several constraints, hence the search space $\mathcal{L}$ has to be restricted to valid Laplacians, i.e., symmetric, positive definite, non-diagonal entries should be non-positive etc. This imposes numerous constraints and makes it computationally demanding to solve.

Kalofolias [3] argued that an equivalent representative of L can be the adjacency matrix W whose optimization over its search space is not only easier but also intuitive. In fact, the symmetry and zero-diagonal of W helps reduce the dimension of search space from $R^{m\times m}$ to $R^{m(m-1)/2}$. Moreover, learning of W provides us with more "visibly upfront" intuitions

about the graph than L. This also has an implication of why authors in [3] converted original formulation of Dong et al.[7] in Laplacian space to $\mathcal{W}$ space.

Sparsity: Since optimization is now over the variable vec(W), it becomes easier to look for effect of $f(W)$ on sparsity. A common choice of to prevent sparsity is to ensure a logarithmic barrier: it prevents the weights from going to zero. This can also be similarly imposed by keeping degree per node i,e,, $norm(W\mathbf{1})$ small. The respective coefficients to these prior terms then directly effect the sparsity of learned graphs - which is quite intuitive and practical.

Complexity: From the ongoing discussion of sparsity and search space: a question which directly follows is of complexity. What would be the computational scale of solving such an optimization problem? It turns out that even when searching for W^* in $\mathcal{W} \in R^{m(m-1)/2}$, the complexity is $\mathcal{O}(m^2)$ which is not an efficient method. This method becomes computationally infeasible for large graphs. An alternative approach based on edge mask was proposed by Perraudin and Kalofolias [4] wherein they reduced the search space to only $|\mathcal{E}^{allowed}|$ which should be of smaller order than m^2.

Probabilistic approach for noisy case: In the present works of graph topology inference, there hasn't been many approaches to tackle the problem of noisy graph, i.e., when the graph is not deterministic. This problem can arise when the underlying graph is not known to follow any single "valid" Laplacian, but a combination of several "valid" Laplacians, resulting in an "invalid" mixture of graphs. This randomness has to be dealt when the nodes giving out information have any element of randomness in them, which is found to be true in case of brain signals. Hence, a probabilistic treatment of graph learning becomes necessary when such is the case. Maretic and Frossard [6] have proposed a simple yet elegant solution to solve this mixture of graphs problem.

III. Standard Approaches to Graph Learning

A. k-nearest neighbour

The early approach of learning a graph by k-nearest neighbour is to compute edge weights given X from the Gaussian function

$$w_{ij} = exp\big(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\big) \tag{2}$$

and put the weights less than a threshold to zero. Then the k-nearest neighbors are connected with an edge. The optimization of the problem can also be accommodated in the model (1) by setting up $f(W) = 2\sigma^2 \sum_{ij} W_{ij}(logW_{ij} - 1)$. This approach is very rigid, as it fails miserably in community structured graphs due to its nature of assigning same number of neighbors to every node. More than that, the cost needed to compute W inevitably takes $\mathcal{O}(m^2)$ computations. Although, it fails to keep up with expected bar, it has the advantage of always yielding k edges/nodes when given an input k.

B. l2 model

Dong et. al.[7] proposed an optimization problem for learning the graph fitting into model (1) with $f(W)$ given by $f(W) = \alpha\|W\mathbf{1}\|^2 + \alpha\|W\|_F^2$.

$$\min_{W\in\mathcal{W}} \|W \circ Z\| + \alpha\|W\mathbf{1}\|^2 + \alpha\|W\|_F^2 \tag{3}$$

$s.t., \|W\|_{1,1} = s$. This model penalizes large degrees through the first term of $f(W)$ and hence allows us to tune the sparsity of graph by parameter α. An interesting property of (3) is: [3, Proposition 3]

Proposition 1: *Let $F(Z, \alpha, s)$ denote the solution of model (3) for input distances Z and parameters α and s. Then the following property holds:*

$$F(Z, \alpha, s) = sF(Z, \alpha s, 1) \tag{4}$$

This means that the scale of our distance matrix does not effects the connectivity and we can obtain all possible types of F by tuning only α.

