Spectral Graph Theory Lecture 18

The Conjugate Gradient and Diameter

Daniel A. Spielman November 5, 2012

18.1 About these notes

These notes are not necessarily an accurate representation of what happened in class. The notes written before class say what I think I should say. The notes written after class way what I wish I said.

My description of the Conjugate Gradient method is based on the manuscript of Nisheeth Vishnoi [Vis12]. It is the simplest explanation of the Conjugate Gradient that I have seen.

18.2 Overview

I begin this lecture by describing the Conjugate Gradient algorithm for solving systems of linear equations. I finish by discussing implications for the diameters of graphs.

18.3 Review of Last Lecture

In the last lecture we encountered both the first-order Richardson iteration and the Chebyshev method for solving linear equations. We saw that these methods work because of the existence of special polynomials. Given an ϵ, and $0 < \lambda_1 < \lambda_n$, we constructed a polynomial $q(x)$ of degree

$$\ln(2/\epsilon)\left(\sqrt{\lambda_n/\lambda_1}+1\right)/2$$

such that

$$\begin{aligned} q(0) &= 1, \text{ and} \\ |q(x)| &\leq \epsilon, \text{ for } \lambda_1 \leq x \leq \lambda_n. \end{aligned}$$

Given a positive-definite matrix $\boldsymbol{A}$ with eigenvalues between λ_1 and λ_n and a right-hand side vector $\boldsymbol{b}$, we saw that

$$\boldsymbol{x}_t \stackrel{\text{def}}{=} p(\boldsymbol{A})\boldsymbol{b}$$

satisfied

$$\|\boldsymbol{x}_t - \boldsymbol{x}\| \leq \epsilon \|\boldsymbol{x}\|,$$

where $\boldsymbol{x}$ is the vector for which $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and p is the polynomial such that

$$q(x) = 1 - xp(x).$$

In this lecture we will learn about the Conjugate Gradient algorithm which essentially finds the optimal polynomial for any $\boldsymbol{A}$ and $\boldsymbol{b}$, quickly.

18.4 The Matrix Norm

The Conjugate Gradient algorithm will be optimal when we measure the error in the matrix norm rather than in the Euclidean norm. We define the matrix norm by

$$\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}}.$$

We say that $\boldsymbol{x}_t$ is an ϵ-approximate solution if

$$\|\boldsymbol{x}_t - \boldsymbol{x}\|_{\boldsymbol{A}} \leq \epsilon \|\boldsymbol{x}\|_{\boldsymbol{A}}.$$

While this at first looks like an unusual way to measure error, it turns out to be very useful. First, many algorithms naturally produce bounds on the error in the matrix norm. Second, for almost every application that I know of in which one wants to solve systems of linear equations, the matrix norm provides the right way to measure the quality of approximate solutions. We will see one such way next week.

We should observe that both the Richardon and Chebyshev methods achieve ϵ error in the $\boldsymbol{A}$-norm. Let q and p be polynomials satisfying the conditions stated in the previous section. Then,

$$\|p(\boldsymbol{A})\boldsymbol{b} - \boldsymbol{x}\|_{\boldsymbol{A}} = \|p(\boldsymbol{A})\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}\|_{\boldsymbol{A}}.$$

Now, write $\boldsymbol{x} = \sum_i c_i \boldsymbol{\phi}_i$. Then,

$$\boldsymbol{x} - p(\boldsymbol{A})\boldsymbol{A}\boldsymbol{x} = \sum_i c_i q(\lambda_i)\boldsymbol{\phi}_i,$$

so

$$\|p(\boldsymbol{A})\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}\|_{\boldsymbol{A}}^2 = \sum_i c_i^2 \lambda_i q(\lambda_i)^2 \leq \epsilon^2 \sum_i c_i^2 \lambda_i = \epsilon^2 \|\boldsymbol{x}\|_{\boldsymbol{A}}.$$

18.5 Optimality in the A-norm

The iterative methods that we consider begin with the vector $\boldsymbol{b}$, and then perform multiplications by $\boldsymbol{A}$ and take linear combinations with vectors that have already been produced. So, after t iterations they produce a vector that is in the span of

$$\left\{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \boldsymbol{A}^2\boldsymbol{b}, \ldots, \boldsymbol{A}^t\boldsymbol{b}\right\}.$$

This subspace is called the $t+1$st Krylov subspace generated by $\boldsymbol{A}$ and $\boldsymbol{b}$.

