ORIE 6334 Bridging Continuous and Discrete Optimization Nov 4, 2019

Lecture 15

Lecturer: David P. Williamson *Scribe: Abhishek Shetty*

1 Iterative Methods

For a graph G and a supply vector b, we would like to solve the linear system $L_G p = b$ for the potential p. We would also like to construct algorithms that take advantage of the sparsity of G. Even writing down $L_G^\dagger$ explicitly takes $O\left(n^2\right)$ time and space. This motivates us towards exploring iterative methods for solving linear systems of equations. To solve a linear system $A\mathbf{x} = \mathbf{b}$, iterative algorithms only involve multiplication of a matrix A with vectors, and for a matrix A whose sparsity is m, this can be done in time $O(m)$. One disadvantage of such methods is that unlike other methods like Gaussian elimination, this only returns an approximate solution, the gap becoming smaller the longer the algorithm runs. However, they are quite fast and require a low amount of space.

The basic idea behind iterative methods is that to solve a system of linear equations $A\mathbf{x} = \mathbf{b}$, where A is symmetric and positive definite, we start with a vector $\mathbf{x^0}$, perform the linear operation A on it (along with some vector additions) to get $\mathbf{x^1}$, and iteratively keep performing these operations to get the sequence $\mathbf{x^0}, \mathbf{x^1}, ..., \mathbf{x^t}$ and we stop when $\mathbf{x^t}$ is sufficiently close to the vector $\mathbf{x}^*$ which satisfies $A\mathbf{x}^* = \mathbf{b}$. The exact details will be outlined below, but we can see that the only expensive operation is multiplying by the matrix A, which is fast if A is sparse.

Before we dive into the algorithm, we note that if $A\mathbf{x}^* = \mathbf{b}$, then for any scalar α, we have $\alpha A\mathbf{x}^* = \alpha\mathbf{b}$, rearranging which gives us

$$\mathbf{x}^* = (I - \alpha A)\mathbf{x}^* + \alpha\mathbf{b}.$$

This tells us that $\mathbf{x}^*$ is the fixed point of the affine transformation indicated by the equation, and naturally leads to an iterative algorithm, called the Richardson Iteration. Formally, consider the following algorithm.

Algorithm 1: Richardson Iteration

$\mathbf{x_0} \leftarrow 0$
for $t \leftarrow 1$ **to** k **do**
  $\mathbf{x_t} \leftarrow (I - \alpha A)\mathbf{x_{t-1}} + \alpha\mathbf{b}$

[0]This lecture is based on scribe notes by Shijin Rajakrishnan for a previous version of this course and Lecture 12 of Daniel Spielman's course on Spectral Graph Theory.

Remark 1 *Note that if we rewrite our linear system as $A^{1/2}\mathbf{x} = A^{-1/2}\mathbf{b}$ (assuming A is invertible), then the Richardson iteration is equivalent to gradient descent for the square loss $\frac{1}{2}\left\|A^{1/2}\mathbf{x} - A^{-1/2}\mathbf{b}\right\|_2^2$ with step size α.*

In order to analyze the algorithm, consider the following definition.

Definition 1 (Spectral Norm) *Given a matrix M, define its spectral norm as*

$$\|M\| = \sup_{\mathbf{x}\neq 0} \frac{\|M\mathbf{x}\|_2}{\|\mathbf{x}\|_2}.$$

When A is symmetric, this can also be equivalently defined as

$$\|M\| = \max_i |\mu_i|$$

where μ_i are the eigenvalues of M.

Suppose that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of the matrix $I - \alpha A$. Then the eigenvalues of $I - \alpha A$ are $1 - \alpha\lambda_1 \geq 1 - \alpha\lambda_2 \geq \cdots \geq 1 - \alpha\lambda_n$, and thus

$$\|I - \alpha A\| = \max_i |1 - \alpha\lambda_i| = \max\left(|1 - \alpha\lambda_1|, |1 - \alpha\lambda_n|\right)$$

This is minimized when we take $\alpha = \frac{2}{\lambda_1 + \lambda_n}$, yielding $\|I - \alpha A\| = 1 - \frac{2\lambda_1}{\lambda_1 + \lambda_n}$.

