Graph Sparsification Approaches for Laplacian Smoothing

Veeranjaneyulu Sadhanala[1] **Yu-Xiang Wang**[1,2] **Ryan J. Tibshirani**[1,2]
[1] Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Department of Statistics, Carnegie Mellon University, Pittsburgh, PA 15213

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 can give statistically sensible solutions, with significant computational savings.

1 INTRODUCTION

We study efficient computation in large-scale graph-based Gaussian and logistic smoothing problems. To fix notation, let $G = (V, E, w)$ be 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 $\lambda \geq 0$ is a tuning parameter. Of particular interest are the settings with real-valued observations, $\mathcal{Y} = \mathbb{R}$, and (say) binary observations $\mathcal{Y} = \{0, 1\}$. To set up the Gaussian and logistic smoothing problems for these two settings, we view $y = (y_1, \dots y_n)$ as independent draws from a model with parameters $\beta^* = (\beta_1^*, \dots \beta_n^*)$, satisfying

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

respectively, and accordingly the natural choices of loss functions are

$$f(\beta) = \|y - \beta\|_2^2, \quad \text{or} \tag{2}$$
$$f(\beta) = \sum_{i=1}^n [-y_i \beta_i + \log(1 + \exp(\beta_i))], \tag{3}$$

respectively. It is easy to see that that the penalty term in (1) is $\beta^T L \beta = \sum_{(i,j) \in E} w_{ij} (\beta_i - \beta_j)^2$, hence 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 [4, 18, 28, 27, 6, 5].

1.1 Computational Considerations

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

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

the solution of a linear system in a symmetric diagonally dominant (SDD) matrix $I + \lambda L$. Many recent advances have been made in solving SDD linear systems (e.g., [20, 22, 12, 13, 14, 9, 15, 8]); 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 logistic 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, and again, each iteration here can be challenging when L is large and dense.

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

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 Two Illustrative Examples

We consider three types of sparsification in our work: (i) spectral graph sparsification, (ii) uniform sampling of edges from G, and (iii) k-neighbors (kN) sampling of edges (basically a local uniform sampler). The high-level messages are: the spectral sparsifier is the most costly, but leads to a sparse graph and surrogate solution that are *always close* to G and the original solution; the uniform and kN samplers are faster, but do not have the same universal guarantees; the uniform sampler can fail for imbalanced graphs; and lastly, the kN sampler is practically robust across various settings and performs just as well as spectral sparsification.

In Figure 1, left and middle panels, we plot the mean squared errors (MSEs, defined as $\|\hat{\beta} - \beta^*\|_2^2/n$) and computation times for a simulated Gaussian smoothing problem (1), (2), across 100 values of λ and 50 repetitions. The graph G was a synthetic scale-free graph with $n = 1000$ nodes and $m = 130,001$ edges, generated using the Barbasi-Albert preferential attachment model [1]. The underlying signal β^* was taken to the sum of the 20 (unit norm) eigenvectors of the graph Laplacian L corresponding to the smallest eigenvalues (these bottom 20 eigenvalues have components that are relatively smooth over the graph). The observed signal y was obtained by adding mean zero Gaussian noise to β^*, componentwise, with variance equal to the sample variance of the components of β^*.

We compare directly solving (1) with sparsifying first and solving (5). The middle panel in the figure shows the cumulative computation time, counting the time to first construct a sparsified Laplacian $\tilde{L}$ (done only once) and then to solve some number of problems (1), (2): here, the first 100 problems are given by solving (1), (2) for a single y and 100 values of λ, the next 100 are given by solving (1), (2) for a second simulated y and the same 100 values of λ, and so on. We see that the spectral sparsifier takes a nontrivial amount of time to construct $\tilde{L}$, but then leads to more efficient solutions, as the spectrally sparsified graph has only 7780 edges. The two alternative sparsifiers, based on uniform and kN sampling, are much more efficient at constructing $\tilde{L}$, and lead to just as efficient solutions, as their graphs again have about 7800 edges. The left panel in the figure shows that the estimates following all three sparsifiers have MSE curves, averaged over 50 repetitions, that are very close to that of the original solution computed over the original graph.

In the right panel of Figure 1, we plot the MSEs for another simulation setup using a semi-synthetic graph G with $n = 1582$ nodes and $m = 321,661$ edges. This graph was constructed using the airport graph from Section 6, with 791 nodes and 9216 edges, and connecting it by a single edge to a complete graph over 791 nodes, therefore giving a final graph with a highly imbalanced "dumbbell" structure. The underlying signal β^* was defined in two parts: over the airport graph, its components were defined by summing of the bottom 20 eigenvectors of the airport graph Laplacian matrix, and over the complete graph, its components were set to 0. The observed signal y was generated around β^* with componentwise Gaussian noise over the airport graph (and no noise over the complete graph).

