Spectral Graph Theory Lecture 19

Preconditioning

Daniel A. Spielman November 7, 2012

19.1 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.

Eigen style.

19.2 Overview

Preconditioning is an approach to solving linear equations in a matrix $\boldsymbol{A}$ by finding a matrix $\boldsymbol{B}$ that is approximates $\boldsymbol{A}$, but such that it is easy to solve linear equations in $\boldsymbol{B}$. In this lecture, I explain

1. how this works when $\boldsymbol{B}$ is a very good approximation of $\boldsymbol{A}$,
2. how one can precondition the Chebyshev and Conjugate Gradient methods, and
3. how one can use low-stretch spanning tree preconditioners to solve linear equations in Laplacian matrices in time $O(m^{4/3} \log m)$.

19.3 Strong Approximations

For this lecture[1], I will say that a positive definite matrix $\boldsymbol{B}$ is an ϵ-approximation of a positive definite matrix $\boldsymbol{A}$ if

$$(1-\epsilon)\boldsymbol{B} \preccurlyeq \boldsymbol{A} \preccurlyeq (1+\epsilon)\boldsymbol{B}.$$

We will now see that if $\boldsymbol{B}$ is an ϵ-approximation of $\boldsymbol{A}$ then we can use solutions in linear systems in $\boldsymbol{B}$ to find solutions to linear systems in $\boldsymbol{A}$. Assume that we want to solve the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We will see that $\boldsymbol{B}^{-1}\boldsymbol{b}$ is not too far from $\boldsymbol{x}$, at least in the $\boldsymbol{A}$-norm. To start, we will need to obtain bounds on the eigenvalues of

$$\boldsymbol{I} - \boldsymbol{A}\boldsymbol{B}^{-1}. \tag{19.1}$$

[1] I tend to use slightly different definitions of this concept in different lectures.

The first issue we need to deal with is that this matrix is not necessarily symmetric, so it is not immediately clear that it is diagonalizable. This will not be a problem, as every positive-definite matrix has a square root. Let

$$\boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Psi}^T = \boldsymbol{A}$$

be the spectral factorization of $\boldsymbol{A}$ with eigenvectors contained in the columns of $\boldsymbol{\Psi}$ and the eigenvalues on the diagonals of $\boldsymbol{\Lambda}$. Then,

$$\boldsymbol{A}^{1/2} \stackrel{\text{def}}{=} \boldsymbol{\Psi}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\Psi}^T$$

is the *square root* of $\boldsymbol{A}$, and it is well-defined if all the eigenvalues of $\boldsymbol{A}$ are positive. The square root is also a symmetric, positive definite matrix. So, we can multiply the expression in (19.1) on the left by $\boldsymbol{A}^{-1/2}$ and on the right by $\boldsymbol{A}^{1/2}$ to see that it is similar to, and thus has the same eigenvalues as, the symmetric matrix

$$\boldsymbol{I} - \boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2}.$$

We could have just as well showed that it is similar to

$$\boldsymbol{I} - \boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}.$$

Let me first relate the eigenvalues of $B^{-1}A$ to our $\preccurlyeq$ notation.

Lemma 19.3.1. *Let A and B be positive definite matrices such that*

$$\alpha B \preccurlyeq A \preccurlyeq \beta B.$$

Then all the eigenvalues of $B^{-1}A$ lie between α and β.

Proof. We will just prove the upper bound. We have

$$\begin{aligned}
\lambda_{max}(B^{-1}A) &= \lambda_{max}(B^{-1/2}AB^{-1/2}) \\
&= \max_{\boldsymbol{x}} \frac{\boldsymbol{x}^T B^{-1/2}AB^{-1/2}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \\
&= \max_{\boldsymbol{y}} \frac{\boldsymbol{y}^T A\boldsymbol{y}}{\boldsymbol{y}^T B\boldsymbol{y}}, && \text{settting } \boldsymbol{y} = B^{-1/2}\boldsymbol{x}, \\
&\leq \beta.
\end{aligned}$$

□

So, if $\boldsymbol{B}$ is an ϵ-approximation of $\boldsymbol{A}$ then all of the eigenvalues of

$$\boldsymbol{I} - \boldsymbol{A}\boldsymbol{B}^{-1}$$

have absolute value at most ϵ.