Now we turn to the analysis of the convergence of the Richardson Iteration.

$$\begin{aligned} x_* - x_t &= \left[(I - \alpha A)x_* + \alpha b\right] - \left[(I - \alpha A)x_{t-1} + \alpha b\right] \\ &= (I - \alpha A)(x_* - x_{t-1}) \\ &= (I - \alpha A)_2(x_* - x_{t-2}) \\ &= (I - \alpha A)^t(x_* - x_0) \\ &= (I - \alpha A)^t x_*. \end{aligned}$$

We define $\mathbf{x^t}$ to be close to the $\mathbf{x}^*$ when the norm of their difference is a small fraction of the norm of $\mathbf{x}^*$. Then

$$\begin{aligned} \|\mathbf{x}^* - \mathbf{x^t}\| &= \|(I - \alpha A)^t \mathbf{x}^*\| \\ &\leq \|(I - \alpha A)^t\| \|\mathbf{x}^*\| \\ &= \left(1 - \frac{2\lambda_1}{\lambda_1 + \lambda_n}\right)^t \|\mathbf{x}^*\| \\ &\leq \exp\left(\frac{-2\lambda_1 t}{\lambda_1 + \lambda_n}\right) \|\mathbf{x}^*\|, \end{aligned}$$

where the final step used the fact that $1 - x \leq e^{-x}$. We set

$$t = \frac{\lambda_1 + \lambda_n}{2\lambda_1} \ln\left(\frac{1}{\epsilon}\right) = \left(\frac{\lambda_n}{2\lambda_1} + \frac{1}{2}\right) \ln\left(\frac{1}{\epsilon}\right),$$

so that we obtain

$$\|\mathbf{x}^* - \mathbf{x}^\mathbf{t}\| \leq \epsilon \|\mathbf{x}^*\|.$$

We can see that the speed of convergence, i.e the number of iterations required to get close to the solution to $A\mathbf{x} = \mathbf{b}$, depends on the ratio of the largest and smallest eigenvalues, $\frac{\lambda_n}{\lambda_1}$, and that the larger it is, the longer it takes for the algorithm to converge to the approximate solution.

Definition 2 *For a symmetric, positive definite matrix A with eigenvalues $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$, its* **condition number** *is defined as*

$$\kappa(A) = \frac{\lambda_n}{\lambda_1}$$

This algorithm was just one of the examples of an iterative methods to find an approximate solution to a linear system. There are other, faster methods (such as the Chebyshev method and Conjugate Gradient) that find an ϵ-approximate solution in $O\left(\sqrt{\kappa(A)} \ln\left(\frac{1}{\epsilon}\right)\right)$ iterations.

2 Preconditioning

We can see that if we modify the initial problem so that the condition number decreases, then the algorithms will run faster. Of course, since we change the problem, we need to worry about the implications on how far the new solution is from the old, and how the algorithm changes by changing the initial matrix. First, let us setup some notation.

Definition 3 *Given a matrix M, we say that $M \succeq 0$ iff M is positive semidefinite. Given a pair of matrices A and B, we say $B \succeq A$ iff $B - A \succeq 0$*

One such idea is *precondition* the matrix. For a matrix $B \succ 0$ (or $B \succeq 0$) that is symmetric and has the same nullspace as A, instead of solving $A\mathbf{x} = \mathbf{b}$, solve $B^\dagger A\mathbf{x} = B^\dagger \mathbf{b}$. Now we apply the iterative methods to the matrix $B^\dagger A$. This provides an improvement because we will prove that for a careful choice of B, we can reduce the condition number of the new matrix, and thus approximate the solution faster.

For solving $L_G\mathbf{p} = \mathbf{b}$, we precondition by $L_H^\dagger$, where H is a subgraph of G. In particular, we precondition by $L_T^\dagger$ where T is a spanning tree of G. This idea is attributed to Vaidya ([1]). Now the relevant condition number is $\lambda_n(L_T^\dagger L_G)/\lambda_2(L_T^\dagger L_G)$: We know that the smallest eigenvalue is zero, and thus look at the smallest positive eigenvalue for the condition number, which assuming the graph is connected is λ_2.

Claim 1 *For any subgraph H of G, $L_H \preceq L_G$.*

Proof: For all $\mathbf{x}$,

$$\begin{aligned}\mathbf{x}^T L_H \mathbf{x} &= \sum_{(i,j)\in H} (x(i) - x(j))^2 \\ &\leq \sum_{(i,j)\in E} (x(i) - x(j))^2 \\ &= \mathbf{x}^T L_G \mathbf{x}.\end{aligned}$$

Thus, $\mathbf{x}^T(L_G - L_H)\mathbf{x} \geq 0$. From this, we infer that $L_H \preceq L_G$. □

Claim 2 *$L_T^\dagger L_G$ has the same spectrum as $L_T^{\dagger/2} L_G L_T^{\dagger/2}$, where $L_T^{\dagger/2} = \sum_{i:\lambda_i \neq 0} \frac{1}{\sqrt{\lambda_i}} \mathbf{x_i} \mathbf{x_i}^T$ and $\lambda_i, \mathbf{x_i}$ are corresponding eigenvalues and eigenvectors of L_T.*

