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

Lecture 10: Overview and Leverage Score

Lecturer: Yin Tat Lee

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

In these two weeks, we discuss how to solve linear systems. Solving linear systems is the backbone of convex optimization. In principal, we can use interior point methods to solve any convex optimization and it often involves only solving dozens of linear systems. Therefore, if we can solve any linear systems in linear time, then we can probably solves most of the convex optimization problem in nearly linear time in practice.

10.1 How MATLAB solve $Ax = b$?

Instead of start with theory results, let me go over some basic on how MATLAB solves linear systems. Basically, MATLAB check if the matrix A is one of the following special matrices first. (For simplicity, I only write down the running time for dense matrices of that subclass.)

1. Diagonal matrix: $O(n)$ time
2. Tridiagonal matrix: $O(n)$ time
3. Banded matrix with bandwidth ℓ ($x_{ij} = 0$ for $|i - j| > \ell$): $O(n\ell^2)$ time
4. (Permuted) Triangular matrix: $O(n^2)$ time
5. Hessenberg matrix ($x_{ij} = 0$ for all $i \geq j - 1$): $O(n^2)$ time.

If the matrix is not one of the special matrices above, then:

1. If A is not square, use QR decomposition.
2. If A is Hermitian and the diagonal is all positive or all negative, try to use Cholesky decomposition.
3. Otherwise, use LDL decomposition.

For certain sparse matrix, the running time of QR, Cholesky and LDL can be linear time. In general, it depends on the sparsity pattern of the matrix and how many non-zero the algorithm creates during the process. These depends heavily on the ordering of the variables we used to run these algorithms. We will discuss this topic later.

Even though many of these algorithms have simple descriptions, it is an art to correctly implement these algorithms:

1. Make sure the algorithm is numerically stable.
2. Carefully decide the order to process/store the matrix to avoid cache misses. Memory bandwidth can be one limiting factor of these algorithms.
3. Group the calculation together to take advantage of special instruction sets. Modern computer (in 2018) can do up to 32 single-precision floating point operations per core per cycle.

4. Take advantage of all cores and GPU.

Take my desktop CPU as an example (6 cores). If programmed correctly, every cycle can run up to 192[3] single-precision floating point operations (using two 8-wide FMA instructions). In comparison, a naive implementation can easily take 3-5 cycle to finish one floating point operations. This gap will become even larger in the future.

10.2 What do we know about $Ax = b$ in general?

Here is a list of easy matrices that I can think of. If you know other important class of matrices that can be solved in nearly linear time, please tell me.

Theorem 10.2.1. *We can solve following matrices $A \in \mathbb{R}^{n \times n}$ in nearly linear time:*

- *Toeplitz matrix [59]: $A_{ij} = a_{i-j}$.*
- *Well-conditioned matrix [58]: Except for $\tilde{O}(1)$ many eigenvalues, the eigenvalues of $A^\top A$ is contained in $[\ell, u]$ where $\frac{u}{\ell} = \tilde{O}(1)$.*
- *Well-conditioned matrix [57]: $\frac{\sum_j \sqrt{\sum_i |a_{ij}|^2}}{\sqrt{\lambda_{\min}(A^\top A)}} = O(n)$[4] and each rows has similar number of non-zeros.*
- *Laplacian matrix [66]: $A = A^\top$ and $A_{ii} \geq \sum_{i \neq j} |A_{ij}|$.*
- *M matrix [63]: $A = B^\top B$ where each row of B contains two non-zeros.*
- *Connection Laplacian [64]:$A = A^\top$ and $A_{ii} \succeq \sum_{j \neq i} \|A_{ij}\|_2 \cdot I$ where A_{ij} are Hermitian block matrix.*
- *Directed Laplacian [60]: $A_{ij} \leq 0$ for all $i \neq j$ and $1^\top A = 0$.*
- *Hierarchical matrix: Off diagonal block of A can be approximated by $\tilde{O}(1)$ rank matrix. (Rough intuition only.)*

Remark. Toeplitz matrix usually comes from signal processing and integral equations. Well-conditioned matrix often comes from machine learning problems. Laplacian matrix, M matrix comes from graph problems. Connection Laplacian comes from signal processing. Directed Laplacian comes from Markov chain. Hierarchical matrix comes from the inverse of partial differential equations.