C. log model

The "*log model*" was proposed by Kalofolias in [3], which achieved state of the art performance. This was built upon the l2 model with a difference: they penalized the logarithm of degree of each node instead of the norm of complete degree vector.

$$\min_{W\in\mathcal{W}} \|W \circ Z\| - \alpha\mathbf{1}^T log(W\mathbf{1}) + \beta\|W\|_F^2 \tag{5}$$

This prevented extreme sparsity by connecting each node to atleast one other node! This model achieved better results than the l2 model both on artificial datasets as well as on clustering and classification task on real datasets.

Although it yields better results than (3), its computational time is similar to that of l2 model, yet it suffers from the problem of grid search over two parameters α, β for tuning the sparsity of learned graph. This apparently could have made it cumbersome to employ on real datasets, but it has few nice properties obtained by Perraudin and Kalofolias in [4] which makes it state-of-the-art.

Proposition 2: *Let $F(Z, \alpha, s)$ denote the solution of model (5) for input distances Z and parameters α and β. Then the following property holds:*

$$F(Z, \alpha, \beta) = \alpha F(Z, 1, \alpha\beta) \tag{6}$$

This means that if we fix a scale, say $\|W\| = s$, then we can fix $\alpha = 1$ and search for β to obtain a desired edge density.

In [4], the authors suggested a way to obtain the hyper parameters to control the desired edge density, i.e., avg. number of edges/node. [4 ,Proposition 1] is restated here:

Proposition 2: *Let $F(Z, \alpha, \beta)$ denote the solution of model (5) for input distances Z and parameters α and β. Then same solution can be found with $\theta = \frac{1}{\sqrt{\alpha\beta}}$ and $\delta = \sqrt{\frac{\alpha}{\beta}}$.*

$$F(Z, \alpha, \beta) = \delta F(\theta Z, 1, 1) \tag{7}$$

This means that if we set δ to maintain scale, the only parameter left to tune sparsity is θ. Hence, given some k, we can obtain k-sparse graph by choosing [4, (17)]

$$\theta_k \in \Big(\sum_{j=1}^{n} \frac{1}{n\sqrt{kZ_{k+1,j}^2 - B_{k,j}Z_{k+1,i}}}, \sum_{j=1}^{n} \frac{1}{n\sqrt{kZ_{k,j}^2 - B_{k,j}Z_{k,i}}} \Big] \tag{8}$$

where $B_{k,j} = \sum_{i=1}^{k} Z_{i,j}$.

D. Improving complexity by edge masks

In above graph learning methods, almost all the ${}^{m}C_2$ edges between m nodes are considered, that results in $\mathcal{O}(m^2)$ computations per iterations. We can utilise an edge mask to restrict $|\mathcal{E}|$ to $|\mathcal{E}^{allowed}|$ so that number of variables to learn also reduces. To approximate the edge set, approximate nearest neighbors method [5] comes in hand. While calculating pairwise distance matrix Z requires $\mathcal{O}(m^2 d)$ computations for d-dimensional data, *approximate nearest neighbor (ANN)* achieve similar accuracy in with overall complexity of $\mathcal{O}(mlog(m)d)$, hence allowing us to scale our standard algorithms for large datasets, even with order of $m \approx 1M$.

To scale the above mentioned 3 algorithms to large datasets, the computation of Z is performed by ANN in $\mathcal{O}(mlog(m)d)$ cost, followed by respective learning algorithms. The edge mask is chosen such that for desired k-sparse graphs, $|\mathcal{E}^{allowed}|$ is chosen to be $\mathcal{O}(mkr)$, where r is a small multiplicative factor greater than 1. This means that initially we relax the mask to $\mathcal{O}(mkr)$ edges, and the learning algorithm eventually reduces it to desired order of $\mathcal{O}(mk)$ of edges per node.

E. Mixture models for noisy graphs

In [6]'s work, a graph laplacian mixture model is proposed. There is a possibility that the underlying graph is assumed to be a gaussian mixture model of K several graphs $\mathcal{G}_{\|}$, each with weight $p(z_{m,k} = 1) = \alpha_k$. We model data in cluster k with mean μ_k and graph Laplacian L_k. With this model, we follow the simple expectation maximization routine to alternatively learn mixture weightsα, μ and $\mathbf{L}$.

Although this may seem like a digression from our current approaches, this also provides us a way to learn single graph if we set K = 1 in mixture model. The objective is defined as:

$$arg\max_{\mu,\mathbf{L}} \sum_{m} ln\mathcal{N}(x_m|\mu, L^+)p(L) \tag{9}$$

Note: It was earlier proposed to simulate this[6] model's performance against other approaches, but could not be realized since three other related algorithms were found suitable. Hence, this model did not involved any experiments.

