Graph Sparsification Approaches for Laplacian Smoothing

Veeranjaneyulu Sadhanala*, **Yu-Xiang Wang***, **Alexander Smola**
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213
{vsadhana,yuxiangw}@cs.cmu.edu, alex@smola.org

Ryan Tibshirani
Statistics & Machine Learning Departments,
Carnegie Mellon University, Pittsburgh, PA 15213
ryantibs@cmu.edu

Abstract

Given a statistical estimation problem where regularization is performed according to the structure of a large, dense graph G, we consider fitting the statistical estimate using a *sparsified* surrogate graph $\tilde{G}$, which shares the vertices of G but has far fewer edges, and is thus more tractable to work with computationally. We examine three types of sparsification: spectral sparsification, which can be seen as the result of sampling edges from the graph with probabilities proportional to their effective resistances, and two simpler sparsifiers, which sample edges uniformly from the graph, either globally or locally. We provide strong theoretical and experimental results, demonstrating that sparsification before estimation gives statistically sensible solutions, with potentially huge computational savings.

1 Introduction

We study efficient computation in large-scale graph-based regression and classification problems. To fix notation, let $G = (V, E, w)$ denote an undirected graph, with vertex set $V = \{1, \dots n\}$, edge set $E \subseteq \{(i, j) : i, j \in V\}$ of size $m = |E|$, and weights w_{ij}, $(i, j) \in E$. Recall that the Laplacian matrix $L \in \mathbb{R}^{n \times n}$ of G is

$$L_{ij} = \begin{cases} \sum_{(i,\ell) \in E} w_{i\ell} & \text{if } i = j \\ -w_{ij} & \text{if } i \neq j \end{cases}, \quad i, j \in V.$$

Given observations $y = (y_1, \dots y_n) \in \mathcal{Y}^n$ over nodes of G, we consider a family of Laplacian-regularized estimation problems

$$\min_{\beta \in \mathbb{R}^n} \; f(\beta) + \lambda \beta^T L \beta, \tag{1}$$

where f is a smooth, convex loss, and $\lambda \geq 0$ a tuning parameter. Of particular interest are regression and classification problems, corresponding to real-valued observations, $\mathcal{Y} = \mathbb{R}$, and (say) binary observations, $\mathcal{Y} = \{0, 1\}$, respectively. We may view the components of $y = (y_1, \dots y_n)$ as independent draws from a model with parameters $\beta^* = (\beta_1^*, \dots \beta_n^*)$, satisfying

$$\mathbb{E}(y_i) = \beta_i^*, \; i \in V, \quad \text{OR} \quad \log\left(\frac{\mathbb{P}(y_i = 1)}{\mathbb{P}(y_i = 0)}\right) = \beta_i^*, \; i \in V,$$

for the regression and classification settings, respectively. Accordingly, the natural choices of loss functions in these two settings are, respectively,

$$f(\beta) = \|y - \beta\|_2^2, \quad \text{OR} \tag{2}$$

$$f(\beta) = \sum_{i=1}^n [-y_i \beta_i + \log(1 + \exp(\beta_i))]. \tag{3}$$

*The first two authors contributed equally.

It is easy to see that that the penalty term in (1) is $\beta^T L\beta = \sum_{(i,j)\in E} w_{ij}(\beta_i - \beta_j)^2$, and therefore the solution $\hat{\beta}$ in this problem will have components that vary smoothly over adjacent nodes in the graph G, with a larger value of λ translating to a higher degree of smoothness. Of course, an implicit assumption in using such an estimate $\hat{\beta}$ for β^* is that the latter is itself smooth over G. Some definitive references on Laplacian-based methods are [2, 13, 19, 18, 4, 3].

1.1 Computational Considerations

In the regression case (2), the solution in (1) is simple,

$$\hat{\beta} = (I + \lambda L)^{-1} y, \tag{4}$$

which requires solving a linear system in the symmetric diagonally dominant (SDD) matrix $I + \lambda L$. Significant recent advances have been made in solving SDD linear systems (e.g., [15, 17, 9, 10, 11, 7, 12, 5]); but when G is large and has many edges, i.e., L is large and dense, this can still be a highly nontrivial computational problem. For the classification setting (3), the problem (1) does not admit a closed-form solution, but Newton's method essentially reduces (1) to an iterative sequence of SDD linear systems; each iteration here can be challenging when L is large and dense.