Now, let's see that $\boldsymbol{B}^{-1}\boldsymbol{b}$ is a good approximation of $\boldsymbol{x}$ in the $\boldsymbol{A}$-norm. We have

$$\begin{aligned}
\left\|\boldsymbol{B}^{-1}\boldsymbol{b}-\boldsymbol{x}\right\|_{\boldsymbol{A}} &= \left\|\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{b}-\boldsymbol{A}^{1/2}\boldsymbol{x}\right\| \\
&= \left\|\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}^{1/2}\boldsymbol{x}\right\| \\
&= \left\|\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2}(\boldsymbol{A}^{1/2}\boldsymbol{x})-\boldsymbol{A}^{1/2}\boldsymbol{x}\right\| \\
&\leq \left\|\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2}-\boldsymbol{I}\right\| \left\|\boldsymbol{A}^{1/2}\boldsymbol{x}\right\| \\
&\leq \epsilon \left\|\boldsymbol{A}^{1/2}\boldsymbol{x}\right\| \\
&= \epsilon \left\|\boldsymbol{x}\right\|_{\boldsymbol{A}}.
\end{aligned}$$

Remark: This result crucially depends upon the use of the $\boldsymbol{A}$-norm. It fails under the Euclidean norm.

If we want a better solution, we can just compute the residual and solve the problem in the residual. That is, we set

$$\boldsymbol{x}_1 = \boldsymbol{B}^{-1}\boldsymbol{b},$$

and compute

$$\boldsymbol{r}_1 = \boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_1 = \boldsymbol{A}(\boldsymbol{x}-\boldsymbol{x}_1).$$

We then use one solve in $\boldsymbol{B}$ to compute a vector $\boldsymbol{x}_2$ such that

$$\left\|(\boldsymbol{x}-\boldsymbol{x}_1)-\boldsymbol{x}_2\right\|_{\boldsymbol{A}} \leq \epsilon \left\|\boldsymbol{x}-\boldsymbol{x}_1\right\|_{\boldsymbol{A}} \leq \epsilon^2 \left\|\boldsymbol{x}\right\|_{\boldsymbol{A}}.$$

So, $\boldsymbol{x}_1+\boldsymbol{x}_2$, our new estimate of $\boldsymbol{x}$, differs from $\boldsymbol{x}$ by at most an ϵ^2 factor. Continuing in this way, we can find an ϵ^k approximation of $\boldsymbol{x}$ after solving k linear systems in $\boldsymbol{B}$. This procedure is called *iterative refinement*.

For an example of how one might use a technique like this, consider solving a system in the Laplacian of a graph that is close to a hypercube. To be concrete, let H be the hypercube and let G be a weighted hypercube in which all the edge weights are between 1 and 2. Then, $\boldsymbol{L}_H$ is a 1/2-approximation of $\boldsymbol{L}_G$. Last class we saw that we could solve systems in H in time $O(m\log n)$, where m is the number of edges in H. This means that we can get 2^{-k}-approximate solutions to systems in $\boldsymbol{L}_G$ in time $O(km\log n)$.

That said, this is not necessarily a good idea. As all of the eigenvalues of $\boldsymbol{L}_G$ will be between 2 and $4\log_2 n$, directly applying Chebyshev or CG to this matrix would probably be faster.

19.4 Preconditioned Chebyshev

Preconditioning is usually applied with much weaker approximations. In this case, the running time of preconditioned solvers is determined by the eigenvalues of $\boldsymbol{A}\boldsymbol{B}^{-1}$, which are the same as the eigenvalues of $\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}$. Let $0<\lambda_1<\cdots<\lambda_n$ be the eigenvalues of $\boldsymbol{A}\boldsymbol{B}^{-1}$. The

preconditioned Chebyshev method can solve a system of equations in $\boldsymbol{A}$ to ϵ-accuracy in the $\boldsymbol{A}$-norm in approximately $\sqrt{\lambda_n/\lambda_1}\ln\epsilon^{-1}$ iterations (the same bound as before). In each iteration it will perform one multiplication by $\boldsymbol{A}$ and one linear solve in $\boldsymbol{B}$.