The Conjugate Gradient will find the vector $\boldsymbol{x}_t$ in this subspace that minimizes the error in the $\boldsymbol{A}$-norm. It will do so by computing a very useful basis of this subspace. But, before we describe this basis, let's examine the error in the $\boldsymbol{A}$ norm.

We have

$$\|\boldsymbol{x}_t - \boldsymbol{x}\|_{\boldsymbol{A}}^2 = \boldsymbol{x}_t^T \boldsymbol{A}\boldsymbol{x}_t - 2\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}_t^T \boldsymbol{A}\boldsymbol{x}_t - 2\boldsymbol{b}^T \boldsymbol{x}_t + \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x}.$$

So, minimizing the error in the $\boldsymbol{A}$-norm is equivalent to minimizing

$$\frac{1}{2}\boldsymbol{x}_t^T \boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{b}^T \boldsymbol{x}_t,$$

which is how the objective is usually presented.

Let $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_t$ be a basis of the $t+1$st Krylov subspace, and let

$$\boldsymbol{x}_t = \sum_{i=0}^{t} c_i \boldsymbol{p}_i.$$

Then, we have

$$\begin{aligned}\frac{1}{2}\boldsymbol{x}_t^T \boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{b}^T \boldsymbol{x}_t &= \frac{1}{2}\left(\sum_{i=0}^{t} c_i \boldsymbol{p}_i\right)^T \boldsymbol{A}\left(\sum_{i=0}^{t} c_i \boldsymbol{p}_i\right) - \boldsymbol{b}^T \left(\sum_{i=0}^{t} c_i \boldsymbol{p}_i\right) \\ &= \frac{1}{2}\sum_{i=0}^{t} c_i^2 \boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i - \sum_{i=0}^{t} c_i \boldsymbol{b}^T \boldsymbol{p}_i + \frac{1}{2}\sum_{i \neq j} c_i c_j \boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_j.\end{aligned}$$

To simplify the selection of the optimal constants c_i, the Conjugate Gradient will compute a basis $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_t$ that makes the rightmost term 0. That is, it will compute a basis such that $\boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_j = 0$ for all $i \neq j$. Such a basis is called an $\boldsymbol{A}$-orthogonal basis.

When the last term is zero, the objective function becomes

$$\sum_{i=0}^{t}\left(\frac{1}{2}c_i^2 \boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i - c_i \boldsymbol{b}^T \boldsymbol{p}_i\right).$$

So, the terms corresponding to different is do not interact, and we can minimize the sum by minimizing each term individually. The term

$$\frac{1}{2}c_i^2 \boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i - \boldsymbol{b}^T \boldsymbol{p}_i$$

is minimized by setting its derivative in c_i equal to zero, which gives

$$c_i = \frac{\boldsymbol{b}^T \boldsymbol{p}_i}{\boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i}.$$

It remains to describe how we compute this $\boldsymbol{A}$-orthogonal basis. The algorithm begins by setting

$$\boldsymbol{p}_0 = \boldsymbol{b}.$$

The next vector should be $\boldsymbol{A}\boldsymbol{p}_0$, but $\boldsymbol{A}$-orthogonalized with respect to $\boldsymbol{p}_0$. That is,

$$\boldsymbol{p}_1 = \boldsymbol{A}\boldsymbol{p}_0 - \boldsymbol{p}_0 \frac{(\boldsymbol{A}\boldsymbol{p}_0)^T \boldsymbol{A}\boldsymbol{p}_0}{\boldsymbol{p}_0^T \boldsymbol{A}\boldsymbol{p}_0}.$$