1.2 Sparsfication Before Laplace Smoothing

The motivating question for this paper is as follows: given a large graph G with many edges, and a desired Laplacian-regularized estimate defined by (1), can G be *sparsified* before we compute this estimate? That is, can we instead solve

$$\min_{\beta\in\mathbb{R}^n} \; f(\beta) + \lambda' \beta^T \tilde{L} \beta, \tag{5}$$

for a sparse matrix $\tilde{L}$—the Laplacian corresponding to a graph $\tilde{G}$ that approximates G, but has far fewer edges? Our work answers this in the affirmative. We examine sparsification tactics that can provide drastic computational speedups, and still result in Laplacian-regularized solutions (5) that are provably close to the original solutions (1) from the full graph G.

1.3 Summary of Results

Here is a summary of our contributions.

- When $\tilde{L}$ is built from a spectral sparsifier, we derive strong stability bounds between the solutions of (5) and (1).
- When L is built from the uniform and kN sampling strategies, we derive multiplicative bounds between quadratic forms in the original and sparsified Laplacians that hold with high probability, but only over smooth inputs (not uniformly).
- We give convincing experiments to demonstrate the stability of solutions from spectral sparsfication across all problems, and of solutions from uniform and kN sampling across several real problems where the desired estimates are smooth.
- We show that spectral sparsifiers, and even more so, the uniform and kN sparsifiers, lead to drastic computational speedups in practice.
- We present a number of useful extensions, including spectral sparsification for Cartesian product graphs, normalized Laplacians, and kernels.

2 Stability Bounds for Spectrally sparsified graphs

One of the primary tools we investigate for producing sparse approximations to graphs is *spectral sparsification*, an area that is relatively young, but has generated immense interest in the theoretical computer science community (e.g., [15, 16, 17, 14, 1, 8]). Two Laplacians matrices $L, \tilde{L}$ are said to be σ *spectrally similar* if

$$\frac{1}{\sigma} x^T \tilde{L} x \le x^T L x \le \sigma \cdot x^T \tilde{L} x \quad \text{for all } x. \tag{6}$$

Remarkably, [15, 16, 17] proved that *any graph* G with n nodes and Laplacian matrix L has an approximator $\tilde{G}$ with n nodes and Laplacian $\tilde{L}$ such that $L, \tilde{L}$ are $(1+\epsilon)$ spectrally similar, and $\tilde{G}$ has nearly $O(n/\epsilon^2)$ edges. [14] made this idea more concrete by giving a simple algorithm for producing a $(1+\epsilon)$ spectral approximation $\tilde{L}$, whose graph has $O(n \log n/\epsilon^2)$ edges. [8] strengthened this result, giving faster algorithms with the same or better guarantees.

2.1 Stability Bounds For Regression

Suppose that we have a $(1+\epsilon)$ spectral approximation $\tilde{L}$ to the original Laplacian L intended for smoothing. The notion of spectral similarity yields a strong stability bound between the two solutions of Laplacian-regularized regression problems, given next.

Theorem 1. *Assume that the Laplacian matrices $L, \tilde{L}$ are $(1+\epsilon)$ spectrally similar. Let $\hat{\beta}, \hat{\theta}$ denote solutions in problems* (1)*,* (5)*, respectively, under the regression loss* (2)*. The following bounds hold, for any $\lambda \geq 0$.*

(a) If $\lambda' = \lambda$, then $\|\hat{\theta} - \hat{\beta}\|_2^2 \leq \lambda(1+2\epsilon)|\hat{\theta}^T \tilde{L} \hat{\theta} - \hat{\beta}^T L \hat{\beta}| + 3\lambda\epsilon \hat{\beta}^T L \hat{\beta}$;

(b) If $\lambda' = 2\lambda$, then $\|\hat{\theta} - \hat{\beta}\|_2^2 \leq 2\lambda(1+2\epsilon)\hat{\beta}^T L \hat{\beta}$.