Proof: Consider an eigenvector $\mathbf{x}$ of $L_T^\dagger L_G$ of eigenvalue λ such that $\langle \mathbf{x}, \mathbf{e} \rangle = 0$. Then since $L_T^\dagger L_G \mathbf{x} = \lambda \mathbf{x}$, on setting $\mathbf{x} = L_T^{\dagger/2} \mathbf{y}$, we get $L_T^\dagger L_G L_T^{\dagger/2} \mathbf{y} = \lambda L_T^{\dagger/2} \mathbf{y}$. Premultiplying both sides by $L_G^{1/2} = \sum_{i:\lambda_i \neq 0} \sqrt{\lambda_i} \mathbf{x_i} \mathbf{x_i}^T$, we obtain $L_T^{\dagger/2} L_G L_T^{\dagger/2} \mathbf{y} = \lambda \mathbf{y}$, implying that λ is an eigenvalue of $L_T^{\dagger/2} L_G L_T^{\dagger/2}$ as well. □

Using these results, we can prove a bound on the smallest positive eigenvalue of $L_T^\dagger L_G$.

Lemma 3 *x*

$$\lambda_2(L_T^\dagger L_G) \geq 1.$$

Proof:

$$\begin{aligned}\lambda_2(L_T^\dagger L_G) &= \lambda_2(L_T^{\dagger/2} L_G L_T^{\dagger/2}) \\ &= \min_{\mathbf{x}:\langle \mathbf{x},\mathbf{e}\rangle = 0} \frac{x^T L_T^{\dagger/2} L_G L_T^{\dagger/2}}{x^T x} \\ &= \min_{\substack{\mathbf{y} = L_T^{\dagger/2}\mathbf{x} \\ \langle \mathbf{x},\mathbf{e}\rangle = 0}} \frac{y^T L_G y}{y^T L_T y} \\ &\geq 1,\end{aligned}$$

where the final step used the fact that $L_T \preceq L_G$. □

So we have bounded the denominator of the condition number of $L_T^\dagger L_G$, and we now turn to upper-bounding the numerator.

3 Connection to Low Stretch Spanning Trees

Suppose that G is a weighted graph, with weights $\frac{1}{w(i,j)} \geq 0$, for each edge $(i,j) \in E$. The above proof for bounding $\lambda_2(L_T^\dagger L_G)$ can be used to prove the same even for the weighted case. From the last lecture, recall that for a spanning tree T of G, and an edge $e = (k,l) \in E$, the stretch of e is defined as

$$\mathrm{st}_T(e) = \frac{\sum_{(i,j) \text{ on } k\text{-}l \text{ path in } T} w(i,j)}{w(k,l)}$$

and that the total stretch of the graph is

$$\mathrm{st}_T(G) = \sum_{e \in E} \mathrm{st}_T(e).$$

Lemma 4 *([2])* $\mathrm{tr}(L_T^\dagger L_G) = \mathrm{st}_T(G)$.

Proof:

$$\begin{aligned}
\mathrm{tr}(L_T^\dagger L_G) &= \mathrm{tr}\left(L_T^\dagger \sum_{(k,l)\in E} \frac{1}{w(k,l)} (e_k - e_l)(e_k - e_l)^T \right) \\
&= \sum_{(k,l)\in E} \frac{1}{w(k,l)} \mathrm{tr}\left(L_T^\dagger (e_k - e_l)(e_k - e_l)^T \right) \\
&\overset{(a)}{=} \sum_{(k,l)\in E} \frac{1}{w(k,l)} \mathrm{tr}\left((e_k - e_l)^T L_T^\dagger (e_k - e_l) \right) \\
&= \sum_{(k,l)\in E} \frac{1}{w(k,l)} r_{\text{eff}}(k,l) \\
&= \sum_{(k,l)\in E} \frac{1}{w(k,l)} \sum_{\substack{(i,j) \text{ in} \\ k\text{-}l \text{ path in } T}} w(i,j) \\
&= \mathrm{st}_T(G),
\end{aligned}$$

where (a) used the cyclic property of the trace; that is, the trace is invariant under cyclic permutations, and thus $\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB)$ (This is equivalent to the fact that $\mathrm{tr}\,(AB) = \mathrm{tr}\,(BA)$), and $r_{\text{eff}}(k,l)$ is the effective resistance in the tree T for sending one unit of current from k to l, with conductances $\frac{1}{w(i,j)}$. □

From this lemma, we can arrive at the required bound on the largest eigenvalue, $\lambda_n(L_T^\dagger L_G) \leq \mathrm{st}_T(G)$. Thus, from the previous two lemmas, we can see that the condition number of $L_T^\dagger L_G$ is at most $\mathrm{st}_T(G)$, and thus the linear system $L_T^\dagger L_G \mathbf{p} = L_T^\dagger \mathbf{b}$ can be ϵ-approximately solved for $\mathbf{p}$ in $O(\sqrt{\mathrm{st}_T(G)} \ln \frac{1}{\epsilon})$ iterations.