Proof. For Toeplitz matrix, it can be solved by conjugate gradient with a circulant matrix ($A_{ij} = a_{i-j \bmod n}$) as a preconditioner. Note that circulant matrices can be solved by fast Fourier transform.

For the well-conditioned matrix (1), it can be solved by conjugate gradient.

For the well-conditioned matrix (2), it can be solved by accelerated coordinate descent.

For Laplacian and directed Laplacian, it can be solved by repeated squaring $(I - M)^{-1} = \sum_{k=0}^{\infty} M^k = \prod_{k=1}^{\infty}(I + M^{2^k})$ and sparsification on M^{2^k}.

For M matrix, one can find a diagonal matrix D such that DAD is a Laplacian matrix.

For hierarchical matrix, note that matrix matrix multiplication of two hierarchical matrix is hierarchical. Hence, one can inverse hierarchical matrix by schur complement and matrix matrix multiplication. □

Problem. How to generalize these results to non-linear settings?

Unfortunately, we also know that a lot of matrices are pretty difficult [65].

[3] 2 flop/operation x 8 operations/instruction x 2 instructions/cycle x 6 core. Instruction set AVX512 is coming and hence it will become 16 operations/instruction soon.

[4] To understand the formula, note that $\frac{\sum_j \sqrt{\sum_i |a_{ij}|^2}}{\sqrt{\lambda_{\min}(A^\top A)}} \leq \sqrt{n}\sqrt{\frac{\sum_k \lambda_k(A^\top A)}{\lambda_{\min}(A^\top A)}}$.

10.3 Overdetermined systems

After some high level discussions, we move to the main theory result of today:

Theorem 10.3.1 ([61]). *Let $T(m, n, S, \varepsilon)$ be the cost of solving $A^\top Ax = b$ with accuracy ε, i.e.*

$$\left\|x - (A^\top A)^{-1}b\right\|_{A^\top A}^2 \leq \varepsilon \left\|(A^\top A)^{-1}b\right\|_{A^\top A}^2.$$

For $m \geq n$, we have

$$T(m, n, S, \varepsilon) = O(S \cdot \log^{O(1)}(\frac{m}{\varepsilon})) + O(T(O(n \log n), n, S, \frac{1}{m^{O(1)}}) \log^{O(1)}(\frac{m}{\varepsilon})).$$

Alternatively, it says we can solved overdetermined systems by solving few squares systems and reading the input few times. Here I am sloppy on the dependence on ε and the polylogarithmic terms.

10.3.1 Leverage scores

The key concept in this reduction is leverage scores.

Definition 10.3.2. Given a matrix $A \in \mathbb{R}^{m \times n}$. Let $a_i^\top$ be the i^{th} rows of A and the leverage score of the i^{th} row of A is

$$\sigma_i(A) \stackrel{\text{def}}{=} a_i^\top (A^\top A)^+ a_i.$$

Note that $\sigma(A)$ is the diagonal of the projection matrix $A(A^\top A)^+ A^\top$. Since $0 \preceq A(A^\top A)^+ A^\top \preceq I$, we have that $0 \leq \sigma_i(A) \leq 1$. Furthermore, since $A(A^\top A)^+ A^\top$ is a projection matrix, the sum of A's leverage scores is equal to the matrix's rank:

$$\sum_{i=1}^m \sigma_i(A) = \mathrm{tr}(A(A^\top A)^+ A^\top) = \mathrm{rank}(A(A^\top A)^+ A^\top) = \mathrm{rank}(A) \leq n. \tag{10.1}$$

A row's leverage score measures how important it is in composing the row space of A. If a row has a component orthogonal to all other rows, its leverage score is 1. Removing it would decrease the rank of A, completely changing its row space. The *coherence* of A is $\|\sigma(A)\|_\infty$. If A has low coherence, no particular row is especially important. If A has high coherence, it contains at least one row whose removal would significantly affect the composition of A's row space. Two characterizations that helps with this intuition follows:

Lemma 10.3.3. *For all $A \in \mathbb{R}^{m \times n}$ and $i \in [m]$ we have that*

$$\sigma_i(A) = \min_{A^\top x = a_i} \|x\|_2^2.$$

where a_i is the i^{th} row of A.

