CSE 599: Interplay between Convex Optimization and Geometry **Winter 2018**

Lecture 13: Cholesky Decomposition for Laplacian

Lecturer: Yin Tat Lee

Disclaimer: *Please tell me any mistake you noticed.*

In this lecture, we revisit the concepts we taught in the previous few lectures and show how they can be combined to get a simple algorithm for Laplacian systems. This lecture is meant to be expository without rigorous proof. Two key concepts we introduced are the leverage score and Cholesky decomposition. First, we recall what is Laplacian:

Definition 13.0.6. Given a graph $G = (V, E)$ with positive weights $w \in \mathbb{R}^E$. We define the Laplacian L by

$$L = \sum_{(i,j)\in E} w_{i,j}(e_i - e_j)(e_i - e_j)^\top.$$

Equivalently, we have $L = B^\top W B$ where W is the diagonal matrix given by w and each row b_e of B represent an edge e where $b_{e,i} = 1$ and $b_{e,j} = -1$ for $e = (i, j)$.

Note that $v^\top L v = \sum_{(i,j)\in E} w_{i,j}(v_i - v_j)^2$. One particularly important Laplacian matrix is the one given by path $\sum_i (x_i - x_{i+1})^2$, which is the function used to prove various lower bounds for first-order methods.

13.1 Cholesky Decomposition and Schur complement

Another way to compute Cholesky decomposition is via Schur complement:

$$\begin{bmatrix} l_{11} & 0 \\ l_{21} & L_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21}^\top \\ 0 & L_{22}^\top \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21}^\top \\ a_{21} & A_{22} \end{bmatrix}.$$

Solving the equation, we have

$$L_{22}L_{22}^\top = A_{22} - \frac{a_{21}a_{21}^\top}{a_{11}}.$$

We call the later term $A_{22} - \frac{a_{21}a_{21}^\top}{\sqrt{a_{11}}}$ is the Schur complement of the matrix A on the first coordinate. We can think this matrix is the result of eliminating the first coordinate from the matrix A. The formula above shows that we can compute the Cholesky decomposition by computing Schur complement repeatedly. In general, we have the following definition:

Definition 13.1.1. Given a matrix $M = \begin{bmatrix} M_{FF} & M_{FC} \\ M_{CF} & M_{CC} \end{bmatrix}$ where F is a block of variables. We define the Schur complement

$$\mathrm{Sc}(M, F) \stackrel{\text{def}}{=} M_{CC} - M_{CF}M_{FF}^{-1}M_{FC}.$$

It turns out that Schur complement can be defined for any convex function as follows:

Definition 13.1.2. Given a convex function $f(x, y)$ where x and y are blocks of variables. We define the Schur complement

$$\mathrm{Sc}(f, x)(y) = \min_x f(x, y).$$

It is easy to prove that the Schur complement of a convex function is convex. Also, note that if $f(x,y) = \begin{pmatrix} x \\ y \end{pmatrix}^\top M \begin{pmatrix} x \\ y \end{pmatrix}$, then Schur complement of f is simply the quadratic form of the Schur complement of M:

$$\mathrm{Sc}(f,x)(y) = y^\top \mathrm{Sc}(M,x)y.$$

In general, Schur complement makes an object much more complicated. For matrices, $\mathrm{Sc}(M,F)$ often much more dense than M itself. But at least, we can always store a matrix in n^2 space and hence even in worst case, Cholesky decomposition is a polynomial time algorithm. For convex functions, it is unclear how to take advantage of Schur complement.

Problem 13.1.3. Can you design a convex optimization algorithm that is based on Schur complement?

13.2 Schur complement of a graph

Now, we understand what does the Schur complement do on a graph. By abusing notation, we define $\mathrm{Sc}(G,v)$ be the schur complement of the Laplacian of graph G on vertex v. Given a graph G and a vertex v we want to eliminate. Suppose $G = G_1 + G_2$ where G_1 is the set of edges adjacent to v and G_2 is the rest of the graph. From the optimization formulation of Schur complement, we note that

$$\mathrm{Sc}(G,v) = \mathrm{Sc}(G_1,v) + G_2.$$

Therefore, it suffices to understand the Schur complement of a star graph.

