Prediction Correlation via Graph Inference

Abhay Singh

Abstract

We propose a novel framework to improve predictive performance by learning a graph topology underlying a set of data points. Specifically, given data points $X = [x_1, \ldots, x_n]^\top \in \mathbb{R}^{n \times p}$ whose rows represent vertices, we learn edge weights of a symmetric adjacency matrix $A \in \mathbb{R}^{n \times n}$ to improve a linear model's performance in predicting labels $Y \in \mathbb{R}^n$. We restrict A to be doubly stochastic so that the matrix-matrix product AX updates each vertex to be a convex combination of its neighbors. Our framework naturally leads to sparse network topologies due to a smoothness regularization involving the graph Laplacian.

1 Introduction

Given samples of data points and corresponding labels $S := \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^p \times \mathbb{R}$, we are interested in the following question

> Is there a graph that improves prediction on this collection of samples?

In this problem of *graph topology inference*, one represents the ith data point $x_i \in \mathbb{R}^p$ as the ith vertex of a graph, and seeks to determine edge weights between all pairs of data points (x_i, x_j), for all $i \neq j$, in an undirected graph G. Under this lens, each of the p features can be seen as weights to propagate over the graph. One of the most natural and common assumptions about data that lies over the graph is that it varies *smoothly* over the graph, so that feature vectors on nodes which are connected do not differ greatly. There is a straight-forward way to quantify how 'smooth' a set of data points $\{x_i\}_{i=1}^n$ vary across a weighted undirected graph. As before, we have the matrix of data points $X = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^{n \times p}$, where the ith row of X is simply x_i. Then we can measure smoothness as follows,

$$\frac{1}{2} \sum_{i,j=1}^{n} A_{ij} \|x_i - x_j\|_2^2 = \operatorname{tr}(X^\top L X), \tag{1.1}$$

where the non-negative value $A_{ij} \in \mathbb{R}_{\geq 0}$ is the weight of the edge (i, j), and the Laplacian L is defined as $L = D - A$, with $D_{ii} = \sum_j A_{ij}$ and $D_{ij} = 0$ for $i \neq j$. Using this quantity, the assumption of smoothness over the graph states that if two vectors x_i and x_j are 'well-connected' (i.e., A_{ij} is large), then the squared distance $\|x_i - x_j\|_2^2$ is small.

The graph Laplacian L has been long recognized and utilized as a tool for embedding, manifold learning, clustering, and semi-supervised learning. In the case of semi-supervised learning, methods often incorporate the term in (1.3) but use labels (i.e., Y instead of X) and solve a problem of the form

$$\underset{Y}{\text{minimize}} \ \operatorname{tr}(Y^\top L Y) + g(Y), \tag{1.2}$$

where, for example, the labels are only partially known, and the goal is to infer the labels of the unlabeled samples. In this scenario, if $g(Y)$ encourages our final predictions to be 'close' to our known labels, this gives us the famous label spreading algorithm [1]. This is one approach to encourage 'smooth' predictions. For the task of graph inference, this inspires another approach, where the Laplacian can vary,

$$\underset{L \in \mathcal{L}}{\text{minimize}} \ \operatorname{tr}(X^\top L X) + h(L), \tag{1.3}$$

where $\mathcal{L}$ is the space of valid graph Laplacians and $h(L)$ is some regularization on the Laplacian [2].

A recently successful approach [3, 4] is to forego the graph Laplacian L altogether, and directly 'correlate' features before multiplying by a weight matrix. This is the core idea of graph neural networks which are also used in the semi-supervised learning scenario. As one linearized instantiation, a graph neural network defines $\tilde{A} = A + I$ (to add self-loops), then computes the symmetrically adjacency normalized matrix $S := \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$. Pre-multiplying the data matrix X by S, i.e., computing the matrix-matrix product SX, has the effect of correlating features x_i and x_j for which there exists an edge (i, j). This product is weighted by the degrees of both i and j. Writing this out gives us

$$[SX]_i = \frac{1}{\sqrt{(d_i + 1)(d_j + 1)}} \left(x_i + \sum_{j=1}^{n} A_{ij} x_j \right) \tag{1.4}$$

where d_i is shorthand for D_{ii}. Notably the ith row of SX contains feature information of the feature x_i in the ith row of X, since a self-loop has been added, i.e., $\tilde{A}_{ii} = 1$, and we have assumed $A_{ii} = 0$. It is common to increase the 'receptive field' of any one node by incorporating features from k-hop neighborhoods, which is simply done by calculating $S^k X$ for some $k \in \{1, 2, 3, \ldots\}$. For example, setting $k = 2$ aggregates information over two-hop neighborhoods.