Though the computation times for the three sparsifiers are very similar to those from the scale-free simulation (and thus are not shown), the MSE curves are quite different. The key difference is that the solutions obtained using the uniformly sparsified graph do not provide a competitive MSE performance, as the uniform sampler fails to mimick the structure of the original dense graph—it samples too many edges from the complete graph part, and not enough from the airport graph part. The kN sparsifier succeeds by sampling at a more localized level, and its best MSE matches that achieved in the dense case, which is also true of the spectral sparsifier. As described later, kN sampling is close to effective resistance sampling for many types of graphs, and therefore produces sparse graphs similar to that from the spectral sparsification method. The take-away point is that uniform sampling can fail when the graph at hand is imbalanced and/or the underlying signal is not uniformly smooth across edges, but kN sampling behaves favorably across nearly all settings, even imbalanced and/or nonsmooth ones.

1.4 Summary of Results

Here is a summary of our contributions.

- When $\tilde{L}$ comes from a spectral sparsifier, we derive strong stability bounds between the solutions of (5) and (1).

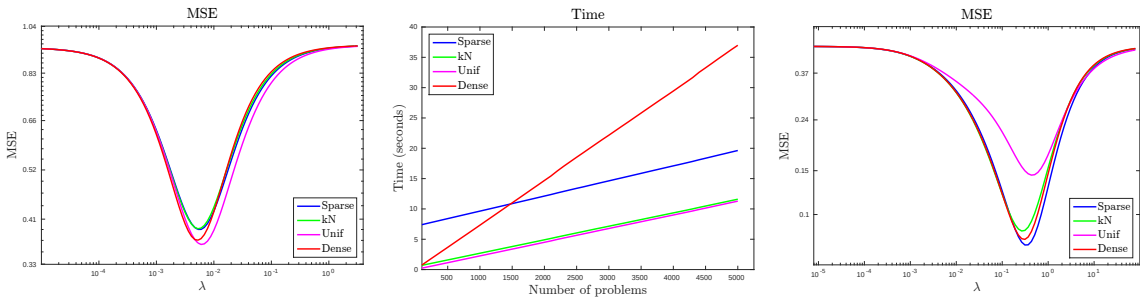

Figure 1: *Left: MSE curves from the scale-free graph simulation; middle: timing curves from this same simulation; right: MSE curves from the dumbbell graph simulation (timing curves from this simulation are similar and hence not shown).*

- When $\tilde{L}$ 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 universally).
- We give experiments that demonstrate the stability of solutions from spectral and kN sparsification across all problems considered, and stability of solutions from uniform sparsification in several 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 extensions on sparsification for Cartesian product graphs, missing data problems, and regression problems (with predictor variables).

2 SPECTRAL SPARSIFICATION

2.1 Review: Spectral Sparsifiers

Spectral sparsification is a relatively young area, but one that has generated immense interest in the theoretical computer science community (e.g., [20, 21, 22, 19, 2, 11]). Here is an overview. Two Laplacians matrices $L, \tilde{L}$ are said to be σ *spectrally similar* if

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

The nature of this relationship is clearly symmetric. Still, for emphasis, we will call $\tilde{L}$ a σ *spectral approximation* to L when (6) holds. Also, denoting by $G, \tilde{G}$ the underlying graphs of $L, \tilde{L}$, we will also use the same terminology (spectral similarity, approximation) for these graphs. As the Laplacian of a graph reflects its edge structure, the above property—a uniform multiplicative bound between Laplacian quadratic forms—provides a very strong tie between $G, \tilde{G}$. E.g., it implies that all eigenvalues of $L, \tilde{L}$ are within a multiplicative factor of $1/\sigma$ to σ, and that all cuts between $G, \tilde{G}$ obey the same multiplicative bounds as well.

Remarkably, [20, 21, 22] proved that *any graph* G with n nodes and Laplacian matrix L has an approximation $\tilde{G}$ 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. [19] 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. [11] strengthened this result, giving faster algorithms with the same or better guarantees. More will be said about computation in Section 2.4.

Spectral graph sparsification has been successfully applied in numerous areas, e.g., in solving Laplacian and SDD linear systems, approximating electrical flows over a graph, computing max flows and min cuts, and performing basic missing data imputation over a graph (for references, see, e.g., the nice review article [3]). To the best of our knowledge, its applications in machine learning have not yet been thoroughly pursued.

2.2 Stability Bounds For Gaussian smoothing

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

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 Gaussian 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

$$\|\hat{\theta} - \hat{\beta}\|_2^2 \leq 2\lambda(1+2\epsilon)\hat{\beta}^T L \hat{\beta}.$$

The proof (and all proofs in this paper) is in the supplementary material. Several remarks are given below.