Lemma 13.2.1. *Let G be a star graph on $[n] \cup \{r\}$ where r is the root and let v_i be the weight of the edge (r,i). Then, $\mathrm{Sc}(G,r)$ is a graph on $[n]$ with the weight w_{ij} given by*

$$w_{ij} = \frac{v_i v_j}{\sum_i v_i}.$$

Proof. The Laplacian of G is

$$G = \begin{bmatrix} \sum v_i & -v_1 & -v_2 & \cdots & -v_n \\ -v_1 & v_1 & 0 & \cdots & 0 \\ -v_2 & 0 & v_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -v_n & 0 & 0 & \cdots & v_n \end{bmatrix}.$$

By the formula $A_{22} - \frac{a_{21}a_{21}^\top}{a_{11}}$, we have that

$$(\mathrm{Sc}(G,r))_{ij} = 1_{i=j} v_i - \frac{v_i v_j}{\sum_i v_i}.$$

Clearly, this is a Laplacian. □

Note that if all v_i are same, then the new graph is simply a complete graph with weight $\frac{1}{n}$. In general, we call this new graph is a weighted complete graph. Therefore, Cholesky decomposition for Laplacian can be rewritten as follows:

1. For all vertices v
   (a) Append the weight of edges adjacent to v to the lower triangular matrix.
   (b) Remove all edges adjacent to v.

(c) Add a weighted complete graph on the neighbors of v.

The sparsity of the lower triangular matrix is the sum of the degree of v. This heavily depends on the ordering of vertices we use. In each iteration, we add $\frac{\deg(v)(\deg(v)-1)}{2} - \deg(v)$ many edges. Therefore, we may want to do it in a greedy way and pick the minimum degree vertices. This roughly corresponding to the minimum degree ordering MATLAB uses for solving general sparse symmetric linear systems.

The main difficulty of the Cholesky decomposition is the sparsity problem, which is caused by adding the weighted complete graph.

13.3 Leverage Score of a graph

But in the previous lectures, we learn how to handle sparsity problem!

Recall that for a general matrix $A^\top A$, the leverage score $\sigma_i(A) = a_i^\top (A^\top A)^{-1} a_i$. For unweighted graph $G = (V, E)$ with Laplacian $L = B^\top B$, the leverage score $\sigma_e(B)$ is exactly the effective resistance of the edge e. Precisely, we consider an electrical network given by G where each edge is replaced by a 1 ohm resistance. Then, $\sigma_{(i,j)}(B)$ is the resistance between the point i and point j.

In general, if there are multiple ways to go from i to j, then $\sigma_{(i,j)}(B)$ is small. If there is only one way to go from i to j, then $\sigma_{(i,j)}(B) = 1$. For example, the leverage score of every edge in a tree are all 1. In opposite, the leverage score of every edge in a complete graph is very small.

The key result of leverage score is the following

Lemma 13.3.1. *Given a matrix $A \in \mathbb{R}^{m\times n}$. Let X be the random matrix that is $\frac{1}{p_i} a_i a_i^\top$ with probability $p_i = \frac{\sigma_i(A)}{\sum_i \sigma_i(A)}$. For $T = O(\frac{n \log n}{\varepsilon^2})$, with probability $1 - \frac{1}{n^{O(1)}}$, we have that*

$$(1-\varepsilon)A^\top A \preceq \frac{1}{T}\sum_{i=1}^{T} X_i \preceq (1+\varepsilon)A^\top A$$

where X_i are independent copies of X.

When we apply this theorem to Laplacian, this shows that the Laplacian of any graph can be approximated by a graph with $O(n \log n)$ edges. We call that sparse graph as sparsifier. This can be improved to $O(n)$ edges by [78]. Such sparsifier has many applications in graph theory beyond convex optimization. For example, the sparsifier perverses the number of edge crossing any set. Therefore, it allows us to reduces the problem of any cut problem to a sparse graph.

To apply this to the Cholesky decomposition problem, we now compute the leverage score of a weighted complete graph.