Remark 1. Both bounds in the theorem are non-asymptotic; they assume nothing about the distribution of observations y in the regression problem, and nothing about the original graph G to be sparsified. They are derived purely from the spectral property (6) relating $L, \tilde{L}$, and from optimality characterizations in the problems (1), (5).

Remark 2. The bound in part (a) of the theorem is typically the sharper of the two, since the difference in penalties $|\hat{\theta}^T \tilde{L} \hat{\theta} - \hat{\beta}^T L \hat{\beta}|$ is typically much smaller than either penalty term individually. But it may be somewhat unsatisfactory for theoretical analysis, as it depends on the solution $\hat{\theta}$ of the sparsified problem itself, whose properties are in question. Part (b) eliminates this dependence by over-regularizing the sparsified problem (choosing $\lambda' > \lambda$), delivering a weaker but more transparent final bound.

Similar to Theorem 1 on regression, stability bounds are possible between the solutions of spectrally similar Laplacian-regularized classification problems. But we skip this result for lack of space.

3 Simple Alternative Samplers

We consider two computationally cheap sparsification methods which satisfy the spectral property (6), but only with high probability, at points $x \in \mathbb{R}^n$ that have smooth entries. Recalling that spectral sparsification can be achieved by Effective Resistance(ER) sampling[14], our two proposals also use a sampling paradigm, but they rely on very simple schemes that avoid computing expensive quantities like ERs. They are easily explained as follows.

- **Uniform sparsifier:** sample q edges uniformly from G with replacement, and build $\tilde{G}$ from the sampled edges, with an edge weight $\tilde{w}_e = m w_e / q$ for each new edge e.
- **k-neighbors (kN) sparsifier:** at each node i, select $\min\{k, d_i\}$ edges, with d_i being its degree, in the following manner. If $d_i \leq k$, then add all edges to $\tilde{G}$ that are incident to i, with half of their original edge weights. If $d_i > k$, then sample k edges uniformly with replacement from the neighbors of i, and add them to $\tilde{G}$, with an edge weight $\tilde{w}_e = d_i w_e / (2k)$ for each new edge e.

Note that both schemes use independent sampling, and if an edge is selected more than once, its weight is simply summed up appropriately in $\tilde{G}$. It is clear that computing the surrogate graph $\tilde{G}$ with either of these methods is highly efficient, requiring only $O(m)$ time. Further, it is very likely that sampling without replacement is going to give better results by reducing the variance(say as in Theorem 4 in [6]), but we do not do that to simplify our analysis.

3.1 Restricted Spectral-type Properties

The experiments in the coming sections will show that the graphs obtained by these simple sparsifiers often lead to stable solutions in (5), when compared to (1). Though the uniform and kN sparsifiers do not lead to rigorous worst-case stability bounds as in Theorem 1, they do give rise to a type of restricted spectral property, occurring with high probability.

First, we remark that both samplers deliver an unbiased estimate of the Laplacian quadratic form $x^T L x$, at any $x \in \mathbb{R}^n$. We show that a multiplicative bound holds as in (6), when $\tilde{L}$ is built from a uniform or kN sampler, but this bound only holds for points $x \in \mathbb{R}^n$ whose components are smooth over the original graph.

Theorem 2. *Let $D \in \mathbb{R}^{m \times n}$ denote the weighted incidence matrix of G (i.e., $L = D^T D$). Assume that for some $s > 0$, the point $x \in \mathbb{R}^n$ satisfies*

$$\|Dx\|_\infty^2 \leq s \frac{\|Dx\|_2^2}{m}.$$

Then for $\delta > 0$, the Laplacian $\tilde{L}^{\mathrm{unif}}$ from uniform sampling with $q \geq s^2 \log(2n)/(2\delta^2)$ samples satisfies

$$(1-\delta) x^T L x \leq x^T \tilde{L}^{\mathrm{unif}} x \leq (1+\delta) x^T L x, \tag{7}$$