Remark 1. Both bounds above are non-asymptotic; they assume nothing about the distribution of observations y in the smoothing problem, and nothing about the original graph G to be sparsified. They are derived

using only the spectral property (6) relating $L, \tilde{L}$, and 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. Figure 2 shows the average squared ℓ_2 norm $\|\hat{\beta} - \hat{\theta}\|_2^2/n$ between solutions $\hat{\beta}, \hat{\theta}$, and the two bounds from the theorem (on the correct scale, i.e., divided by n), for the scale-free graph example in Section 1.3, as functions of λ.

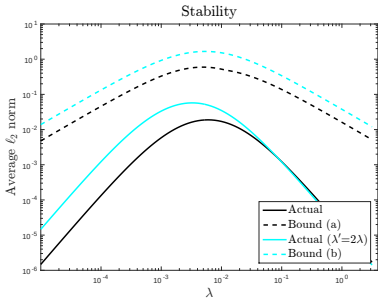

Figure 2: *Average squared ℓ_2 metric between solutions for the scale-free graph example.*

Note that the bound in part (a) 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, delivering a weaker but more transparent final bound.

Remark 3. Though it is the weaker of the two, statistically speaking, the bound in part (b) is not actually weak in absolute terms. The standard statistical analysis for regularized M-estimators (e.g., [25, 17, 7]), applied to the problem (1) with the Gaussian loss (2), prescribes a choice of λ that yields a bounded penalty term as n increases. Such a choice dominates the "empirical process term", in the language of [25], and depends on the properties of the graph G, resulting in an estimation error rate

$$\frac{\|\hat{\beta} - \beta^*\|_2^2}{n} = O_{\mathbb{P}}\Big(\frac{\lambda}{n}\Big). \tag{7}$$

As the bounds in Theorem 1 hold for any value of λ, part (b) implies that if the solution $\hat{\beta}$ over the original graph achieves an error rate as in (7), per the standard analysis, then so must $\hat{\theta}$, the solution over the sparse graph. This follows by simply using $\hat{\beta}^T L \hat{\beta} = O_{\mathbb{P}}(1)$, and the triangle inequality. In other words, under the same conditions that are required in order for the original solution $\hat{\beta}$ to perform well, theoretically, we can expect the sparsified solution $\hat{\theta}$ to perform just as well, in terms of error rate. A formal analysis of an error bound of the form (7) for $\hat{\beta}$, and a proof that $\hat{\theta}$ obtains the same rate, are given in the supplement.

2.3 Stability Bounds for Logistic Smoothing

Similar to Theorem 1 on the Gaussian loss, stability bounds are possible between the solutions of spectrally similar logistic smoothing problems. The next theorem uses the notation $\pi(b) = 1/(1+e^{-b})$ for the inverse logistic link (i.e., sigmoid) function.

Theorem 2. *Assume that the Laplacian matrices $L, \tilde{L}$ are $(1+\epsilon)$ spectrally similar. Consider the logistic loss (3), and let $\hat{\beta}$ denote the solution in (1). Let $\delta > 0$ be such that $2\delta \leq \pi(\hat{\beta}_i) \leq 1 - 2\delta$, $i = 1, \ldots n$. Consider placing the constraint $\delta \leq \pi(\theta_i) \leq 1 - \delta$, $i = 1, \ldots n$ on the problem (5), and let $\hat{\theta}$ denote the corresponding solution. Then the same bounds hold as in parts (a) and (b) of Theorem 1, except with the right-hand sides multiplied by a constant $C = 2/(\delta(1-\delta))$.*

Remark 4. We added the constraint $\delta \leq \pi(\theta_i) \leq 1 - \delta$, $i = 1, \ldots n$ to problem (5) for technical reasons, and often in practice this constraint is not tight. Note that this is equivalent to a (convex) ℓ_∞ norm bound on θ. An alternative would be to add a ridge penalty to the criterion in (1), which would make it strongly convex over its entire domain; see Theorem 3.

Remark 5. The above results are non-asymptotic as written, but an asymptotic interpretation makes more sense, given the large value of the constant C multiplying the bounds. The theorem says that, for a sequence of logistic smoothing problems where the fitted probabilities are not degenerate (are bounded away from 0 and 1), the solutions from the original and sparse graphs are $O(\lambda)$ apart in the squared ℓ_2 metric. This is statistically favorable, because the estimation error for the original logistic smoothing problem, following the typical analysis, is of the same order, as in (7).

Remark 6. Analogous results hold for the Poisson loss (e.g., when estimating a smooth density from counts over the graph), and for a general exponential family loss as well. Details are omitted for brevity.