When the graph topology is already provided, models can use $S^k X$ as the input data (instead of X) and this will correlate predictions. To illustrate why, we present a simple example. Supposes we had a graph on two features x_u and x_v with a single edge (u, v), then computing SX will simply average both nodes' features. Thus, any prediction on the updated row of u will be identical to the prediction on the the row of v. More generally for all graphs, the ith row of SX incorporates the addition of x_j for which $A_{ij} \neq 0$, and vice versa due to A being symmetric. Then the predictions from a linear classifier will be more similar due to the premultiplication of S. In the case of linear regression, this results in solving

$$\underset{\beta}{\text{minimize}} \ \|Y - \left(S^k X\right) \beta\|_2^2, \tag{1.5}$$

to create a predictor $\hat{Y} = S^k X \hat{\beta}$, where $\hat{\beta}$ is the solution to problem (1.5). In practice, problem (1.5) will be calculated on a subset of the labels (e.g., the training samples), and then the prediction will be used for the test samples.

The use of the adjacency matrix as a way to quantify smoothness through the Laplacian, as well as encourage smooth predictions over the graph through the matrix-matrix product inspires our method, as we will see in Section 2. Our work is related to the idea of linear smoothers, used in nonparametric statistics [5].

2 Methodology

We combine the approaches of problem (1.3) and (1.5), which inspires our method. In particular, we wish to learn an adjacency matrix A that is smooth over the data, i.e., the value of the trace term

$\operatorname{tr}(X^\top LX)$ is small, where L is the graph Laplacian of A. We also want the same matrix A to improve predictive performance of a linear model on the data, which we will formalize as follows,

Desired approach.

$$\begin{aligned} \underset{A,\beta}{\text{minimize}} \quad & \|Y - \left(\frac{I+A}{2}\right) X\beta\|_2^2 + \lambda \cdot \operatorname{tr}\left(X^\top LX\right) \\ \text{subject to} \quad & A \in \mathbf{\Omega}_n,\ A = A^\top,\ A_{ii} = 0, \end{aligned} \tag{2.1}$$

where $\mathbf{\Omega}_n$ is the Birkhoff polytope, which has geometry amenable for optimization, and is defined as

$$\mathbf{\Omega}_n = \{A \in \mathbb{R}^{n\times n} \mid A \geq 0, A\mathbf{1} = \mathbf{1}, A^\top \mathbf{1} = \mathbf{1}\}. \tag{2.2}$$

A famous result is that the Birkhoff polytope is the convex hull of all $n \times n$ permutation matrices. The choice of the hyperparameter λ involves a trade-off between smoothness over predictions and smoothness over data.

We should make some statements on the structure of problem (2.1). We can see that the term $\left(\frac{I+A}{2}\right) X$ has the effect of mixing feature information by taking a convex combination over learned neighborhoods, to improve prediction. Averaging with the identity matrix ensures that each row does not 'forget' its own feature information. Another interesting point is the calculation of the trace term, $\operatorname{tr}\left(X^\top LX\right)$. First, since A is doubly stochastic, L is invariant to normalization since the diagonal of the degree matrix has ones always. Second, due to an insight from [2], is that we can use the defined matrix $Z_{ij} = \|x_i - x_j\|_2^2$ to rewrite the trace term as

$$\operatorname{tr}(X^\top LX) = \frac{1}{2}\operatorname{tr}(AZ) = \frac{1}{2}\sum_{i,j} \|A_{ij}Z_{ij}\|_1, \tag{2.3}$$

since both $A_{ij}, Z_{ij} \geq 0$. In words, the smoothness term is a weighted ℓ_1 norm of the adjacency matrix A and encodes *weighted sparsity*, penalizing edges connecting distant rows of X. The interpretation given by the authors is that when the given distances come from a smooth manifold, the corresponding graph has a sparse set of edges, preferring only the ones associated to small distances in Z. Furthermore, rewriting the trace term as a weighted sum (i.e., the last term in (2.3)) is inexpensive to optimize, as it simply an Hadamard product, followed by a sum, which is $O(n^2)$. In contrast performing two matrix-matrix and taking the trace of the resulting matrix is $O(n^3)$.