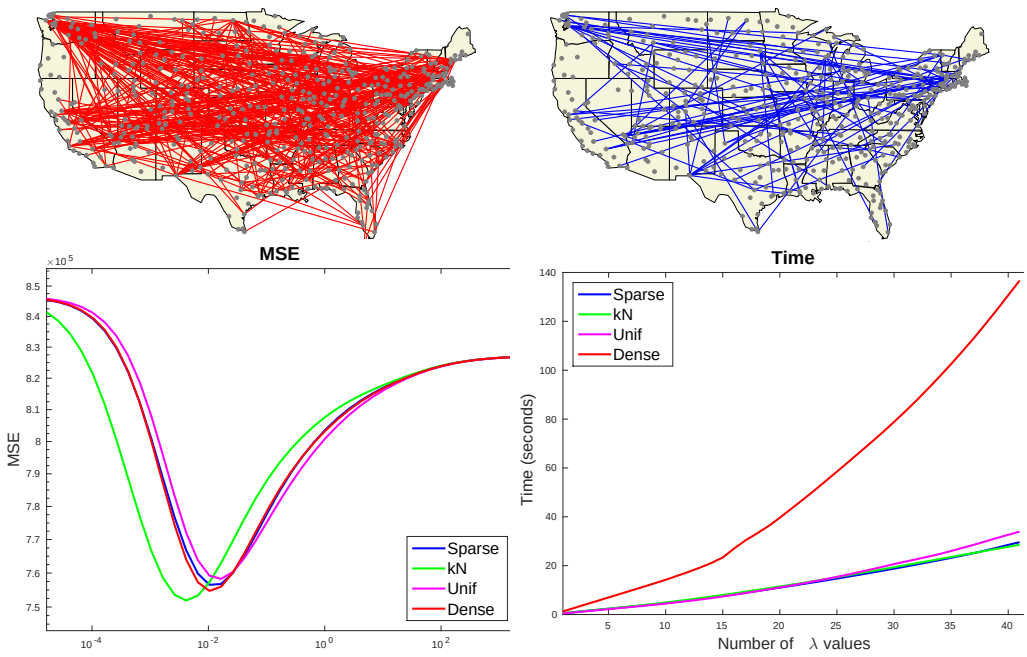

Figure 1: *(a) (Top two) Dense and spectrally sparsified graphs over major US cities (each has been downsampled to 10% edges) and (b)(Bottom) MSE and timing results for the US flu trends data set.*

with probability at least $1 - 1/n$*. Moreover, the Laplacian matrix* $\tilde{L}^{\mathrm{kN}}$ *from kN sampling with* $k^2 \geq s^2 d_{\max} \log(2n)/(n\delta^2)$ *satisfies the same result in* (7)*.*

4 Extensions

We provide two extensions to the work in the previous sections. In the first we show that to sparsify a Cartesian product of two graphs, it is sufficient to sparsify the two graphs separately. This is easy to show but can have huge computational savings, as demonstrated in the experiments in Section 5.

The second extension is to diffusion kernels. In particular, if L_N is the normalized Laplacian of a graph and $\tilde{L}_N$ is a $(1+\epsilon)$ spectral sparsifier of L_N then defining the kernels $K = \exp(-\sigma^2 L_N/2)$, $\tilde{K} = \exp(-\sigma^2 \tilde{L}_N/2)$ for a constant $\sigma > 0$, we show that

$$\|K - \tilde{K}\|_2 \leq \epsilon \sigma^2 e^{\sigma^2}.$$

5 Flu trends across major US cities

We now study flu incidence in major US cities across time. We built a graph from of $n = 791$ cities, and defined edges for each pair of cities that had a flight between them. Edge weights were set according to the log total number of passengers that travelled between cities in the year 2014, made available by the US Department of Transportation.[1] The total number of edges was $m = 9216$. Until recently, Google published city-specific weekly estimates of the percentage of influence-like-illness, based on the volume of related search queries it received.[2] We considered these estimates, called Google Flu Trends (GFT), across a four year span starting in July 2011, for a total of $T = 209$ timepoints. Note that GFT was only available at 74 cities (of the 791 total in our graph).