Lemma 10.3.4. *For all $A \in \mathbb{R}^{m \times n}$ and $i \in [m]$, we have that $\sigma_i(A)$ is the smallest t such that*

$$a_i a_i^\top \preceq t \cdot A^\top A. \tag{10.2}$$

Sampling rows from A according to their exact leverage scores gives a spectral approximation for A with high probability. Sampling by leverage score overestimates also suffices. To prove this, we need the following matrix concentration result:

Theorem 10.3.5 (Matrix Chernoff)**.** *Given a sequence of independent random self-adjoint matrices $M_k \in \mathbb{R}^{n\times n}$ such that $\mathbb{E}M_k = I$ and $0 \preceq M_k \preceq R \cdot I$. Then, for $T = \Theta(\frac{R}{\varepsilon^2}\log n)$, we have that*

$$(1-\varepsilon)I \preceq \frac{1}{T}\sum_{k=1}^{T} M_k \preceq (1+\varepsilon)I$$

with probability $1 - \frac{1}{n^{O(1)}}$.

One can see why $\frac{R}{\varepsilon^2}\log n$ is necessary by understanding the case for diagonal matrices.

Lemma 10.3.6. *Given a vector u of leverage score overestimates, i.e., $\sigma_i(A) \leq u_i$ for all i. Let X be the random matrix that is $\frac{1}{p_i}a_i a_i^\top$ with probability $p_i = \frac{u_i}{\|u\|_1}$. For $T = O(\frac{\|u\|_1 \log n}{\varepsilon^2})$, 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.

Proof. Note that $\mathbb{E}X = A^\top A$ and that

$$0 \preceq X = \frac{1}{p_i}a_i a_i^\top \preceq \frac{\|u\|_1}{\sigma_i}a_i a_i^\top \preceq \|u\|_1 \cdot A^\top A.$$

Now, the statement simply follows from matrix Chernoff bound with $M_k = (A^\top A)^{-\frac{1}{2}}X_k(A^\top A)^{-\frac{1}{2}}$ and $R = \|u\|_1$. □

To see why spectral approximation is useful, we note that

Exercise 10.3.7. Given symmetric matrices B and C such that $\frac{1}{2}B \preceq C \preceq 2B$. Consider the algorithm $x' = x - \frac{1}{8}B^{-1}(Cx - b)$. Then, we have that

$$\left\|x' - C^{-1}b\right\|_C^2 \leq \frac{3}{4}\left\|x - A^{-1}b\right\|_C^2.$$

Remark. You can show it either by applying gradient descent or a direct calculation.

Combining this exercise and Lemma 10.3.6, we have that

$$T(m,n,S) = \text{cost of compute } \sigma_i + O^*(T(\tilde{O}(n),n,S)) \tag{10.3}$$

where we used that $\|\sigma\|_1 = O(n)$. (I ignored the parameter on ε here.) However, computing σ exactly is too expensive for many purposes. In [**?**], they showed that we can compute leverage scores, σ, approximately by solving only polylogarithmically many regression problems. This result uses the fact that

$$\sigma_i(A) = \left\|A(A^\top A)^+ A^\top e_i\right\|_2^2$$

and that by the Johnson-Lindenstrauss Lemma these lengths are persevered up to a multiplicative error if we project these vectors onto certain random low dimensional subspaces.