Let q_t be a polynomial that is 1 at 0 and that has absolute value less than ϵ at each of the eigenvalues λ_i, and let p_t be the polynomial such that $q_t = 1 - xp_t(x)$. To use this polynomial to solve the system, we set

$$\boldsymbol{x}_t = p_t(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{B}^{-1}\boldsymbol{b}.$$

To bound the error in the $\boldsymbol{A}$-norm of this vector, we compute

$$\begin{aligned}
\|\boldsymbol{x} - \boldsymbol{x}_t\|_{\boldsymbol{A}} &= \left\|\boldsymbol{A}^{1/2}\boldsymbol{x} - \boldsymbol{A}^{1/2}\boldsymbol{x}_t\right\| \\
&= \left\|\boldsymbol{A}^{1/2}\boldsymbol{x} - \boldsymbol{A}^{1/2}p_t(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{B}^{-1}\boldsymbol{b}\right\| \\
&= \left\|\boldsymbol{A}^{1/2}\boldsymbol{x} - \boldsymbol{A}^{1/2}p_t(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{x}\right\| \\
&= \left\|\boldsymbol{A}^{1/2}\boldsymbol{x} - \boldsymbol{A}^{1/2}p_t(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2}(\boldsymbol{A}^{1/2}\boldsymbol{x})\right\| \\
&\leq \left\|\boldsymbol{I} - \boldsymbol{A}^{1/2}p_t(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2}\right\| \left\|(\boldsymbol{A}^{1/2}\boldsymbol{x})\right\|.
\end{aligned}$$

We now prod this matrix into a more useful form:

$$\boldsymbol{I} - \boldsymbol{A}^{1/2}p_t(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2} = \boldsymbol{I} - p_t(\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2})\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2} = q_t(\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2}).$$

So, we find

$$\|\boldsymbol{x} - \boldsymbol{x}_t\|_{\boldsymbol{A}} \leq \left\|q_t(\boldsymbol{A}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{A}^{1/2})\right\| \left\|(\boldsymbol{A}^{1/2}\boldsymbol{x})\right\| \leq \epsilon \|\boldsymbol{x}\|_{\boldsymbol{A}}.$$

19.5 Preconditioned Conjugate Gradient

We can apply the same idea to precondition the conjugate gradient method. In each iteration, the method will mutliply a vector by $\boldsymbol{A}$ and solve a system of equations in $\boldsymbol{B}$. After t iterations, it will find the vector $\boldsymbol{x}_t$ that minimizes

$$\|\boldsymbol{x} - \boldsymbol{x}_t\|_{\boldsymbol{A}}$$

that is in the span of

$$\left\{\boldsymbol{B}^{-1}\boldsymbol{b}, \boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{B}^{-1}\boldsymbol{b}, \ldots, (\boldsymbol{B}^{-1}\boldsymbol{A})^t\boldsymbol{B}^{-1}\boldsymbol{b}\right\}.$$

If there is a polynomial q of degree $t+1$ that is one at zero and that has absolute value less than ϵ at each of the eigenvalues of $\boldsymbol{B}^{-1}\boldsymbol{A}$, then the error of the tth solution produced by the Preconditioned Conjugate Gradient (PCG) will be less than ϵ.

The code for the PCG is almost identical to that for CG, it finds an $\boldsymbol{A}$-orthogonal basis of the Krylov space generated by $\boldsymbol{B}^{-1}\boldsymbol{A}$. Alternatively, one can view the algorithm as running the ordinary Conjugate Gradient on the matrix $\boldsymbol{B}^{-1/2}\boldsymbol{A}\boldsymbol{B}^{-1/2}$ and with right-hand side $\boldsymbol{B}^{-1/2}\boldsymbol{b}$. However, it never needs to actually compute the square roots.