IV. Simulations

We compare the three models: kNN, l2 model and log model on artificial and real datasets. Also, later we scale the three models by using FLANN library for implementing ANN which computes Z using edge mask in $\mathcal{O}(mlog(m))$ order. We call these the *large scale* models.

TABLE I: Performance on Artificial Data

Graph Community	Heat Diffusion *kNN*	*l2*	*log*
RGG			
edge l-1	0.78	0.70	**0.48**
edge l-2	0.70	0.56	**0.39**
degree l-1	0.17	0.16	**0.11**
degree l-2	0.86	**0.44**	0.73

A. Artificial Dataset

The scheme of experiments is as follows: Generate a graph, generate a smooth signal, learn the original graph from three algorithms and compare them based on some metrics. The details are:

Graph Types: Community, David Sensor Net, Swiss Roll and Random Geometric Graph.

(*almost* each with m = 100).

Signal: We generate random Gaussian i.i.d signals $x_0 \in R^{1000}$ drawn from $\mathcal{N}(0, I)$ and filter them through three types of filters: a) Tikhonov b) $\frac{1}{\sqrt{\lambda}}$ c) Heat Diffusion, followed by adding 10% noise in $l-2$ sense.

Metrics: We measured the relative $l-1$ and $l-2$ error between edge weights and degrees between the learned graph and ground truth. The norm of each $vec(W)$ was scaled to make them uniform across all cases.

The experiment was performed for all many combinations, but here we report only a few due to lack of space. Table 1 shows the results.

B. Real Datasets

The simulations for real dataset of USPS digits was carried out exactly as in the paper [3]. The class distributions, parameters values etc. were taken to be same as in the paper. We performed two experiments on learnt graph: spectral clustering and label propagation. Label propagation was based on the work of Zhu et al.[8]. Figure 1a shows the spectral clustering error as k varies, while figure 1b shows error after label propagation.

Note: The experiment on MNIST digit database as done in paper [3] was not performed because of scalability issues: it has 60K images, and without ANN to speed it up, it was taking too long.

C. Large scale datasets and experiments

Computational time scaling: simulations were done to compare time taken to compute a graph on word2vec database. The time of small scale log model and large scale log model were plotted in Figure 2.

Approximation quality of large scale model: When computing edge mask, we use FLANN library that implements the fast ANN algorithm. However, it comes with reduced performance. We plot the relative error between these two log models for graph learnt by 1000 images of MNIST.

Quality of edges and connectivity: We analyse the connectivity of nodes in different classes. In histograms, the

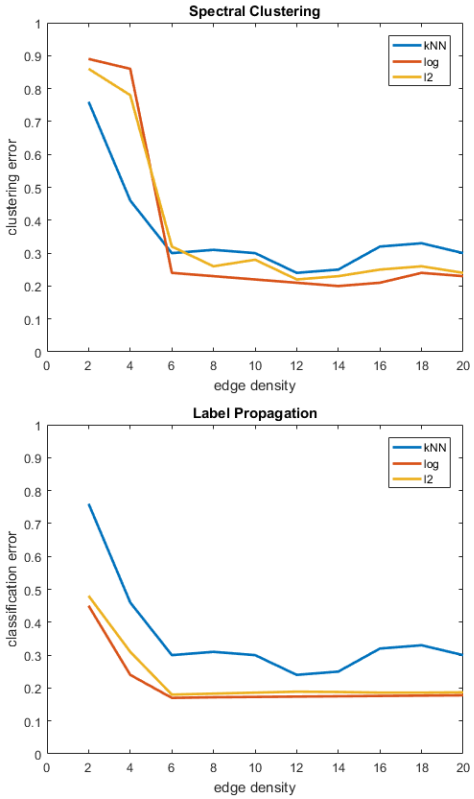


Fig. 1: Spectral clustering and label propagation error for graph learned from 1005 USPS digits

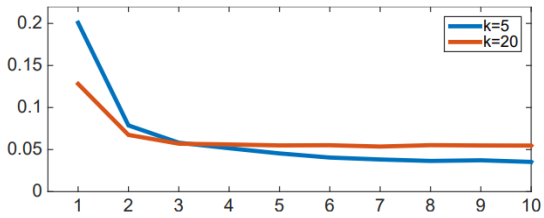


Fig. 2: approximation error between large scale and small scale log model

plot of norm of degrees is plotted: higher value means good intra-class connectivity. The A-NN graph could not take into account the different class sizes, also gives out many wrong edges. The log model gives similar connectivity across all edges, while the l2 model assigned most of its connectivity to label "1". Figure 3 shows the respective histogram plots.