Ideally we would like our problem to be convex with respect to its inputs; convex optimization problems have the wonderful property that every local minimum is a global minimum, i.e., we can reliably find the solution of a convex optimization problem. Unfortunately our posed problem (2.1) is not jointly-convex with respect to the optimization variables A and β. There are many ways to attack such a problem. For example, we can ignore the lack of convexity and global smoothness, differentiate where possible and apply stochastic gradient descent to the error term. However, it is important to note that although the problem is not jointly-convex with respect to A and β, it is convex with respect to each variable when the other is kept fixed. This inspires a more principled approach, namely where we fix A and solve for β, then fix β while we solve for A, and repeat until convergence. This is the approach we take and is similar to alternating least squares.

Algorithm. Initialize $\hat{A} \leftarrow$ I, and repeat 1. and 2. until convergence

1. $\hat{\beta}$ calculated as solution to

$$\underset{\beta}{\text{minimize}} \quad \|Y - \left(\frac{I + \hat{A}}{2}\right) X\beta\|_2^2,$$

to predict $\hat{Y} = \left(\frac{I+\hat{A}}{2}\right) X\hat{\beta}$.

2. $\hat{A}$ calculated as solution to

$$\begin{aligned} \underset{A}{\text{minimize}} \quad & \|Y - \left(\frac{I + A}{2}\right) \hat{Y}\|_2^2 + \lambda \cdot \operatorname{tr}\left(X^\top L X\right) \\ \text{subject to} \quad & A \in \mathbf{\Omega}_n,\ A = A^\top,\ A_{ii} = 0. \end{aligned}$$

3 Experiments on synthetic data

We detail experimental results for three synthetic tasks.

3.1 Two-block stochastic block model

The stochastic block model (SBM) is a well-established generative model to produce random graphs that exhibit community structure [6]. The SBM has three parameters. The first is the value n, the number of vertices in graph. The second is a partition of the vertices $\{1, 2, \ldots, n\}$ into disjoint sets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r$. The last is a symmetric $r \times r$ matrix P of edge probabilities specifying the likelihood P_{ij} of edges to form between communities $\mathcal{C}_i$ and $\mathcal{C}_j$. For our case, we select $n = 200$ with $r = 2$, and partition into two equisized clusters: $\mathcal{C}_1 = \{1, \ldots, 100\}$ and $\mathcal{C}_2 = \{101, \ldots, 200\}$. Finally, we set the probabilities based on [7]. The work of [7] also provides features on each node. The label on each node is simply the community label.

The task is to learn a matrix A which will be optimal for predictions. Note that the actual graph G generated by the SBM is not one that should necessarily be used in improving predictions, due to the presence of between-community edges. Since the provided features cannot be used for a linear prediction, a method without optimizing for A would not be able to fit this dataset entirely. In this experiment, ideally A should be block diagonal (under permutation of the rows) with equal weight in both blocks, where each block represents a community. Running our algorithm is able to recover this result, and we show the visualization of the adjacency matrix in Figure 1. Note that the result is not a dense matrix, due to the sparsity-inducing trace term. In this case, using the actual graph of the SBM to perform prediction does not give full accuracy, as in problem (1.5), whereas our method does (see Table 1).

Table 1: Comparison between results on stochastic block model experiment.

Regression Model	R^2 score
Ours	0.998
X	0.680
SX	0.752
S^2X	0.855
S^3X	0.458

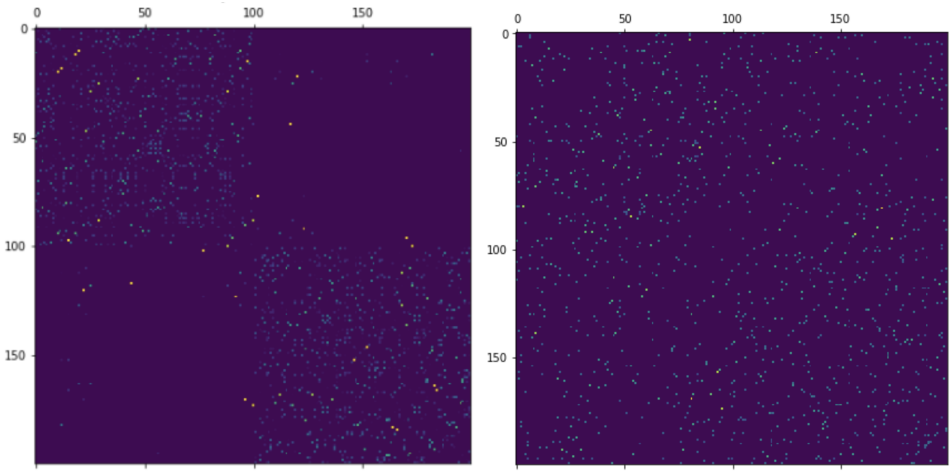

Figure 1: Left: Learned adjacency matrix from two-block stochastic block model experiment, which creates a block diagonal structure. Right: Generated stochastic block model.