It is immediate that

$$\boldsymbol{p}_0 \boldsymbol{A}\boldsymbol{p}_1 = 0.$$

In general, we set

$$\boldsymbol{p}_{t+1} = \boldsymbol{A}\boldsymbol{p}_t - \sum_{i=0}^{t} \boldsymbol{p}_i \frac{(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_i}{\boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i}.$$

Let's verify that $\boldsymbol{p}_{t+1}$ is $\boldsymbol{A}$-orthogonal to $\boldsymbol{p}_i$ for $i \leq t$, assuming that $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_t$ are $\boldsymbol{A}$-orthogonal. We have

$$\begin{aligned}
\boldsymbol{p}_j^T \boldsymbol{A}\boldsymbol{p}_{t+1} &= \boldsymbol{p}_j^T \boldsymbol{A}\boldsymbol{A}\boldsymbol{p}_t - \sum_{i=0}^{t} \boldsymbol{p}_j^T \boldsymbol{A}\boldsymbol{p}_i \frac{(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_i}{\boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i} \\
&= \boldsymbol{p}_j^T \boldsymbol{A}^2 \boldsymbol{p}_t - \boldsymbol{p}_j^T \boldsymbol{A}\boldsymbol{p}_j \frac{(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_i}{\boldsymbol{p}_j^T \boldsymbol{A}\boldsymbol{p}_j} \\
&= 0.
\end{aligned}$$

The computation of $\boldsymbol{p}_{t+1}$ is greatly simplified by the observation that all but two of the terms in this sum are zero: for $i < t-1$,

$$(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_i = 0.$$

To see this, note that

$$(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_i = \boldsymbol{p}_t^T \boldsymbol{A}(\boldsymbol{A}\boldsymbol{p}_i),$$

and that $\boldsymbol{A}\boldsymbol{p}_i$ is in the span of $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{i+1}\}$. So, this term will be zero if $i+1 < t$.

That means that

$$\boldsymbol{p}_{t+1} = \boldsymbol{A}\boldsymbol{p}_t - \boldsymbol{p}_t \frac{(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_t}{\boldsymbol{p}_t^T \boldsymbol{A}\boldsymbol{p}_t} - \boldsymbol{p}_{t-1} \frac{(\boldsymbol{A}\boldsymbol{p}_t)^T \boldsymbol{A}\boldsymbol{p}_{t-1}}{\boldsymbol{p}_{t-1}^T \boldsymbol{A}\boldsymbol{p}_{t-1}}.$$

So, one can compute $\boldsymbol{p}_{t+1}$ from $\boldsymbol{p}_t$ and $\boldsymbol{p}_{t-1}$ while using only a constant number of multiplications by $\boldsymbol{A}$ and a constant number of vector operations. This means that one can compute the entire basis $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_t$ while performing only $O(t)$ multiplications of vectors by $\boldsymbol{A}$ and $O(t)$ vector operations.

The computation of $\boldsymbol{x}_t$ by

$$\boldsymbol{x}_t = \sum_{i=0}^{t} \boldsymbol{p}_i \frac{\boldsymbol{b}^T \boldsymbol{p}_i}{\boldsymbol{p}_i^T \boldsymbol{A}\boldsymbol{p}_i}.$$

Only requires an additional $O(t)$ more such operations.

In fact, only t multiplications by $\boldsymbol{A}$ are required to compute $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_t$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$: every term in the expressions for these vectors can be derived from the products $\boldsymbol{A}\boldsymbol{p}_i$. Thus, the Conjugate Gradient algorithm can find the $\boldsymbol{x}^t$ in the $t+1$st Krylov subspace that minimizes the error in the $\boldsymbol{A}$-norm in time $O(tn)$ plus the time required to perform t multiplications by $\boldsymbol{A}$. **Warning:** the algorithm that I have presented here differs from the true Conjugate gradient in that the

true Conjugate Gradient re-arranges this computation to keep the norms of the vectors involved reasonably small. Without this adjustment, the algorithm that I've described will fail in practice as the vectors $\boldsymbol{p}_i$ will become too large.

