Edge Conductance Estimation with MCMC

Ashish Bora*

Advisors: Prof. Vivek Borkar*, Prof. Rajesh Sundaresan, Dr. Dinesh Garg*****

*** IIT Bombay, **IISc Bangalore, ***IBM Research, Bangalore**

Introduction

What is conductance?

Imagine a graph as an electrical network with the edges as unit resistances joining the two endpoints. Then for any two nodes in the graph, we define conductance as the source current if we were to apply a source of unit voltage across the two nodes. If the voltage source is applied across an edge, we get edge conductance. Resistance is inverse of conductance.

Why estimate conductance?

Conductance computation considers all paths between two nodes and weighs them by their strength. Thus, it can be viewed as a distance metric robust w.r.t. edge or node deletions unlike other metrics like shortest distance.

Spectral sparsification is the problem of compressing the graph, such that the Laplacian quadratic form of the sparsifier approximates that of the original. In [2], it is shown that approx. edge conductances can be used to construct high quality sparsifiers.

Problem Statement

In this project, we focus on edge conductance estimation, as is necessary for spectral sparsification. Our algorithms and analysis can be easily adapted for estimating conductances between non-adjacent nodes

In many practical situations, the graph under consideration can be very huge and dynamic. In such cases, we need adaptive algorithms with low computation per iteration, low memory footprint, and amenability to parallelization. Thus we use MCMC based estimation procedures.

Objective:

Given an undirected, unweighted, finite graph, G = (V,E), we wish to design MCMC based algorithms to estimate all edge conductances. Throughout, we shall assume that the markov chain associated with the simple random walk on the graph is aperiodic., with a unique stationary distribution.

Basic Definitions and Notations

- $m = |E|$, $n = |V|$
- $N(i)$ is the neighborhood of node i. i.e. $N(i) = \{j|(i,j) \in E\}$
- $d(i)$ = the degree of node $i = |N(i)|$
- $d_{max} = \max_{i \in V} d(i)$ = the the maximum degree.
- Unique stationary distribution of simple random walk $= \pi$
- $D_{avg} = \sum_{i \in V} d(i)\pi(i) = \sum_{i \in V} \frac{d(i)^2}{2m}$
- t_{mix} = the $(1/4)$ mixing time of the simple markov chain
- For every $(i,j) \in E$, let G_{ij} be the effective conductance and R_{ij}, the effective resistance. $R_{max} = \max_{(i,j) \in E} R_{ij}$

Hitting time distributions have exponential tails. Let λ_{min} be the exponent of the slowest decaying tail.

Proposed Algorithms

Algorithm 1 VisitProbMCMC : Based on visit probability before return

1. Input N, $G = (V, E)$
2. For each $i \in V$, initialize $N_i = 0$.
3. For each $(i,j) \in E$, initialize $\widehat{p}_{ij} = \widetilde{p}_{ij} = 0$.
4. Sample initial node X_0, from the stationary distribution $\pi(i) = d(i)/2m$.
5. for $t = 0$ to $N - 1$
   Let $X_t = i$
   (a) for each j in $N(i)$:
      i. $\widehat{p}_{ij} \leftarrow (\widehat{p}_{ij} N_i + \widetilde{p}_{ij})/(N_i + 1)$
      ii. $\widetilde{p}_{ij} \leftarrow 0$
      iii. $\widetilde{p}_{ji} \leftarrow 1$
   (b) $N_i \leftarrow N_i + 1$
   (c) Jump to one of the neighbors of the current node with equal probability
      i.e. $\mathbb{P}(X_{t+1} = j | X_t = i) = 1/d(i) \qquad \forall j \in N(i)$
6. For every $(i,j) \in E$ output $\widehat{G}_{ij} = \frac{d(i)}{2}\widehat{p}_{ij} + \frac{d(j)}{2}\widehat{p}_{ji}$

Interpretation for the variables and logic in the above algorithm:

1. $\widetilde{p}_{ij}$ denotes the success or failure of visiting j in an instance of a return path from i to i contained in the random walk.
2. In the long run, every visit to node i marks end of a return path. Thus, on visiting node i, we can update the $\widehat{p}_{ij}$ using $\widetilde{p}_{ij}$.
3. Also, in the long run, every visit to node i will necessarily be part of a return path of every other node, $\widetilde{p}_{ji}$ can be updated to 1 on vistiing i.
4. N_i is the number of times node i was visited
5. $\widehat{p}_{ij}$ which is just average of $\widetilde{p}_{ij}$ is an estimate of p_{ij}.
6. Now, $G_{ij} = \frac{d(i)}{2}p_{ij} + \frac{d(j)}{2}p_{ji}$. We have estimated $\widehat{G}_{ij} = \frac{d(i)}{2}\widehat{p}_{ij} + \frac{d(j)}{2}\widehat{p}_{ji}$

Algorithm 2 HittingTimeMCMC : Based on hitting time estimation

1. Input N, G = (V,E)
2. For each $(i,j) \in E$, initialize $\widehat{H}_{ij} = N_{ij} = 0, LV_{ij} = -1$
3. Sample initial node X_0, from the stationary distribution $\pi(i) = d(i)/2m$.
4. for $t = 0$ to $N - 1$
   (a) Let $X_t = i$. for each j in $N(i)$:
      i. if $LV_{ji} \geq 0$
         A. $\widehat{H}_{ji} \leftarrow ((t - LV_{ji}) + \widehat{H}_{ji} N_{ji})/(N_{ji} + 1)$
         B. $N_{ji} \leftarrow N_{ji} + 1$
         C. $LV_{ji} \leftarrow -1$
      ii. if $LV_{ij} < 0$
         A. $LV_{ij} \leftarrow t$
   (b) Jump to one of the neighbors of the current node with equal probability
      i.e. $\mathbb{P}(X_{t+1} = j | X_t = i) = 1/d(i) \qquad \forall j \in N(i)$
5. For every $(i,j) \in E$ output $\widehat{R}_{ij} = \frac{1}{2m}(\widehat{H}_{ij} + \widehat{H}_{ji})$