Lemma 13.3.2. *Given a positive vector $v \in \mathbb{R}^n$. Consider the weighted complete graph on n vertices with the weight given by $w_{ij} = v_i v_j$ for $i \neq j$. Then, the leverage score of the edge (i, j) is given by*

$$\sigma_{(i,j)} = \frac{v_i + v_j}{\sum_i v_i}.$$

Proof. Let $\alpha = \sum_i v_i$. Then $L = \alpha V - vv^\top$ where V is the diagonal of v. Note that

$$L(\frac{e_i}{v_i} - \frac{e_j}{v_j}) = \alpha(e_i - e_j).$$

Hence,

$$\frac{\sigma_{(i,j)}}{v_i v_j} = (e_i - e_j)^\top L^{-1}(e_i - e_j) = \frac{1}{\alpha}(\frac{e_i}{v_i} - \frac{e_j}{v_j})^\top (e_i - e_j) = \frac{1}{\alpha}(\frac{1}{v_i} + \frac{1}{v_j}).$$

□

This gives a simple algorithm to sample a weighted complete graph: first sample i with probability proportional to v_i, then sample an edge from i.

13.4 "Approximate Cholesky Decomposition"

Now, we consider the following algorithm:

1. Let G be the given graph and $\alpha = \log^{O(1)} n$.
2. For $i = 1, 2, \cdots, n$
   (a) Pick a random vertex v_i.
   (b) Append the weight of edges adjacent to v to the lower triangular matrix.
   (c) Remove all edges adjacent to v_i.
   (d) Add a sparsifier of the weighted complete graph on the neighbors of v_i with $O(\alpha \log n) \cdot \deg(v_i)$ many edges.

Let K_i be the weighted complete graph at the i^{th} iteration and $\widetilde{K}_i$ is the sparsifier. Then, this algorithm finds the Cholesky decomposition of the matrix

$$G + \sum_{i=1}^{n} \left(\widetilde{K}_i - K_i \right).$$

Let $Y_i = G^{-\frac{1}{2}} \left(\widetilde{K}_i - K_i \right) G^{-\frac{1}{2}}$. It suffices to prove that

$$-\frac{1}{2} I \preceq \sum_{i=1}^{n} Y_i \preceq \frac{1}{2} I.$$

Since we find the sparsifier by sampling, we have that $\mathbb{E} Y_i = 0$. Therefore, $\sum_{i=1}^{n} Y_i$ is a martingale. By matrix concentration, we have that

$$\left\| \sum_{i=1}^{n} Y_i \right\|_{\text{op}} = O(\log n) \max_i \|Y_i\|_{\text{op}} + O(1) \left\| \sum_i \mathbb{E} Y_i^2 \right\|_{\text{op}}^{\frac{1}{2}}.$$

Let G_i be the graph at iteration i. Intuitively, G_i is not much larger than G. Therefore, $G^{-\frac{1}{2}} \left(\widetilde{K}_i - K_i \right) G^{-\frac{1}{2}}$ is roughly upper bounded by $G_{i-1}^{-\frac{1}{2}} \left(\widetilde{K}_i - K_i \right) G_{i-1}^{-\frac{1}{2}}$. The benefit of the latter term is that everything depends on the current iteration. Therefore, if this intuition is correct, it suffices to prove for the first iteration.

Lemma 13.4.1. *We have that $\|Y_1\|_{\text{op}} \leq \frac{1}{\sqrt{\alpha}}$. (Ignored small probability event here.)*

Proof. By Lemma 13.3.1, we have that

$$(1 - \frac{1}{\sqrt{\alpha}}) K_1 \preceq \widetilde{K}_1 \preceq (1 + \frac{1}{\sqrt{\alpha}}) K_1.$$

Since $K_1 \preceq G$, we have

$$-\frac{1}{\sqrt{\alpha}} G \preceq \widetilde{K}_1 - K_1 \preceq \frac{1}{\sqrt{\alpha}} G.$$

□

Lemma 13.4.2. *We have that* $\mathbb{E}Y_1^2 \preceq \frac{6}{\sqrt{\alpha}n}G$.