3.2 Smooth signals

We also wished to extend our model to a transductive setting, where only some label information is observed (i.e., training labels), and the goal is to predict on the remaining vertices (test samples). This involves modifying the algorithm as follows, where $[\,\cdot\,]_{\text{train}}$ refers to the subset of rows that are training samples.

Transductive algorithm. Initialize $\hat{A} \leftarrow \mathrm{I}$, and repeat 1. and 2. until convergence

1. $\hat{\beta}$ calculated as solution to

$$\underset{\beta}{\text{minimize}} \quad \|[Y - \left(\frac{I+\hat{A}}{2}\right) X]_{\text{train}}\beta\|_2^2,$$

   to predict $\hat{Y} = \left(\frac{I+\hat{A}}{2}\right) X\hat{\beta}$.

2. $\hat{A}$ calculated as solution to

$$\begin{aligned} \underset{A}{\text{minimize}} \quad & \|[Y - \left(\frac{I+A}{2}\right)\hat{Y}]_{\text{train}}\|_2^2 + \lambda \cdot \operatorname{tr}\left(X^\top L X\right) \\ \text{subject to} \quad & A \in \mathbf{\Omega}_n,\ A = A^\top,\ A_{ii} = 0, \end{aligned}$$

   to predict $\left(\frac{I+\hat{A}}{2}\right)\hat{Y}$.

The smooth data is generated as follows: we sample 100 points in the unit square $[0,1]^2$ as vertices for the graph, and connect half the edges which have similar norm, and connect half the edges that have small pairwise distances. The task is to predict the norm of each point, which is a non-linear relation. The feature information is calculated $x_i = h(L_G) = Vh(\Lambda)V^\top z_i$, where $z_i \sim \mathcal{N}(0, \mathrm{I})$ and $L_G = V\Lambda V^\top$ is the eigendecomposition of the Laplacian of the underlying graph in the unit square $[0,1]^2$. With a slight abuse of notation, h is defined as $h(\lambda) = (1+10\lambda)^{-1}$. Because smaller eigenvalues of L_G (which are the smoother 'modes' over the graph, given by the eigenvectors) are made larger by h, and vice versa for the smaller eigenvalues, then the data matrix X is also smooth, i.e., the value $\operatorname{tr}(X^\top L_G X)$ is small for the underlying Laplacian L_G. We use a random 80-10-10 train-validation-test split. The results are listed in Table 2. The visualization is shown in Figure 2.

Table 2: Comparison between results on predicting norm in smooth signal experiment.

Regression Model	Train R^2 score	Test R^2 score
Ours	0.996	0.947
X	0.959	0.330
SX	0.965	0.691
S^2X	0.950	0.830