Theorem 10.3.8 (Johnson-Lindenstrauss Lemma)**.** *Given vectors $v_i \in \mathbb{R}^n$. Let $N = \Theta(\frac{\log n}{\varepsilon^2})$ and $S \in \mathbb{R}^{N\times n}$ is a random matrix with each entries is an independent standard Gaussian with variance $\frac{1}{N}$. Then, we have that*

$$(1-\varepsilon)\|v_i\|_2^2 \leq \|Sv_i\|_2^2 \leq (1+\varepsilon)\|v_i\|_2^2.$$

In particular, this lemma shows that we can approximate $\sigma_i(A)$ via $\left\|SA(A^\top A)^+ A^\top e_i\right\|_2^2$. The benefit of this is that we can compute $SA(A^\top A)^+$ by solving $\frac{\log n}{\varepsilon^2}$ many linear systems. In other words, we have that

$$\text{cost of approximate } \sigma_i = O^*(S) + O^*(T(m,n,S)).$$

Putting it into (10.3), we have this useless formula

$$T(m,n,S) = O^*(S) + O^*(T(m,n,S)) + O^*(T(\tilde{O}(n),n,S)).$$

This is a chicken and egg problem. To solve an overdetermined system faster, we want to use leverage score to sample the rows. And to approximate the leverage score, we need to solve the original overdetermined system. (Even worse, we need to solve it few times.)

10.3.2 Uniform Sampling

The key idea to break this chicken and egg problem is to use uniform sampling. We define

$$\sigma_{i,S} = a_i^\top (A_S^\top A_S)^{-1} a_i$$

where A_S is A restricted to rows in S. Note that $A_S^\top A_S$ is an overestimate of σ_i. Hence, it suffices to bound $\|\sigma_{i,S}\|_1$. The key lemma is the following:

Lemma 10.3.9. *We have that*

$$\mathbb{E}_{|S|=k} \sum_{i=1}^m \sigma_{i,S\cup\{i\}} \leq \frac{mn}{k}.$$

Proof. Note that

$$\mathbb{E}_{|S|=k} \sum_{i=1}^m \sigma_{i,S\cup\{i\}} = \mathbb{E}_{|S|=k} \sum_{i\notin S} \sigma_{i,S\cup\{i\}} + \mathbb{E}_{|S|=k} \sum_{i\in S} \sigma_{i,S\cup\{i\}}.$$

Note that $\sum_{i\in S} \sigma_{i,S\cup\{i\}} = \sum_{i\in S} \sigma_{i,S} \leq n$. Hence, the second term is bounded by n.

For the first term, we note that "sample a set S of size k, then sample $i \notin S$" is same as "sample a set T of size $k+1$, then sample $i \in T$". Hence, we have

$$\begin{aligned}\mathbb{E}_{|S|=k}\mathbb{E}_{i\notin S}\sigma_{i,S\cup\{i\}} &= \mathbb{E}_{|T|=k+1}\mathbb{E}_{i\in T}\sigma_{i,T} \\ &\leq \mathbb{E}_{|T|=k+1}\frac{n}{k+1} = \frac{n}{k+1}\end{aligned}$$

Hence, we have that

$$\mathbb{E}_{|S|=k} \sum_{i=1}^m \sigma_{i,S\cup\{i\}} \leq \frac{n}{k+1}(m-k) + n = n\cdot\frac{m+1}{k+1}.$$

□

Next, using Sherman Morrison formula, we have that

$$\sigma_{i,S\cup\{i\}} = \begin{cases} \sigma_{i,S} & \text{if } i \in S \\ \frac{1}{1+\frac{1}{\sigma_{i,S}}} & \text{else} \end{cases}.$$