2.4 Computational Analysis

The topic of spectral graph sparsification has mostly been of theoretical interest due to the fact that spectral sparsification algorithms themselves can be costly in practice. [2] proved that for any graph, there exists an $(1+\epsilon)$ spectral sparsifier with only $O(n/\epsilon^2)$ edges, but finding this in practice is not easy. The most commonly used algorithm is to samples $O(n \log n/\epsilon^2)$ edges, with each edge being sample with probability proportional to its effective resistance (ER), which, while costly to compute exactly (based on L^+), can be approximated by solving $O(\log n)$ Laplacian linear systems [19]. This scheme was sped up by [11], who also described even faster methods that require more

Algorithm	Time complexity	Number of edges
[19]	$O(m \log^3 n)$	$O(n \log n/\epsilon^2)$
[11] (A)	$\tilde{O}(m \log^2 n)$	$O(n \log n/\epsilon^2)$
[11] (B)	$\tilde{O}(m \log n)$	$\tilde{O}(n \log^3 n/\epsilon^2)$
[11] (C)	$O(m \log \log n)$	$\tilde{O}(n \log^5 n/\epsilon^2)$

Table 1: *Methods for* $(1+\epsilon)$ *spectral approximations. The* $\tilde{O}(\cdot)$ *notation hides* $\mathrm{poly}(\log \log n)$ *terms.*

edges to compensate for a poorer approximation quality of ERs. See Table 1 for a summary of methods.

An approach that has weaker theoretical guarantees, but works very well in practice, was developed by [10]. This author constructs a spectral sparsifier by iteratively computing weighted spanners (a specific type of subgraph); though the theory requires $O(\log^2 n/\epsilon^2)$ spanners each with $O(n \log n)$ edges to obtain a $(1+\epsilon)$ spectral sparsifier, we find that in practice $O(\log n/\epsilon^2)$ spanners each with $O(n)$ edges is often enough, and in all of the experiments in this paper, we rely on this spanner algorithm for spectral sparsification.

The pertinent question for our paper is: how does the cost of sparsification weigh in, when comparing the costs of problems (5) and (1)? It depends on the loss function f, and the number of solutions in (5), (1) to be found, e.g., solutions across multiple values of the tuning parameter λ (as would be done in cross-validation for choosing λ). In the Gaussian case (2), if we want to compute a Laplacian-smoothed estimate only once, then we may be better off just solving the dense problem (1) directly, given the fast SDD linear system solvers available for (4), as constructing $\tilde{L}$ will likely itself be more costly. E.g., the SDD solver of [8] runs in $\tilde{O}(m\sqrt{\log n})$ time, which is already faster than all but one of the sparsification times in Table 1. But when we wish to solve (1) repeatedly (say, at multiple tuning parameter values), forming $\tilde{L}$ and instead solving (5) repeatedly can lead to tangible savings. A rough calculation can be carried out as follows (ignoring all constants and $\mathrm{poly}(\log \log n)$ terms). Assume that we solve all linear systems with the method of [8]. Computing N Laplacian-smoothed estimates over the original graph, versus computing a spectral sparsification with (say) method (A) of [11] and then N estimates over the sparse graph, costs

$$N m \sqrt{\log n} \quad \text{versus} \quad m \log^2 n + N \frac{n \log n}{\epsilon} \sqrt{\log n}$$

operations, respectively. Hence, spectral sparsification becomes worth it once we solve at least

$$N \approx \frac{m}{m - n \log n/\epsilon} \log^{3/2} n$$

smoothing problems, or, if we approximate the leading factor by 1 (valid for $m \gg n$), at least $N \approx \log^{3/2} n$ smoothing problems. This calculation was very rough, but we have found it to be in loose agreement with our empirical experiments. E.g., as the scale-free graph simulation in Section 1.3 increases in size, the crossing points between the red and blue timing curves in the middle panel of Figure 1 grows slowly with the number of nodes n, roughly logarithmically.

Solving (1) with a logistic loss (3) is a more challenging problem, and the computational tradeoff sits even more strongly in favor of spectral sparsification. Using, e.g., Newton's method on problem (1) means solving SDD linear systems a few dozen times, rather than just once; thus, even when an estimate is desired at a single value of λ, the cost of constructing a spectral approximation $\tilde{L}$ is usually worth it. When solutions are sought over a grid of λ values, sparsification before estimation is usually a clear win, computationally. The examples throughout this paper provide concrete evidence of the computational assertions here, especially those in Sections 4 and 6.

3 ALTERNATIVE SPARSIFIERS

3.1 Simple Alternative Samplers

We examine 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 ER sampling, 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 below.