In order to impute the GFT signal at the remaining 717 cities, we constructed the chain product of the US cities graph described above, of length 209. See Figure 1(a). Leaving out 50% of the GFT observations, chosen at random over cities and timepoints, we fit Laplacian-smoothed regression estimates (1), (2) on the remaining GFT observations. This allowed us to computed MSEs on the left-out data. Figure 1(b) compares these results to those obtained by sparsifying the graph before Laplacian smoothing. It is important to note that the times for sparsification here are extremely cheap, even for the spectral sparsifier, because, as described in Section 4, only the US cities graph (not the full chain product graph over space and time) needs to be sparsified. Further, all sparsifiers deliver sound performance in terms of the MSEs of the subsequent solutions.

[1] `http://www.rita.dot.gov/bts/`

[2] `http://www.google.org/flutrends`

References

[1] Batson, J., Spielman, D. and Srivastava, N. [2008], 'Twice-Ramanujan sparsifiers', *Proceedings of the ACM Annual Symposium on Theory of Computing* **40**, 255–262.

[2] Belkin, M. and Niyogi, P. [2002], 'Using manifold structure for partially labelled classification', *Advances in Neural Information Processing Systems* **15**.

[3] Belkin, M. and Niyogi, P. [2005], 'Towards a theoretical foundation for Laplacian-based manifold methods', *Proceedings of the Annual Conference on Learning Theory* **18**.

[4] Belkin, M., Niyogi, P. and Sindhwani, V. [2005], 'On manifold regularization', *International Conference on Artificial Intelligence and Statistics* **8**.

[5] Cohen, M., Kyng, R., Miller, G., Pachocki, J., Peng, R., Rao, A. and Xu, S. C. [2014], 'Solving SDD linear systems in nearly $m \log^{1/2} n$ time', *Proceedings of the ACM Annual Symposium on Theory of Computing* **46**, 343–352.

[6] Hoeffding, W. [1963], 'Probability inequalities for sums of bounded random variables', *Journal of the American Statistical Association* **58**(301), 13–30.

[7] Kelner, J., Orecchia, L., Sidford, A. and Zhu, Z. A. [2013], 'A simple, combinatorial algorithm for solving SDD systems in nearly-linear time', *Proceedings of the ACM Annual Symposium on Theory of Computing* **45**, 911–920.

[8] Koutis, I., Levin, A. and Peng, R. [2012], 'Improved spectral sparsification and numerical algorithms for SDD matrices', *International Symposium on Theoretical Aspects of Computer Science* **29**, 266–277.

[9] Koutis, I., Miller, G. and Peng, R. [2010], 'Approaching optimality for solving SDD linear systems', *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science* **51**, 235–244.

[10] Koutis, I., Miller, G. and Peng, R. [2011], 'A nearly-$m \log n$ time solver for SDD linear systems', *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science* **52**, 590–598.

[11] Koutis, I., Miller, G. and Peng, R. [2012], 'A fast solver for a class of linear systems', *Communications of the ACM* **55**(10), 99–107.

[12] Lee, Y. T. and Sidford, A. [2013], 'Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems', *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science* **54**, 147–156.

[13] Smola, A. and Kondor, R. [2003], 'Kernels and regularization on graphs', *Proceedings of the Annual Conference on Learning Theory* **16**.

[14] Spielman, D. and Srivastava, N. [2008], 'Graph sparsification by effective resistances', *Proceedings of the ACM Annual Symposium on Theory of Computing* **40**, 563–568.

[15] Spielman, D. and Teng, S.-H. [2004], 'Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems', *Proceedings of the ACM Annual Symposium on Theory of Computing* **36**, 81–90.

[16] Spielman, D. and Teng, S.-H. [2011], 'Spectral sparsification of graphs', *SIAM Journal on Computing* **40**(4), 981–1025.

[17] Spielman, D. and Teng, S.-H. [2014], 'Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems', *SIAM Journal on Matrix Analysis and Applications* **35**(5), 835–885.

[18] Zhou, D., Huang, J. and Scholkopf, B. [2005], 'Learning from labeled and unlabeled data on a directed graph', *Proceedings of the International Conference on Machine Learning* **22**.

[19] Zhu, X., Ghahramani, Z. and Lafferty, J. [2003], 'Semi-supervised learning using Gaussian fields and harmonic functions', *Proceedings of the International Conference on Machine Learning* **20**.