Proof. Since $\|Y_1\|_{\text{op}} \leq \frac{1}{\sqrt{\alpha}}$, we have

$$Y_1^2 \preceq \frac{1}{\sqrt{\alpha}}|Y_1| \preceq \frac{1}{\sqrt{\alpha}}\left(G^{-\frac{1}{2}}\widetilde{K}_1 G^{-\frac{1}{2}} + G^{-\frac{1}{2}}K_1 G^{-\frac{1}{2}}\right).$$

Next, we note that $\widetilde{K}_1 \preceq 2K_1$ (w.h.p.), we have

$$Y_1^2 \preceq \frac{3}{\sqrt{\alpha}}G^{-\frac{1}{2}}K_1 G^{-\frac{1}{2}}.$$

Hence, we have

$$\mathbb{E}Y_1^2 \preceq \frac{3}{\sqrt{\alpha}}G^{-\frac{1}{2}}\mathbb{E}K_1 G^{-\frac{1}{2}}.$$

Now, recall that K_1 is the Schur complement of a star S_v rooted at v. Using that $\text{Sc}(A, F) \preceq A$ for any matrix A, we have that

$$K_1 \preceq S_v.$$

Since we sample v uniformly, we have that

$$\mathbb{E}K_1 \preceq \mathbb{E}_v S_v \preceq \frac{2}{n}G$$

where the last sentence follows from the factor that every edge is picked with probability $\frac{2}{n}$. Hence, we have that $\mathbb{E}Y_1^2 \preceq \frac{6}{\sqrt{\alpha}n}G$. □

So, if this holds for all Y_i, then we would have

$$\left\|\sum_{i=1}^{n} Y_i\right\|_{\text{op}} = O\left(\frac{\log n}{\sqrt{\alpha}}\right).$$

Therefore, setting α large enough, we indeed have an approximate Cholesky decomposition of G.

Exercise 13.4.3. Formalize the proof by proving the bounds for Y_i.

13.5 Density Problem

13.5.1 Solution 1 - recursively sparsification and elimination

The previous algorithm is much better than the classical Cholesky decomposition because each step adds $\widetilde{O}(\deg(v))$ many edges instead of $O(\deg(v)^2)$. Unfortunately, it is still not enough to make sure the number of edges keeps bounded throughout. More preciously, if we add $C \cdot \deg(v)$ many edges each iteration, then the number of edges at worst case would double after $\frac{n}{C}$ iterations. Say it differently, if $T(m, n)$ is the cost of solving Laplacian system, we showed that

$$T(m, n) = \widetilde{O}(m) + \widetilde{O}(T(2^t m, (1 - \frac{1}{C})^t n)).$$

On the other hand, by sampling techniques, we know that

$$T(m, n) = \widetilde{O}(m) + \widetilde{O}(T(\frac{m}{k}, n) + T(kn \log n, n))$$

for any k. Now, the answer is very simple – combine them together. The algorithm is simply this:

- While
  - If $m \geq \rho n$ for some very large ρ
    - Deduce the sparsity by sampling
  - Do Cholesky decomposition

Note that if $m \geq \rho n$, the sample technique can reduce the number edges by as much as $\sqrt{\rho}$. Therefore, if ρ is large enough, we can affort to resample only after many iterations of Cholesky decomposition. By paying with the parameters and doing everything carefully, this would gives a nearly linear time algorithm for solving Laplacian.

They key interesting point about the proof is that we use nothing except that Schur complement of Laplacian is Laplacian!

13.5.2 Solution 2 - multi-edge

In the original paper [81], they take advantage the fact that we only need to compare our error with G instead of G_t. Then, they show that if we repeat every edge by K times, then throughout the algorithm, all edges has $b_i^\top G^{-1} b_i \leq \frac{1}{K}$. Therefore, we can sample K times less edge in the sampling for the weighted complete graph. Hence, they make sure every step, we only sample $\deg(v)$ edges every steps where $\deg(v)$ is counting the number of repeated edges. Using this, they show the number of multi-edges is bounded during the algorithm.