- **Uniform sparsifier:** we sample q edges from G with probabilities proportional to edge weights, and with replacement. We build $\tilde{G}$ from the sampled edges $\tilde{E}$, with equal edge weights W/q where $W = \sum_{e \in E} w_e$.
- **k-neighbors (kN) sparsifier:** at each node u, we select $\min\{k, d_u\}$ edges, with d_u being its degree, in the following manner. If $d_u \leq k$, then we add all edges to $\tilde{G}$ that are incident to u, with half of their original edge weights. If $d_u > k$, then we sample k edges from the neighbors $N(u)$ of u with probabilities proportional to edge weights, and with replacement. We add them to $\tilde{G}$, with equal edge weights $W_u/(2k)$ where $W_u = \sum_{v \in N(u)} w_{u,v}$.

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 cheap, requiring only $O(m)$ time.[1]

[1] In practice, to ensure connectivity in the sparsified graph from either method, we can start with a random

3.2 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 directly lead to rigorous worst-case stability bounds as in Theorems 1 and 2, they do give rise to a type of restricted spectral property, occurring with high probability.

First, we remark that both samplers deliver an unbiased Laplacian $\tilde{L}$, i.e., with $\mathbb{E}(\tilde{L}) = \mathbb{E}(L)$. It suffices to show that $\mathbb{E}(\tilde{w}_e) = w_e$ for each fixed edge $e \in E$. For uniform sampling, denote by N_e the number of times e is sampled. We have $\mathbb{E}(\tilde{w}_e) = \mathbb{E}(N_e W/q) = w_e$, as $\mathbb{E}(N_e) = qw_e/W$. For kN sampling, denote by $N_{e,u}$ the number of times the edge $e = (u, v)$ is chosen when sampling locally at node u; the contribution to $\tilde{w}_e$ here is $N_{e,u}W_u/2k$ if $d_u > k$ and $w_e/2$ otherwise. As $\mathbb{E}(N_{e,u}) = kw_e/W_u$ if $d_u > k$, the expected contribution to $\tilde{w}_e$ while sampling at node u is $w_e/2$. By symmetry, the same is true at node v, so $\mathbb{E}(\tilde{w}_e) = w_e$. Thus for both methods, $\mathbb{E}(x^T\tilde{L}x) = x^T L x$ for all x.

Next, using this unbiased property and Hoeffding's inequality, we derive multiplicative bounds as in (6), for $\tilde{L}$ constructed by the uniform or kN samplers.

Lemma 1. *Write the Laplacian matrix as $L = D^T D$, where $D \in \mathbb{R}^{m\times n}$ is the edge incidence matrix. Denote $w_{\min} = \min_{e\in E} w_e$ and $W_{\max} = \max_{u\in V} W_u$. Assume that for some $s > 0$, the point x satisfies*

$$\frac{\|Dx\|_\infty^2}{\|Dx\|_2^2} \leq s\frac{w_{\min}}{W}. \tag{8}$$

Then for $\epsilon > 0$, the Laplacian $\tilde{L}^{\text{unif}}$ from uniform sampling with $q \geq s^2(1+1/\epsilon)^2/2 \cdot \log(2n)$ samples satisfies

$$\frac{1}{1+\epsilon}x^T L x \leq x^T \tilde{L}^{\text{unif}} x \leq (1+\epsilon)x^T L x,$$

with probability at least $1-1/n$. The same result holds for the Laplacian $\tilde{L}^{\text{kN}}$ from kN sampling provided that $k \geq n(sW_{\max}/W)^2/(4\delta^2) \cdot \log(2n)$.

Remark 7. The condition (8) is a type of smoothness or *incoherence* condition on $x \in \mathbb{R}^n$, with respect to G. It says that the m-dimensional vector Dx has roughly balanced entries, where we note that $\|Dx\|_2^2 = x^T L x$. This excludes Dx having "spiked" entries, i.e., x having large differences over a small number of edges, but not over most edges.

Remark 8. The bound on k for the kN sampler could be tightened by defining "local" versions of the quantities $w_{\min}, W, s$, though this complicates notation and we do not pursue this. In fact, it is likely that sharper theory could be given for the kN sampler, in which a smoothness condition like (8) is not required. The expected number of times an edge (u, v) is sampled in kN sampling is proportional to $w_{u,v}(1/W_u + 1/W_v)$, assuming all degrees are at least k, and [26] showed that the quantity $1/W_u + 1/W_v$ is a simple but accurate approximation to the ER of the edge (u, v) for several types of graphs. We attribute the good performance of kN sampling to this sound approximation of ERs. This will be pursued further in future work.

spanning tree of G and then proceed with sampling on the remaining edges. Construction of the random spanning tree also is also $O(m)$ time.

4 SIMULATED DIFFUSION OVER GOOGLE+ NETWORK

We demonstrate the benefits of graph sparsfication in Gaussian and logistic Laplacian smoothing problems, using the Google+ social network data set from the Stanford Network Analysis Project [16]. This data set contains 132 ego networks with a list of binary features at each node. We combined the "ego" networks, and worked on a connected subgraph having $n = 49,605$ nodes and $m = 2,999,962$ edges.