18.6 How Good is CG?

The Conjugate Gradient is at least as good as the Chebyshev iteration, in that it finds a vector of smaller error in the $\boldsymbol{A}$-norm in any given number of iterations. The optimality property of the Conjugate Gradient causes it to perform remarkably well.

For example, one can see that it should never require more than n iterations. The vector $\boldsymbol{x}$ is always in nth Krylov subspace. Here's an easy way to see this. Let the *distinct* eigenvalues of $\boldsymbol{A}$ be $\lambda_1, \ldots, \lambda_k$. Now, consider the polynomial

$$q(x) \stackrel{\text{def}}{=} \frac{\prod_{i=1}^{k}(\lambda_i - x)}{\prod_{i=1}^{k} \lambda_i}.$$

You can verify that q is a degree k polynomial such that

$$q(0) = 1, \text{ and}$$
$$q(\lambda_i) = 0, \text{ for all } i.$$

So, CG should be able to find the exact answer to a system in $\boldsymbol{A}$ in $k-1$ iterations. I say "should" because, while this statement is true with infinite precision arithmetic, it doesn't work out quite this well in practice.

Ignoring for now issues of finite arithmetic, let's consider the importance of this for sparse matrices $\boldsymbol{A}$. By a sparse matrix, I mean one with at most cn non-zero entries, for some constant c. That's not a rigorous definition, but it will help guide our discussion. Multiplication by a sparse matrix can be done in time $O(n)$. So, CG can solve a system of equations in a sparse matrix in time $O(n^2)$. Note that this is proportional to how long it would take to just write the inverse of $\boldsymbol{A}$, and will probably be faster than any algorithm for computing the inverse. On the other hand, it only provides the solution to one system in $\boldsymbol{A}$.

For another interesting example, consider the hypercube graph on n vertices. It only has $\log_2 n$ distinct eigenvalues. So, CG will only need $\log_2 n$ iterations to solve linear systems in the Laplacian of the hypercube. While there are other fast algorithms the exploit the special structure of the hypercube, CG works well when one has a graph that is merely very close to the hypercube.

In general, CG works especially quickly on matrices in which the eigenvalues appear in just a few clusters, and on matrices in which there are just a few extreme eigenvalues. We will learn more about this in the next lecture.

18.7 Laplacian Systems, again

This would be a good time to re-examine what we want when our matrix is a Laplacian. The Laplacian does not have an inverse. Rather, we want a polynomial in the Laplacian that approximates its pseudo-inverse (which we defined back in Lecture 8). If we were exactly solving the system of linear equations, we would have found a polynomial p such that

$$p(\boldsymbol{L})\boldsymbol{b} = \boldsymbol{x},$$

where $\boldsymbol{b} = \boldsymbol{L}\boldsymbol{x}$, so this gives

$$p(\boldsymbol{L})\boldsymbol{L}\boldsymbol{x} = \boldsymbol{x}.$$

Of course, this is only reasonable if $\boldsymbol{x}$ is in the span of $\boldsymbol{L}$. If the underlying graph is connected, this only happens if $\boldsymbol{x}$ is orthogonal to the all-1s vector. Of course, $\boldsymbol{L}$ sends constant vectors to zero. So, we want

$$p(\boldsymbol{L})\boldsymbol{L} = \boldsymbol{\Pi},$$

where $\boldsymbol{\Pi}$ is the projection matrix that sends the constant vectors to zero, and acts as an identity on the vectors that are orthogonal to the constant vectors. Recall that $\boldsymbol{\Pi} = \frac{1}{n}\boldsymbol{L}_{K_n}$.