19.6 Preconditioning by Trees

Vaidya [Vai90] had the remarkable idea of preconditioning the Laplacian matrix of a graph by the Laplacian matrix of a subgraph. If H is a subgraph of G, then

$$\boldsymbol{L}_H \preccurlyeq \boldsymbol{L}_G,$$

so all eigenvalues of $\boldsymbol{L}_H^{-1}\boldsymbol{L}_G$ are at least 1. Thus, it remsinds for use to find subgraphs that are easy to invert and such that the largest eigenvalue of $\boldsymbol{L}_H^{-1}\boldsymbol{L}_G$ is not too big.

It is relatively easy to show that linear equations in the Laplacian matrices of trees can be solved exactly in linear time. One can either do this by finding an LU-factorization with a linear number of non-zeros, or by viewing the process of solving the linear equation as a dynamic program that passes up once from the leaves of the tree to a root, and then back down.

We will now show that a special type of tree, called a *low-stretch spanning tree* provides a very good preconditioner. To begin, let T be a spanning tree of G. Write

$$\boldsymbol{L}_G = \sum_{(u,v)\in E} w_{u,v}\boldsymbol{L}_{u,v} = \sum_{(u,v)\in E} w_{u,v}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T.$$

We will actually consider the trace of $\boldsymbol{L}_T^{-1}\boldsymbol{L}_G$. As the trace is linear, we have

$$\begin{aligned}
\mathrm{Tr}\left(L_T^{-1}\boldsymbol{L}_G\right) &= \sum_{(u,v)\in E} w_{u,v}\mathrm{Tr}\left(L_T^{-1}\boldsymbol{L}_{u,v}\right) \\
&= \sum_{(u,v)\in E} w_{u,v}\mathrm{Tr}\left(L_T^{-1}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T\right) \\
&= \sum_{(u,v)\in E} w_{u,v}\mathrm{Tr}\left((\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T\boldsymbol{L}_T^{-1}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)\right) \\
&= \sum_{(u,v)\in E} w_{u,v}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T\boldsymbol{L}_T^{-1}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v).
\end{aligned}$$

To see why the last step is true, recall that $\mathrm{Tr}\,(AB) = \mathrm{Tr}\,(BA)$ for all matrices A and B. To evalue this last term, we need to know the value of $(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T\boldsymbol{L}_T^{-1}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)$. You already know something about it: it is the effective resistance in T between u and v, and you proved that this equals the distance in T between u and v. Let $T(u,v)$ denote the path in T from u to v, and let $w_1,\ldots,w_k$ denote the weights of the edges on this path. We view the weight of an edge as the reciprocal of its length. So,

$$(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T\boldsymbol{L}_T^{-1}(\boldsymbol{\chi}_u - \boldsymbol{\chi}_v) = \sum_{i=1}^{k}\frac{1}{w_i}. \tag{19.2}$$

Even better, the term (19.2) is something that has been well-studied. It was defined by Alon, Karp, Peleg and West [AKPW95] to be the *stretch* of the unweighted edge (u,v) with respect to the tree T. Moreover, the *stretch* of the edge (u,v) with weight $w_{u,v}$ with respect to the tree T is defined to be exactly

$$w_{u,v}\sum_{i=1}^{k}\frac{1}{w_i},$$

where again $w_1, \ldots, w_k$ are the weights on the edges of the unique path in T from u to v. A sequence of works, begining with [AKPW95], has shown that every graph G has a spanning tree in which the sum of the stretches of the edges is low. The best result so far is due to [AN12], who prove the following theorem.

Theorem 19.6.1. *Every weighted graph G has a spanning tree subgraph T such that the sum of the stretches of all edges of G with respect to T is at most*

$$O(m \log n \log\log n),$$

where m is the number of edges G. Moreover, one can compute this tree in time $O(m \log n \log\log n)$.

Thus, if we choose a low-stretch spanning tree T, we will ensure that

$$\mathrm{Tr}\left(L_T^{-1} \boldsymbol{L}_G\right) = \sum_{(u,v)\in E} w_{u,v} (\boldsymbol{\chi}_u - \boldsymbol{\chi}_v)^T \boldsymbol{L}_T^{-1} (\boldsymbol{\chi}_u - \boldsymbol{\chi}_v) \leq O(m \log n \log\log n).$$

In particular, this tells us that $\lambda_{max}(L_T^{-1} \boldsymbol{L}_G)$ is at most $O(m \log n \log\log n)$, and so the Preconditioned Conjugate Gradient will require at most $O(m^{1/2} \log n)$ iterations, each of which requires one multiplication by $\boldsymbol{L}_G$ and one linear solve in $\boldsymbol{L}_T$.

