

Laplacian Systems and Randomized Kaczmarz

Lecturer: Sushant Sachdeva

Scribe: Sam Anklesaria

1. INTRODUCTION TO LAPLACIAN SYSTEMS

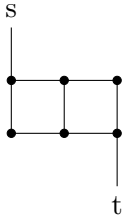
Let $G = (V, E)$ be a graph of m edges and v vertices. The matrix L is a Laplacian for graph G if $L = \sum_{(x,y) \in E} (e_x - e_y)^T (e_x - e_y)$ where e_x is the vector with a one in the x -th coordinate, and zeros elsewhere. We can notice a few useful properties:

- (1) $L \succeq 0$
- (2) $L\mathbf{1} = 0$, where $\mathbf{1}$ is the vector of all ones.
- (3) If G is connected, L has only one zero eigenvalue.

Laplacian systems are of the form $Lv = b$. We can assume $b^T \mathbf{1} = 0$. Even though L is not invertible, if G is connected, $\exists! v$ such that $Lv = b$ and $v^T \mathbf{1} = 0$. Then $v = L^+ b$, where L^+ is the pseudoinverse of L . In particular, if the spectral decomposition of $L = \sum_{i=1}^n \lambda_i u_i u_i^T$ has $\lambda_i > 0$ for $i \geq 2$, then the pseudoinverse $L^+ = \sum_{i=2}^n \frac{1}{\lambda_i} u_i u_i^T$.

2. APPLICATION TO ELECTRICAL SYSTEMS

Laplacians can be used to solve electrical systems. Assume G represents a network of resistors, where each edge has resistance 1.



Say we send one unit of current along this network, with input at node s and output from node t . What is the flow?

We can pick an arbitrary orientation for each edge. The flow on each edge $f_{(x,y)}$ will be positive in the chosen direction and negative in the opposite direction. The net flow at a vertex x is the amount current flowing out minus the amount of current flowing in:

$$\sum_{y:(x,y) \in E} f_{(x,y)} - \sum_{y:(y,x) \in E} f_{(y,x)} = \begin{cases} 1 & \text{if } x = s \\ -1 & \text{if } x = t \\ 0 & \text{otherwise} \end{cases}$$

More generally, we can use an n by m incidence matrix B . Each column will represent an edge, (x, y) with a -1 in the x -th position and a 1 in the y -th position. Then for any flow f , the net flow at vertex v is just the v -th component of Bf . In the case of unit flow from s to t , $Bf = e_s - e_t$. The system $Bf = b$ will be referred to as the flow constraints, where b gives the net flow out of each vertex.

The energy of a flow f is given by

$$\mathcal{E}(f) = \sum_{(x,y) \in E} \frac{1}{2} f_{(x,y)}^2 = \frac{1}{2} \|f\|^2$$

The electrical flow through a system will be the flow that minimizes energy while satisfying the flow constraints. So assuming the net flow out of each vertex is given by the vector b , we want to

pick f^* to minimize $\frac{1}{2} \|f\|^2$ such that $Bf = b$. We can solve this by minimizing the Lagrangian $L(f, v) = \frac{1}{2} \|f\|^2 - v^T [Bf - b]$. By the KKT conditions, $\frac{d}{df} L(f^*, v^*) = 0$, so $f^* = B^T v^*$. This factor v^* is simply the voltage of the system. Note that for each edge (x, y) , $(B^T v^*)_{(x,y)} = v_x - v_y$, which is just Ohm's law. If $f^* = B^T v^*$ and $Bf^* = b$, then $(BB^T)v^* = b$. But BB^T is just the Laplacian L . So we can solve for the electrical flow by solving the Laplacian system $Lv^* = b$.

Laplacian systems can be solved using the standard semidefinite programming techniques, such as gradient decent and conjugate gradient. If the graph G is an n cycle, then the condition number $\frac{\lambda_{\max}}{\lambda_{\min}}$ of the Laplacian is $\kappa(l) = \Omega(n^2)$ so conjugate gradient $\in O(n(m+n)\log(1/\epsilon))$. But we can do better than $O(n^2)$. By [Spielman and Teng, 2004], we can compute x such that $\|x - L^+b\| \leq \epsilon \|L^+b\|$ in $\tilde{O}(m \log(\frac{1}{\epsilon}))$, where $\tilde{O}(f(n)) = O(f(n)\text{poly}(\log(n)))$. We will present a proof of this result from [Kelner et al., 2013], using the randomized Kaczmarz method.

3. CIRCULATIONS

A vector $z \in \mathbb{R}^n$ such that $Bz = 0$ is called a circulation if $\forall x, (Bz)_x = 0$. In other words, the net flow at all vertices is 0.

As a simple example, consider a cycle c in the graph. Send 1 unit current along each edge in c - we will denote this flow by $\mathbf{1}_c$. Clearly, $\mathbf{1}_c^T f = 0$. This is just Kirchoff's Law. Other circulations are just linear combinations of cycles- that is $\{z | Bz = 0\} = \text{span}(\{\mathbf{1}_c | c \text{ is a cycle}\})$.