Interpretation for the variables and logic in the above algorithm:

1. $\widehat{H}_{ij}$ is an estimate of hitting time form node i to j (H_{ij}).
2. If LV_{ij} is negative, it means that immediate visit to node j will not yield any estimate for H_{ij}. This happens if the random walk returns to node j without visiting node i.
3. If LV_{ij} is non-negative, it denotes the last visit time of node i relevant for estimation of H_{ij}. Note that this is not the same as latest visit time of node i, because in cases when node i is visited multiple times before hitting j, the first (and not the latest) visit to i is useful for estimating H_{ij}.
4. N_{ij} is the number of estimates for H_{ij}
5. On visiting node i, if we can get an estimate of H_{ji}, we update $\widehat{H}_{ji}$ and since no estimate of H_{ji} can be obtained immediately after that, we set LV_{ji} to -1.
6. If we were not able to get any estimates for H_{ij}, this changes on visiting node i. Thus, LV_{ij} is then set to the node visit time t.
7. Now, $R_{ij} = \frac{1}{2m}(H_{ij} + H_{ji})$. So, we have estimated it as $\widehat{R}_{ij} = \frac{1}{2m}(\widehat{H}_{ij} + \widehat{H}_{ji})$

Theoretical Results

Theorem 1:

In VisitProbMCMC Algorithm if $N = \widetilde{O}\left(\frac{d_{max}(m + d_{max}t_{mix}^2)}{\epsilon^2} log\frac{1}{\delta}\right)$ for $0 < \epsilon \ll 1$ and $0 < \delta < 1$, then the output $\widehat{G}_{ij}$ for each edge $(i,j) \in E$ satisfies

$$\mathbb{P}(|\widehat{G}_{ij} - G_{ij}| < \epsilon) \geq 1 - \delta$$

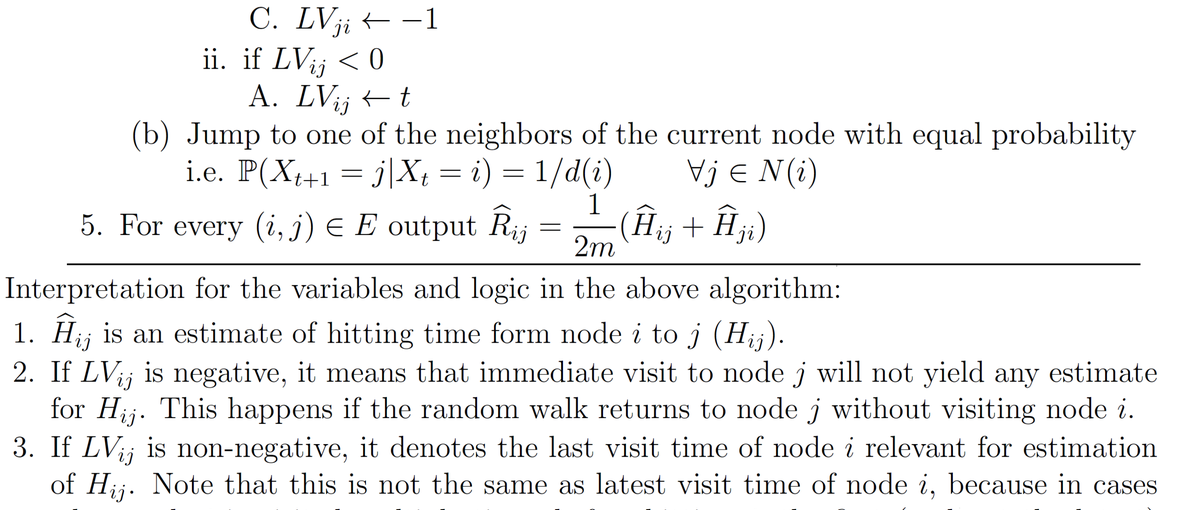
Theorem 2:

In HittingTimeMCMC Algorithm if $N = O\left(\frac{t_{mix}^2}{m^2\epsilon^2} log\frac{1}{\delta} + \frac{R_{max}}{m\epsilon^2\lambda_{min}^2}\left(log\frac{1}{m\epsilon} + log\frac{1}{\delta}\right)^3\right)$ for $0 < \epsilon \ll 1$ and $0 < \delta < 1$, then the output $\widehat{R}_{ij}$ for each edge $(i,j) \in E$ satisfies

$$\mathbb{P}(|\widehat{R}_{ij} - R_{ij}| < \epsilon) \geq 1 - \delta$$

Both algorithms run in $O(m)$ space and $O(D_{avg}N)$ time.

Numerical Experiments



References

1. L. Lovasz, Random Walks on Graphs: A Survey, Combinatorics, Paul Erdos is Eighty
2. DA Spielman, N Srivastava, Graph sparsification by effective resistances, SIAM 2011
3. Daniel Paulin, Concentration inequalities for Markov chains by Marton couplings and spectral methods.(arXiv:1212.2015 [math.PR])