This result is due to Boman and Hendrickson [BH01].

19.7 Improving the Bound on the Running Time

We can show that the Preconditioned Conjugate Gradient will actually run in closer to $O(m^{1/3})$ iterations. Since the trace is the sum of the eigenvalues, we know that for every $\beta > 0$, $\boldsymbol{L}_T^{-1} \boldsymbol{L}_G$ has at most

$$\mathrm{Tr}\left(L_T^{-1} \boldsymbol{L}_G\right) / \beta$$

eigenvalues that are larger than β.

To exploit this fact, we use the following lemma.

Lemma 19.7.1. *Let $\lambda_1, \ldots, \lambda_n$ be positive numbers such that all of them are at least α and at most k of them are more than β. Then, for every $t \geq k$, there exists a polynomial $p(X)$ of degree t such that $p(0) = 1$ and*

$$|p(\lambda_i)| \leq 2 \left(1 + \frac{2}{\sqrt{\beta/\alpha}}\right)^{-(t-k)},$$

for all λ_i.

Proof. Let $r(X)$ be the polynomial we constructed using Chebyshev polynomials of degree $t - k$ for which

$$|r(X)| \leq 2 \left(1 + \frac{2}{\sqrt{\beta/\alpha}}\right)^{-(t-k)},$$

for all X between α and β. Now, set

$$p(X) = r(X) \prod_{i:\lambda_i > \beta} (1 - X/\lambda_i).$$

This new polynomial is zero at every λ_i greater than β, and for X between α and β

$$|p(X)| = |r(X)| \prod_{i:\lambda_i > \beta} |(1 - X/\lambda_i)| \leq |r(X)|,$$

as we always have $X < \lambda_i$ in the product. □

Applying this lemma to the analysis of the Preconditioned Conjugate Gradient, with $\beta = \mathrm{Tr}\left(L_T^{-1} \boldsymbol{L}_G\right)^{2/3}$ and $k = \mathrm{Tr}\left(L_T^{-1} \boldsymbol{L}_G\right)^{1/3}$, we find that the algorithm produces ϵ-approximate solutions within

$$O(\mathrm{Tr}\left(L_T^{-1} \boldsymbol{L}_G\right)^{1/3} \ln(1/\epsilon)) = O(m^{1/3} \log n \ln 1/\epsilon)$$

iterations.

This result is due to Spielman and Woo [SW09].

19.8 Further Improvements

In fact, by combining low-stretch spanning trees and sparse high-quality graph approximations (called sparsifiers), one can get algorithms that solve linear systems in Laplacians in time $O(m \log n \log\log n \log \epsilon$ [ST09, KMP11].

19.9 Questions

The eigenvalues of $\boldsymbol{L}_H^{-1} \boldsymbol{L}_G$ are called generalized eigenvalues. The relation between generalized eigenvalues and stretch is the first result of which I am aware that establishes a combinatorial interpretation of generalized eigenvalues. Can you find any others?

I conjecture that it is possible to construct spanning trees of even lower stretch. Does every graph have a spanning tree of average stretch $2 \log n$?

References

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, February 1995.

[AN12] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC '12)*, pages 395–406, 2012.

[BH01] Erik Boman and B. Hendrickson. On spanning tree preconditioners. Manuscript, Sandia National Lab., 2001.

[KMP11] I. Koutis, G.L. Miller, and R. Peng. A nearly-mlogn time solver for sdd linear systems. In *Foundations of Computer Science (FOCS), 2011 52nd Annual IEEE Symposium on*, pages 590–598, 2011.

[ST09] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *CoRR*, abs/cs/0607105, 2009. Available at `http://www.arxiv.org/abs/cs.NA/0607105`.

[SW09] Daniel A. Spielman and Jaeoh Woo. A note on preconditioning by low-stretch spanning trees. *CoRR*, abs/0903.2816, 2009. Available at `http://arxiv.org/abs/0903.2816`.

[Vai90] 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., 1990.