We simulated a smoothed signal over the nodes, mimicking information diffusion across the network. We picked 10 starter nodes uniformly at random, and drew $p_i \sim \text{Unif}(0, 0.5)$, $i = 1, \ldots 10$ independently. For each starter node i, we then spawned $m/10$ random walks from this node, which proceed as follows: at each step, toss a weighted coin with success probability $1-p_i$; if a success, move to a neighboring node chosen uniformly at random; if a failure, terminate the random walk. A signal β^* was defined by counting the number of visits at each node in the graph over this entire process, and then normalizing by the average count. In the supplement, we analyze a Gaussian smoothing problem, where i.i.d. Gaussian noise is added the components of β^* to form observations y. Here we examine a logistic smoothing problem, where independent Bernoulli random variables are sampled over the nodes, with probabilities given by $\pi(\beta^* - \bar{\beta}^*)$, where $\bar{\beta}^*$ is the sample mean of β^*, and recall $\pi(b) = e^b/(1+e^b)$ denotes the inverse logistic link (i.e., sigmoid) function.

Figure 3 shows the results from directly fitting the Laplacian-smoothed logistic model (1), (3), and the results from using spectral sparsification, uniform sampling, and kN sampling, and then solving (5). The sparsified graphs for spectral, uniform, and kN methods each produce graphs with about $380,000$ edges, and order of magnitude reduction compared to the dense graph. In terms of misclassification rate (MCR) and mean squared error (MSE), the estimates after spectral and kN sparsification match very closely with those from the original dense graph, while the esti-

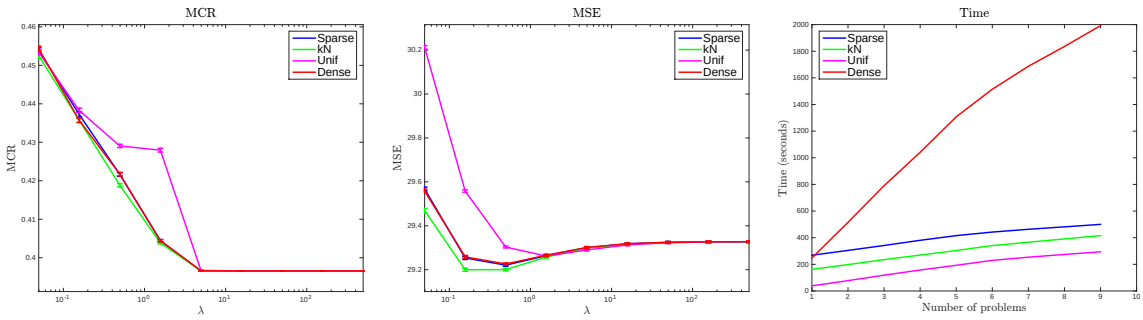

Figure 3: *MCR, MSE, and timing results for a logistic smoothing problem with the Google+ data.*

mates after naive uniform sparsification deviate somewhat. Computationally, all sparsifiers provide a big savings, offering a 5-10 times speedup after 9 smoothing problems (values of λ queried).[2]

5 EXTENSIONS

5.1 Sparsifying Partially Structured Graphs

Here we consider spectral sparsification for partially-structured graphs, in particular, *Cartesian products* of graphs. Recall that for graphs G, H, their Cartesian product $G \times H$ is defined to be a graph with vertices $V = V_G \times V_H$, the (usual) Cartesian product of sets V_G and V_H, and edges

$$((u,v),(u',v')) \in E \iff$$
$$u = u' \text{ and } (v,v) \in E_H, \quad \text{or} \quad v = v' \text{ and } (u,u') \in E_G.$$

The edge weights in $G \times H$ carry over from those in G or H. That is, if $(v,v') \in H$ with edge weight $w_{v,v'}$, then $((u,v),(u',v'))$ is given weight $w_{v,v'}$ in $G \times H$; and similarly for the flip case. The next result shows that, to spectrally sparsify a Cartesian product of two graphs, we must only sparsify each graph individually, and then form their Cartesian product.

Lemma 2. *Let G, H be graphs, and $\tilde{G}, \tilde{H}$ be spectral approximations, respectively, with constants $\sigma_G, \sigma_H > 0$. Then $\tilde{G} \times \tilde{H}$ is a σ spectral approximation of $G \times H$, where $\sigma = \max\{\sigma_G, \sigma_H\}$.*

The fact that spectral approximations can be "decomposed" across Cartesian products is interesting, but in our experience, it is seldom in practice that one encounters a Cartesian product of two dense graphs, each worthy of being sparsified. Far more often, one encounters the Cartesian product of a dense graph G with an already sparse graph H. Lemma 2 also applies to this special case, with $\tilde{H} = H$, and shows that $\tilde{G} \times H$ is a proper spectral approximation of $G \times H$.