But now each iteration consists of multiplying by the matrix $L_T^\dagger L_G$, and initially, we need to compute $L_T^\dagger \mathbf{b}$ as well. Thus we can see that we need to be able to compute the product of a vector with $L_T^\dagger$ in an efficient way. Suppose that we have to compute $\mathbf{z} = L_T^\dagger \mathbf{y}$, equivalently, solve $L_T \mathbf{z} = \mathbf{y}$, then it turns out that since T is not just any subgraph but rather a spanning tree, this computation can be done in time $O(n)$.

To see this, we write down the equations in the system $L_T \mathbf{z} = \mathbf{y}$:

$$d_T(i)z(i) - \sum_{j:\{i,j\}\in T} z(j) = y(i) \qquad \forall i \in V.$$

Suppose that i is a leaf in T, with an incident edge (i, j). Then the relevant equation for this node is $z(i) - z(j) = y(i)$, i.e, $z(i) = z(j) + y(i)$. Note that since i is a leaf, the only equation in which the variable $z(i)$ appears is this one and the equation for $z(j)$. Thus we can substitute for $z(i)$ with $z(j) + y(i)$ and recurse on the smaller tree excluding the vertex i. This recursion will continue until we end up with a single edge (k, l). In this case, we set $z(k) = 0$, and back substitute to find the values of $\mathbf{z}$ for all the other vertices. It can be seen that this process takes $O(n)$ time, as in each step of the recursion, we do constant work and there are $n - 1$ recursive steps.

Thus we can compute the matrix product with $L_T^\dagger L_G$ in time $O(m)$, and recalling that for a graph G, we can find a low stretch spanning tree of stretch $\mathrm{st}_T(G) = O(m \log n \log\log n)$ in time $O(m \log n \log\log n)$, we can see that given the system $L_G \mathbf{p} = \mathbf{b}$, in $O\left(m \log n \log\log n + m\sqrt{\mathrm{st}_T(G)} \ln \frac{1}{\epsilon}\right) = \tilde{O}\left(m^{\frac{3}{2}} \ln \frac{1}{\epsilon}\right)$ time, we can find an ϵ-approximate solution.

Remember that in finding an upper bound for the largest eigenvector of $L_T^\dagger L_G$, we bounded it by its trace. [2] improved upon this running time bound by using the following result.

Theorem 5 ([3]; as stated in [2]) *For matrices $A, B \succeq 0$ with the same nullspace, let all but q eigenvalues of $B^\dagger A$ lie in the interval $[l, u]$, with the remaining eigenvalues larger than u. Then for a vector $\mathbf{b}$ in the rangespace of A, using the preconditioned conjugate gradient algorithm, an ϵ-approximate solution such that $\|\mathbf{x} - A^\dagger \mathbf{b}\|_A \leq \epsilon \|A^\dagger \mathbf{b}\|_A$ can be found in $q + \lceil \frac{1}{2} \ln \frac{2}{\epsilon} \sqrt{\frac{u}{l}} \rceil$ iterations, where $\|\mathbf{x}\|_A = \sqrt{\mathbf{x}^T A \mathbf{x}}$.*

We can use this theorem and since we have a bound on the trace, we can bound the number of large eigenvalues: Set $l = 1$, $u = (\mathrm{st}_T(G))^{\frac{2}{3}}$, then we can have at most $q = u = (\mathrm{st}_T(G))^{\frac{1}{3}}$ eigenvalues of value more than u. Now $\sqrt{\frac{u}{l}} = q$, and thus we get that the number of iteration required to solve the system approximately is $O\left((\mathrm{st}_T(G))^{\frac{1}{3}} \ln \frac{1}{\epsilon}\right) = \tilde{O}\left(m^{\frac{4}{3}} \ln \frac{1}{\epsilon}\right)$.

References

[1] Pravin M. Vaidya. *Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners.* Unpublished manuscript, UIUC

1990. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis, MN.

[2] Daniel A. Spielman and Jaeoh Woo. *A Note on Preconditioning by Low-Stretch Spanning Trees*. ArXiv 2009 `http://arxiv.org/abs/0903.2816`

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