13.6 Extension

13.6.1 Parallel version

Another way to understand Cholesky decomposition is by the following block Gaussian elimination formula

$$M = \begin{bmatrix} I & 0 \\ M_{CF} M_{FF}^{-1} & I \end{bmatrix} \begin{bmatrix} M_{FF} & 0 \\ 0 & M_{CC} - M_{CF} M_{FF}^{-1} M_{FC} \end{bmatrix} \begin{bmatrix} I & M_{FF}^{-1} M_{FC} \\ 0 & I \end{bmatrix}.$$

Previously, we pick the F block is one variable. One benefit is that M_{FF}^{-1} can be computed easily by scalar division. However, notice that one can instead of a independent set, which still have the same property, easy to invert. In [82, 80], we consider a subset F such that M_{FF} is α-SDD, namely for all $i \in F$

$$(M_{FF})_{ii} \geq (1+\alpha) \sum_{i \neq j \in F} |M_{FF}|_{ij}.$$

Again, for such M_{FF}, it is easy to invert. This allows us to design a very fast block Cholesky decomposition. In particular, this shows that with nearly linear time preparation, we can find a linear size approximation Cholesky decomposition. This means that we can solve the same Laplacian repeatedly in exactly linear time!

13.6.2 Intuition for general convex problems

Although these calculations look like an algebraic trick, they are in fact very intuitive when viewed in the non-linear case. Basically, it involves recursive compute the Schur complement of a convex function f until it is small enough. At that point, the problem can be solved trivially. Then, it goes back the chain of Schur complement and extending the solution from the smallest set of variables to the original set of variables.

The major benefit of this recursive algorithm is that it avoids the diameter problem of convex optimization!

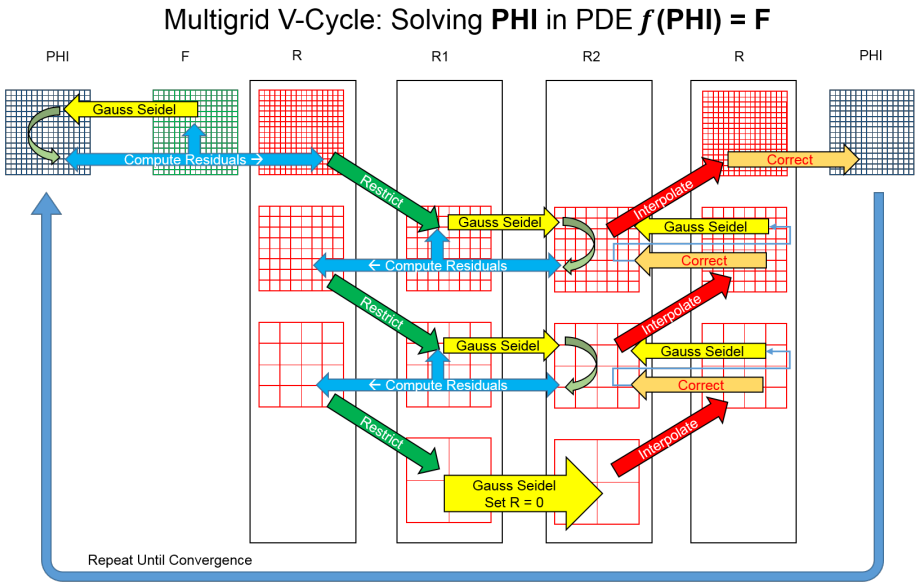

Figure 13.1: A picture from Wikipedia.

13.6.3 Scientific computing

Before the theory investigation of Cholesky decomposition for Laplacian, researchers have been using similar idea to solve all sorts of engineering problems. These methods are called multi-grid methods. They even designed algorithms that works for non-linear problems.

One great open problem is the following:

Problem 13.6.1. Can we use the multi-grid ideas for convex optimization in general?

The only known non-linear theory example I know is that it can be used to solve matrix scaling [79].

References

[78] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.

[79] Michael B Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. *arXiv preprint arXiv:1704.02310*, 2017.

[80] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 842–850. ACM, 2016.

[81] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 573–582. IEEE, 2016.

[82] Yin Tat Lee, Richard Peng, and Daniel A Spielman. Sparsified cholesky solvers for sdd linear systems. *arXiv preprint arXiv:1506.08204*, 2015.