[2]The apparent difference in costs for forming the uniform and kN sparsified graphs is not meaningful, it is only due to the inefficiency of looping in MATLAB.

As a prime example, we consider the Cartesian product of an arbitrary graph with a chain graph, a combination that arises naturally in modeling data that is recorded at regular timepoints over the nodes. Let us call $G \times C$, where C is the chain graph with vertices $V = \{1, \dots T\}$, and edges $E = \{(t, t+1) : t = 1, \dots T-1\}$, the *chain product* of G of length T.

Corollary 1. *Suppose that $\tilde{G}$ is a σ spectral approximation of G. Then the chain product of $\tilde{G}$ of length T forms a σ spectral approximation of the chain product of G of length T.*

This corollary has major computational implications, showing that chain product graph with $O(mT)$ edges can be sparsified at the same cost as a single graph with $O(m)$ edges. This is leveraged in Section 6, where we model weekly flu trends across major US cities.

5.2 Missing Data and Regression Problems

In terms of Laplacian regularization, the focus of the machine learning community has been semi-supervised problems, where data is missing at some (or many) nodes of the graph (e.g., [4, 28, 27, 6, 23, 24]).

Another interesting class of problems are regression problems, where predictor variables $x_1, \dots x_n \in \mathbb{R}^p$ are measured along with the responses $y_1, \dots y_n \in \mathcal{Y}$, and we model either the mean $\mathbb{E}(y_i|x_i)$ (in the linear regression case) or the log odds $\log(\mathbb{P}(y_i = 1|x_i)/\mathbb{P}(y_i = 0|x_i))$ (in the logistic regression case) as $x_i^T\beta^*$, where $\beta^* = (\beta_1^*, \dots \beta_p^*)$ is believed to have smoothly varying components over a graph G. Note that this generalizes our setup so far, by taking $p = n$ and $x_i = e_i \in \mathbb{R}^n$ (the ith standard basis vector), for $i = 1, \dots n$.

We describe an extension of our framework to handle both missing data and predictor variables. Let us write the loss function, in the case of fully observed data, as $f(\beta) = \sum_{i=1}^n \ell(x_i^T\beta; y_i)$. With missing data, let $\Omega \subset \{1, \dots n\}$ denote observed set of nodes. Then we solve

$$\min_{\beta \in \mathbb{R}^n} \sum_{i \in \Omega} \ell(x_i^T\beta; y_i) + \lambda\beta^T L\beta + \mu\|\beta - v\|_2^2. \quad (9)$$

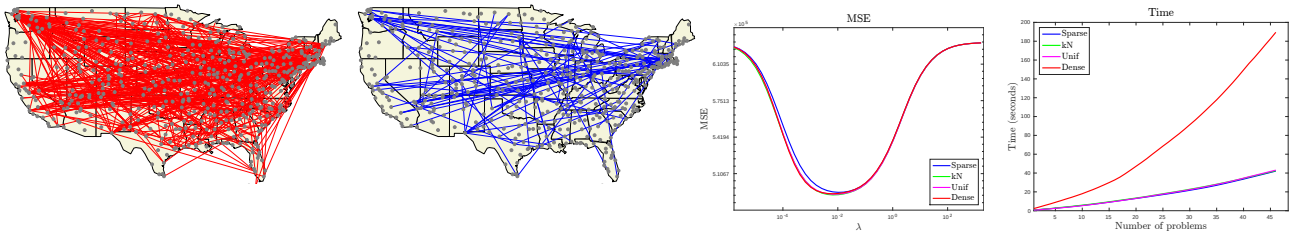

Figure 4: *Left plots: dense and spectrally sparsified airport graphs over major US cities (each has been downsampled to 10% edges, for visibility). Right plots: MSE and timing results for the US flu trends data set.*

In principle, we could interpret $\mu > 0$ above as a second tuning parameter, but for simplicity we take it to be small and fixed. The last term in the criterion in (9) ensures that it is strongly convex, guaranteeing a unique solution in a potentially ill-specified problem. The shrinkage target $v \in \mathbb{R}^n$ is chosen in a simple fashion, e.g., with constant entries $\bar{y} = \frac{1}{|\Omega|}\sum_{i\in\Omega} y_i$ in the Gaussian case, or $\log(\bar{y}/(1-\bar{y}))$ in the logistic case.

Just as before, in cases where L is dense, we propose to approximate L with a sparse matrix $\tilde{L}$, and solve

$$\min_{\beta\in\mathbb{R}^n} \sum_{i\in\Omega} \ell(x_i^T\beta; y_i) + \lambda'\beta^T\tilde{L}\beta + \mu\|\beta - v\|_2^2, \quad (10)$$

instead of (9). Stability bounds relating (9), (10) follow closely from those derived in Sections 2.2 and 2.3.