Let $T \subseteq G$ be a spanning tree for G . For any edge not in the spanning tree, adding that edge creates a unique cycle:

$$\forall e \in E(G) \setminus E(T), \exists! c_e \in T \cup \{e\}$$

This means an electric flow f must obey the following rules:

- (1) Flow constraints: $Bf = b$
- (2) Kirchoff's Law: $\forall e \in G \setminus T, \mathbf{1}_{c_e}^T f = 0$ where $\mathbf{1}_{c_e}$ is the unit circulation along the unique cycle in $T \cup \{e\}$.

4. RANDOMIZED KACZMARZ METHOD

The Kaczmarz method [39] is a simple way of solving linear equations $Ax = b$. Start with arbitrary values for x . While there are constraints that are violated, pick a violated constraint $a_i^T x = b$ and set x to its projection along that constraint.

$$x_{k+1} = x_k + \frac{b_i - a_i^T x_k}{\|a_i\|^2} a_i$$

When solving electrical systems, the linear constraints along which we project will be the ones given by Kirchoff's law. This modified Kaczmarz method from [Strohmer and Vershynin, 2009] proceeds as follows:

- (1) Start with f_0 such that $Bf_0 = b$, so that the flow constraints are obeyed.
- (2) For $t = 1 \dots k$ sample $e \in G \setminus T$ with probability proportional to $|C_e|$ and set

$$f_t = f_{t-1} - \frac{\mathbf{1}_{C_e}^T f_{t-1}}{\|\mathbf{1}_{C_e}\|^2} \mathbf{1}_{C_e}$$

Since $B\mathbf{1}_{C_e} = 0$ the update step will never cause the flow constraints to be violated. More specifically, let $\tau = \sum_{e' \in G \setminus T} |C_{e'}|$ and the probability P_e of picking edge e be

$$\frac{|C_e|}{\sum_{e'} |C_{e'}|} = \frac{|C_e|}{\tau}$$

To implement the Kaczmarz algorithm we just need two operations:

- (1) Query: computing $\mathbf{1}_{c_e}^T f_{t-1}$
- (2) Update: computing $f_{t-1} - \alpha \mathbf{1}_{c_e}$.

We can modify these operations to efficiently store and update flows in T . Pick an arbitrary fixed vertex $s \in V$ to be the root. Adding α to every edge in the circulation $C_{(a,b)}$ is the same as adding α to (a,b) , adding $-\alpha$ to every edge in the path from s to b , and adding α to every edge in the path from s to a .

Let $(d, T_0, T_1, \dots, T_k)$ be a tree decomposition of T if the removal of $d \in V$ partitions T into subtrees $T_0 \dots T_k$, where T_0 is rooted at s and contains d as a leaf, while the other T_i are rooted at d , and each T_i has at most $n/2 + 1$ vertices. For a spanning tree T rooted at s with $n \geq 2$ vertices, we can compute a tree decomposition in $O(n)$ time by starting at s and recursively picking the edge that leads to the largest subtree. Eventually we will find a d such that the size of all its subtrees have no more than $n/2$ vertices, making d the desired vertex separator. We know such a d exists thanks to Jordan in 1869.

Applying this idea recursively results in a separator decomposition tree of depth at most $\log(n)$. For each subtree rooted at d_i we can maintain two values: d_i -drop, the total potential drop on the path from s to d_i , and d_i -ext, the contribution to d_i drop from vertices beyond d . Flow along edges in $G \setminus T$ are stored in a single array. This allows query and update operations to be performed in $O(\log(n))$ time.

[Kelner et al., 2013] shows the convergence of this algorithm: each iteration i of the algorithm computes feasible $f_i \in \mathbb{R}^E$ such that

$$\mathbf{E}[\mathcal{E}(f_i)] - \mathcal{E}(f^*) \leq \left(1 - \frac{1}{\tau}\right)^i (\mathcal{E}(f_0) - \mathcal{E}(f^*))$$

5. LOW STRETCH SPANNING TREES

For the Kaczmarz algorithm to complete quickly, we need spanning trees for which τ is small. It turns out that τ is related to a property of spanning trees known as stretch. Let $P_{(a,b)}$ consist of the unique path in spanning tree T from a to b . Then the stretch of some edge e is the length of P_e , and the stretch of T is the sum of the stretches of its edges. This means that $|C_e| = 1 + \text{stretch}(e)$, and consequently that $\tau = \text{stretch}(T) + m - 2n + 2$.

[Abraham and Neiman, 2012] gives an algorithm to construct low stretch spanning trees. Specifically, there exists a spanning tree $T \subseteq G$ such that

$$\sum_{e \in G \setminus T} \|C_e\| = O(m \log(n) \log(\log(n)))$$

which can be constructed in time $O(m \log(n) \log(\log(n)))$. If we use such a spanning tree with the algorithm from [Kelner et al., 2013], in $\tilde{O}(m \log(1/\epsilon))$ time the algorithm can compute f such that $\mathbf{E}[\mathcal{E}(f)] \leq (1 + \epsilon) \mathcal{E}(f^*)$.

REFERENCES

- Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 395–406, New York, NY, USA, 2012. ACM.
- Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 911–920, New York, NY, USA, 2013. ACM.

Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 81–90, New York, NY, USA, 2004. ACM.

Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, 2009. ISSN 1069-5869.