Wrong edges: We performed simulations to account for edge accuracy of the graph learnt from MNIST digits. It is evident that the large scale log and l2 model outperform ANN by sufficient margin. It is also remarkable that ANN performs poorly as avg. degree per node increases. This is shown in figure 5.

Preventing sparsity: In figure 6, we show the plots of

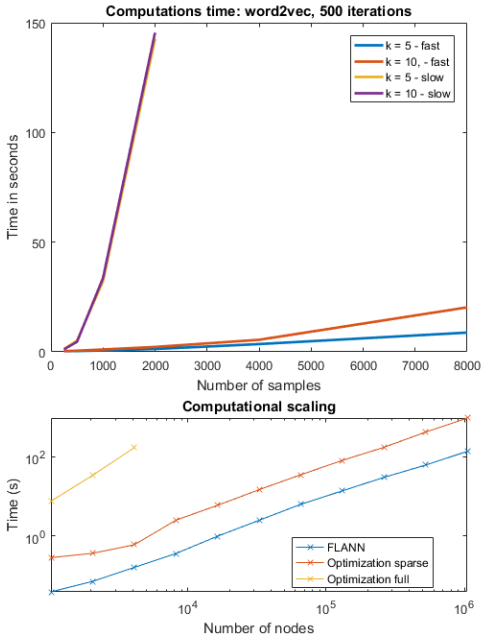


Fig. 3: a) Time scale for graph between word2vec database. b) Graph between 1,000,000 nodes from 68 features (US Census 1990)

number of nodes having degree ¡= 1,2,3 vs k for graph learnt with three models. We observe that for log model, there were still sufficient number of nodes with non-zero degree as k was increased. i.e., the graph had lesser number of isolated nodes. This is an indication of preserving connectivity as compared to ANN model.

Manifold recovery: We perform a simple intuitive experiment on word2vec database to learn a graph with three models and assess the relevance of related words. It was found that l2 and log model cluster the nodes based on an underlying latent structure, whereas ANN connects nodes without weighing the hidden data. Figure 7 shows words connected to "publication" on a 2-hop graph for three models.

V. Conclusions

We studied several methods to learn graph from smooth signals, based on smoothness prior and regularity on adjacency matrix. The earlier models suffered from scalability problem, so a fast method based on ANN was utilised to reduce the complexity order. The state of the art algorithms performed better than traditional kNN method: since they exploit the freedom of sparsity and underlying structure. The graph learnt using log and l2 models performed better on clustering, semi-supervised learning etc. For the problem of random graphs, i.e., when there is a possibility of noise, a probabilistic treatment of mixture models can be used.

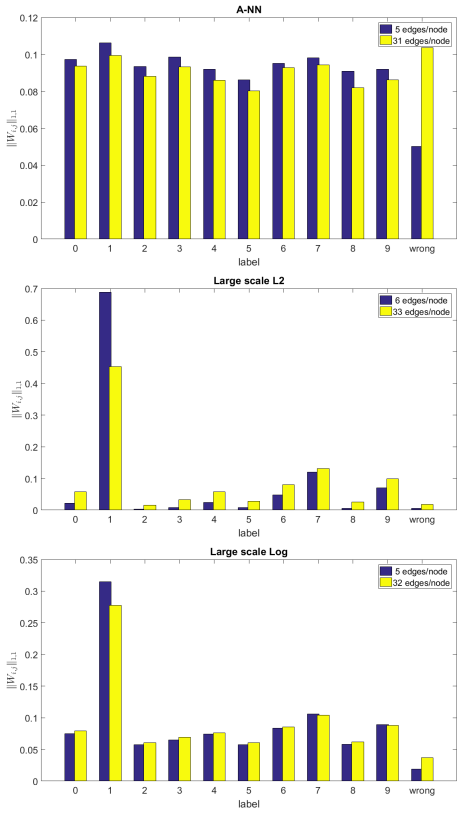


Fig. 4: Interclass connectivity among graph learnt on MNIST database by a) ANN, b) large scale l2, c) large scale log