Theorem 3. *Assume that the Laplacian matrices $L, \tilde{L}$ are $(1+\epsilon)$ spectrally similar. Let $\hat{\beta}, \hat{\theta}$ denote solutions of (9), (10) respectively, where ℓ is an arbitrary convex loss function. Then the same bounds hold as in parts (a) and (b) of Theorem 1, but with the right-hand sides multiplied by a constant $C = 1/\mu$.*

Remark 8. Above, ℓ is a general convex function (not even necessarily smooth), which broadens the scope of previous stability results, and covers, e.g., the hinge loss for binary data. The bounds in Theorem 3 will be weaker than those in Theorem 1 when μ is small (and weaker than those in Theorem 2 when μ is small and the fitted probabilities are far away from 0 and 1).

6 FLU TRENDS IN US CITIES

We now study flu incidence in major US cities across time. We built a graph from of $n = 791$ cities and $m = 9216$ edges, where an edge was defined for each pair of cities that had a regular flight in between them. Edge weights were set according to the log total number of passengers that travelled between cities in the year 2014.[3] Until recently, Google published city-specific weekly estimates of the percentage of influenza-like-illness, based on the volume of related search queries it received. 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.[4]

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 the left two plots of Figure 4. After leaving out 50% of the GFT observations, chosen at random over cities and timepoints, we fit Laplacian-smoothed estimates (1), (2) on the remaining GFT observations. This allowed us to compute MSEs on the left-out data. The right two plots of Figure 4 compares these results to those obtained by sparsifying the graph before Laplacian smoothing. All sparsified graphs have about 30% of the 2,090,672 edges in the original graph. It is important to note that the times for sparsification here are extremely cheap, even for the spectral sparsifier, because, as described in Section 5.1, only the US cities graph (not the full chain product graph over space and time) needs to be sparsified. Furthermore, all sparsifiers deliver sound performance in terms of the MSEs of the subsequent solutions.

7 DISCUSSION

We studied the use of spectral graph sparsification in Laplacian-regularized estimation problems. We gave theory and empirical examples showing that sparsification before Laplacian smoothing can lead to statistically sound estimates and tangible computational savings. We also considered two alternate sparsification schemes, based on uniform and kN sampling of edges. which do not provide the same worst-case guarantees as spectral sparsification, but are much simpler and much cheaper in practice. The kN sampler especially showed consistently strong empirical performance.

Acknowledgements. YW was supported by the Singapore NRF under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. RT was supported by NSF Grant DMS-1309174. We thank Alex Smola for his general help and guidance, and Ioannis Koutis and Shen Chen Xu for help with their sparsifier code.

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

[4] From `http://www.google.org/flutrends/`.

References

[1] Albert, R. and Barabasi, A.-L. [2002], 'Statistical mechanics of complex networks', *Reviews of Modern Physics* **75**, 47–97.

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

[3] Batson, J., Spielman, D., Srivastava, N., and Teng, S.-H. [2013], 'Spectral sparsification of graphs: theory and algorithms', *Communications of the ACM* **56**(8), 87–94.

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

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

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

[7] Buhlmann, P. and van de Geer, S. [2011], *Statistics for High-Dimensional Data*, Springer, Berlin.

[8] 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.

[9] 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.

[10] Koutis, I. [2014], 'Simple parallel and distributed algorithms for spectral graph sparsification', *Proceedings of the ACM symposium on Parallelism in Algorithms and Architectures* **26**, 61–66.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] McAuley, J. and Leskovec, J. [2012], 'Learning to discover social circles in ego networks', *Advances in Neural Information Processing Systems* **25**.

[17] Negahban, S., Ravikumar, P., Wainwright, M. and Yu, B. [2012], 'A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers', **27**(4), 538–557. Statistical Science.

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

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

[20] 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.

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

[22] 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.

[23] Talukdar, P. P. and Crammer, K. [2009], New regularized algorithms for transductive learning, *in* 'Machine Learning and Knowledge Discovery in Databases', Springer, pp. 442–457.

[24] Talukdar, P. P. and Pereira, F. [2010], 'Experiments in graph-based semi-supervised learning methods for class-instance acquisition', *Proceedings of the Annual Meeting of the Association for Computational Linguistics* **48**.

[25] van de Geer, S. [2000], *Empirical Processes in M-Estimation*, Cambridge University Press, Cambridge.

[26] von Luxburg, U., Radl, A. and Hein, M. [2014], 'Hitting and commute times in large random neighborhood graphs', *Journal of Machine Learning Research* **15**, 1751–1798.

[27] 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**.

[28] 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**.