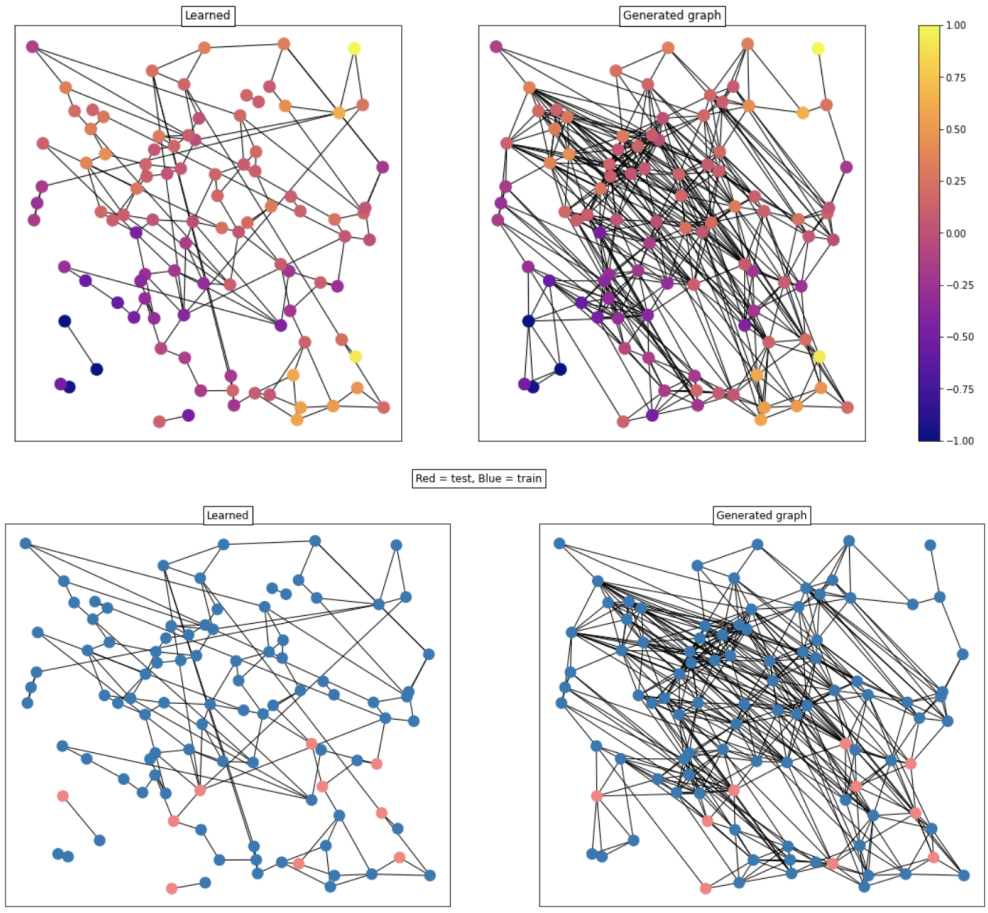

Figure 2: Visualization for the norm prediction experiment of the generated graph (right) and the learned graph (left).

4 Experiments on real-world data: Facebook100

The final set of experiments is on real-world data. They are from the Facebook social network in 2005, where the class is non-homophilous, namely the prediction of gender. The features are attributes such as gender, major, dorm, class year, and university role. We individual Facebook100 universities as datasets. We apply our method as in the previous setting, with 80-10-10 train-validation-test split. On these datasets, $S^k X$ for $k \in \{1, 2, 3\}$ performs very poorly, so the results are omitted. The results are taken over 20 random splits and are listed in Table 3.

Table 3: Comparison of results on Facebook100 datasets.

Dataset	Model	Train R^2 score	Test R^2 score
Amherst41	Ours	0.422 ± 0.12	0.087 ± 0.04
	X	0.705 ± 0.01	-0.172 ± 0.09
	X with ℓ_2 regularization	0.362 ± 0.01	0.097 ± 0.05
Caltech36	Ours	0.313 ± 0.10	0.082 ± 0.07
	X	0.829 ± 0.02	-0.653 ± 0.36
	X with ℓ_2 regularization	0.280 ± 0.10	0.087 ± 0.07
Haverford76	Ours	0.311 ± 0.13	0.044 ± 0.04
	X	0.782 ± 0.01	-0.553 ± 0.24
	X with ℓ_2 regularization	0.268 ± 0.12	0.044 ± 0.04
Reed98	Ours	0.294 ± 0.15	0.048 ± 0.04
	X	0.837 ± 0.01	-0.920 ± 0.35
	X with ℓ_2 regularization	0.257 ± 0.13	0.048 ± 0.05

5 Future work

There are several promising directions for future work that builds on the report. One is motivated by the observation that the alternating least squares approach for optimization seems suboptimal, especially when compared to the ridge regression results in Section 4. In practice, we have seen this may happen since the alternating least squares optimization may produce an A that completely switches two features that it should not, thereby serving as a bad surrogate for the next beta iterate, obtained by solving the associated least squares problem with this surrogate A. Switching this optimization to be less rigid and factor dependent may be the key to obtaining better results; in particular, we suggest that using alternating steps of projected gradient descent may produce a better optimizer. However, this also comes with its own set of problems. In particular, projection to the Birkhoff polytope of the matrix iterate A will have to occur, and this projection is not a trivial operation with a single unambiguous implementation.

6 Conclusion

In this report, we proposed a novel framework for learning a graph topology to improve prediction. We showed several promising results on both synthetic and real datasets. Future work may consist of improving the optimization method so as to make the iterates less affected by intermittent inaccuracies with other iterates throughout the process of alternating least squares.

References

[1] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

[2] Vassilis Kalofolias. How to learn a graph from smooth signals. In *AISTATS*, 2016.

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[4] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019.

[5] Byandreas Buja, Trevor J. Hastie, and Robert Tibshirani. Linear smoothers and additive models. *Annals of Statistics*, 17:453–510, 1989.

[6] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[7] Yash Deshpande, Andrea Montanari, Elchanan Mossel, and Subhabrata Sen. Contextual stochastic block models. In *NeurIPS*, 2018.