Similarly, p gives an ϵ-approximation of the pseudo-inverse if

$$\|p(\boldsymbol{L})\boldsymbol{L} - \boldsymbol{\Pi}\| \leq \epsilon.$$

18.8 Bounds on the Diameter

Our intuition tells us that if we can quickly solve linear equations in the Laplacian matrix of a graph by an iterative method, then the graph should have small diameter. We now make that intuition precise.

If s and t are vertices that are at distance greater than d from each other, then

$$\boldsymbol{\chi}_s^T \boldsymbol{L}^d \boldsymbol{\chi}_t = 0.$$

On the other hand, if $\boldsymbol{L}$ only has k distinct eigenvalues other than 0, then we can form a polynomial p of degree $k-1$ such that

$$\boldsymbol{L}p(\boldsymbol{L}) = \boldsymbol{\Pi}.$$

This allows us to prove the following theorem.

Theorem 18.8.1. *Let G be a connected graph whose Laplacian has at most k distinct eigenvalues other than 0. Then, the diameter of G is at most k.*

Proof. Let d be the diameter of the graph and let s and t be two vertices at distance d from each other. We have

$$\boldsymbol{e}_s^T \boldsymbol{\Pi} \boldsymbol{e}_t = -1/n.$$

On the other hand, we have just described a polynomial in $\boldsymbol{L}$ with zero constant term, given by $\boldsymbol{L}p(\boldsymbol{L})$, that has degree k and such that

$$\boldsymbol{L}p(\boldsymbol{L}) = \boldsymbol{\Pi}.$$

If the degree of this polynomial were less than d, we would have

$$\boldsymbol{e}_s^T \boldsymbol{L}p(\boldsymbol{L})\boldsymbol{e}_t = 0.$$

As this is not the case, we have $d \leq k$. □

We can similarly obtain bounds on the diameter from approximate pseudo-inverses. If p is a polynomial such that

$$\|p(\boldsymbol{L})\boldsymbol{L} - \boldsymbol{\Pi}\| \leq \epsilon,$$

then

$$\boldsymbol{e}_s^T(p(\boldsymbol{L})\boldsymbol{L} - \boldsymbol{\Pi})\boldsymbol{e}_t \leq \|\boldsymbol{e}_s\| \, \|p(\boldsymbol{L})\boldsymbol{L} - \boldsymbol{\Pi}\| \, \|\boldsymbol{e}_t\| \leq \epsilon.$$

If s and t are at distance d from each other in the graph, and if the degree of $p(\boldsymbol{L})\boldsymbol{L}$ has degree less than d, then

$$\boldsymbol{e}_s^T(p(\boldsymbol{L})\boldsymbol{L} - \boldsymbol{\Pi})\boldsymbol{e}_t = \boldsymbol{e}_s^T(-\boldsymbol{\Pi})\boldsymbol{e}_t = 1/n.$$

This is a contradiction if $\epsilon < 1/n$. So, the polynomials we constructed from Chebyshev polynomials imply the following theorem of Chung, Faber and Manteuffel [CFM94]

Theorem 18.8.2. *Let $G = (V, E)$ be a connected graph, and let $\lambda_2 \leq \cdots \leq \lambda_n$ be its Laplacian eigenvalues. Then, the diameter of G is at most*

$$\left(\frac{1}{2}\sqrt{\frac{\lambda_n}{\lambda_2}} + 1\right) \ln 2n.$$

References

[CFM94] F. R. K. Chung, V. Faber, and T. A. Manteuffel. On the diameter of a graph from eigenvalues associated with its Laplacian. *SIAM Journal on Discrete Mathematics*, 7:443–457, 1994.

[Vis12] Nisheeth K. Vishnoi. *Lx = b*, 2012. available at `http://research.microsoft.com/en-us/um/people/nvishno/Site/Lxb-Web.pdf`.