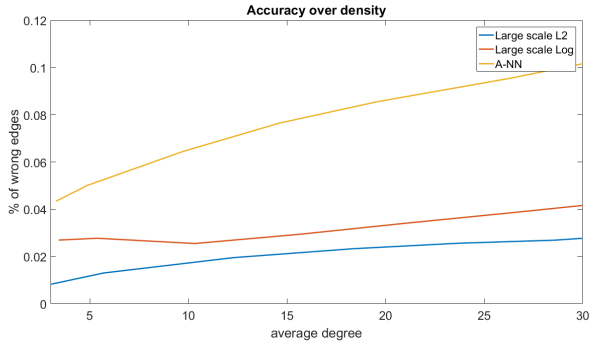


Fig. 5: Edge accuracy of large scale models for MNIST

VI. References

[1]: Tong Zhang, Alexandrin Popescul, and Byron Dom. Linear prediction models with graph regularization for web-page categorization. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 821826. ACM, 2006.

[2]: Nauman Shahid, Nathanael Perraudin, Vassilis Kalofolias, Gilles Puy, and Pierre Vandergheynst. Fast robust pca on graphs. IEEE Journal of Selected Topics in Signal Processing, 10(4):740756, 2016.

[3]: Vassilis Kalofolias. How to learn a graph from smooth signals. In The 19th International Conference

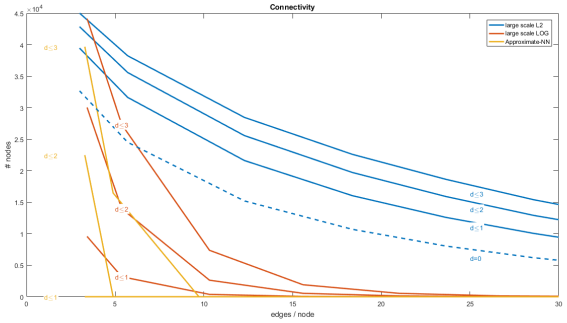
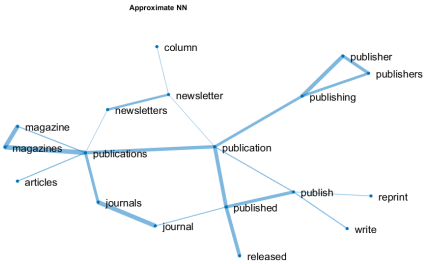
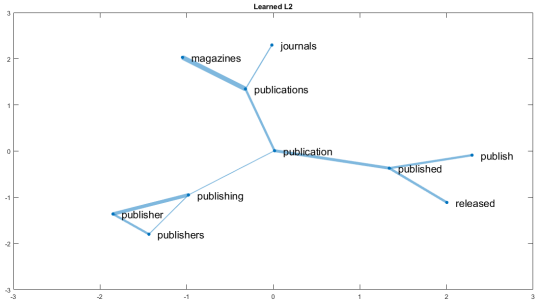


Fig. 6: Number of nodes which have sufficient degree vs k: log model performs better in keeping more number of nodes connected and keeping the graph less sparse





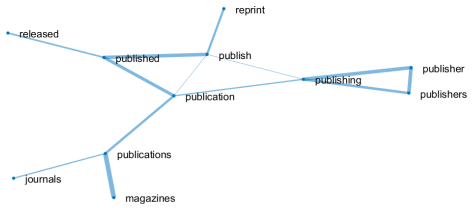


Fig. 7: log and l2 model exploit the underlying manifold to learn the graph, hence have more relevant connections

on Artificial Intelligence and Statistics (AISTATS 2016). Journal of Machine Learning Research (JMLR), 2016.
[4]: Vassilis Kalofolias, Large Scale Graph Learning from Smooth Signals, ICLR 2019.
[5]: Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(11):22272240, 2014.
[6]: Maretic, Frossard, Graph Laplacian mixture model, https://arxiv.org/abs/1810.10053
[7]: Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst. Learning laplacian matrix in smooth graph signal representations. arXiv preprint arXiv:1406.7842v2, 2015.
[8]: Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In ICML, volume 3, pp. 912919, 2003.
[9]: Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. The Journal of Machine Learning Research, 7:23992434, 2006.
[10]: Fei Wang and Changshui Zhang. Label propagation through linear neighborhoods. Knowledge and Data Engineering, IEEE Transactions on, 20(1):5567, 2008.