Namely, we can compute $\sigma_{i,S\cup\{i\}}$ using $\sigma_{i,S}$. Therefore, we have that

$$\text{cost of approximate } \sigma_{i,S\cup\{i\}} = O^*(S) + O^*(T(k,n,S)).$$

Using Lemma 10.3.9, we have now

$$T(m,n,S) = O^*(S) + O^*(T(k,n,S)) + O^*(T(\tilde{O}(\frac{mn}{k}),n,S)).$$

Picking $k = \sqrt{mn}$, we have that

$$T(m,n,S) = O^*(S) + O^*(T(\tilde{O}(\sqrt{mn}),n,S)).$$

Repeating this process $\tilde{O}(1)$ times, we have

$$T(m,n,S) = O^*(S) + O^*(T(\tilde{O}(n),n,S)).$$

10.3.3 Low-rank approximation

Here is another application of leverage score. Suppose we are given a matrix A that is close to rank k. Our goal is to find a subset S of rows of A and some diagonal D such that

$$(1-\varepsilon)AA^\top - \frac{\varepsilon}{k}\|A-A_k\|_F^2 \cdot I \preceq A_S^\top D A_S \preceq (1+\varepsilon)AA^\top + \frac{\varepsilon}{k}\|A-A_k\|_F^2 \cdot I \tag{10.4}$$

where A_k is the best rank k approximation to A, namely $A_k = \mathrm{argmin}_{\mathrm{rank}B=k}\|A-B\|_F^2$.

To understand the guarantee, we note that the additive error of $A_S^\top D A_S$ and $AA^\top$ on any rank k has nuclear norm $2\varepsilon \cdot \|A-A_k\|_F^2$.

To find a rank k matrix B such that $\|A-B\|_F^2 \le (1+\varepsilon)\|A-A_k\|_F^2$ requires more calculations [62], so we focus on proving the guarantee (10.4).

The idea of finding (10.4) is this: given $\|A-A_k\|_F^2$, we can roughly think

$$AA^\top = \text{rank } k \text{ matrix} \pm \frac{\|A-A_k\|_F^2}{k} \cdot I.$$

If $AA^\top$ is exactly rank k, we would have $\sum_i \sigma_i(A) = k$ and hence sampling from $\sigma_i(A)$ would gives us a rank $\tilde{O}(k)$ matrix that approximate $AA^\top$ really well. To avoid the effect of noise term, we consider the ridge leverage scores

$$\sigma_i^\lambda(A) = a_i^\top (A^\top A + \lambda I)^+ a_i.$$

Again, to show how many samples we need, we need to compute $\left\|\sigma^\lambda\right\|_1$.

Lemma 10.3.10. *We have that*

$$\sum_i \sigma_i^\lambda(A) \le k + \frac{1}{\lambda}\|A-A_k\|_F^2.$$

In particular, putting $\lambda = \frac{1}{k}\|A-A_k\|_F^2$ *gives* $\left\|\sigma^\lambda(A)\right\|_1 \le 2k$.

Proof. Let μ_i be the eigenvalues of $A^\top A$. Note that

$$\begin{aligned}
\sum_i \sigma_i^\lambda(A) &= \mathrm{tr}((A^\top A + \lambda I)^+ A^\top A) \\
&= \sum_i \frac{\mu_i}{\mu_i + \lambda} \\
&= k + \frac{1}{\lambda}\sum_{i>k}\mu_i \\
&= k + \frac{1}{\lambda}\|A-A_k\|_F^2.
\end{aligned}$$

□

Next, using this λ, we have

$$\frac{1}{\sigma_i^{\lambda}(A)} a_i a_i^{\top} \preceq A^{\top}A + \lambda I \preceq A^{\top}A + \frac{1}{k}\|A - A_k\|_F^2 \cdot I.$$

Using this and following the proof of Lemma 10.3.6, we get (10.4).

References

[57] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119, 2016.

[58] Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[59] Raymond H Chan and Michael K Ng. Conjugate gradient methods for toeplitz systems. *SIAM review*, 38(3):427–482, 1996.

[60] Michael B Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. *arXiv preprint arXiv:1611.00755*, 2016.

[61] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.

[62] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[63] Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 451–460, 2008.

[64] 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 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 842–850. ACM, 2016.

[65] Rasmus Kyng and Peng Zhang. Hardness results for structured linear systems. *arXiv preprint arXiv:1705.02944*, 2017.

[66] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 81–90, 2004.