MAT 228A Notes

Sam Fleischer

December 1, 2016

1 Last Time

Minimizing $\phi(u)$ is equivalent to minimizing $\|e_k\|_A$.

$$\operatorname{span}\{p_0, \dots, p_{k-1}\} = \operatorname{span}\{Ae_0, A^2e_0, \dots, A^ke_0\}$$

We can write $u_k = u_0 + \alpha_0 p_0 + \alpha_1 p_1 + \dots + \alpha_{k-1}p_{k-1}$, so,

$$u_k = u_0 + c_1Ae_0 + a_2A^2e_0 + \dots + c_kA^ke_0$$

Then substract the solution u from both sides:

$$\begin{aligned} e_k &= e_0 + c_1Ae_0 + c_2A^2e_0 + \dots + c_kA^ke_0 \\ &= q(A)e_0 \end{aligned}$$

where $q(A)$ is a polynomial of the matrix A. This means CG picks $q \in \pi_k$, which is the space of polynomials of degree at most k with $q(0) = 1$. Doing so minimizes $\|e_k\|_A = \|q(A)e_0\|_A$.

2 Analysis

Let A be diagonalizable. Then $A = Q\Lambda Q^{-1}$. Then $A^j = Q\Lambda^jQ^{-1}$ for any integer j. Then

$$q(A) = Qq(\Lambda)Q^{-1} = Q\begin{pmatrix} q(\lambda_1) & 0 & \dots & 0 \\ 0 & q(\lambda_2) & \dots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & q(\lambda_N) \end{pmatrix}Q^{-1}$$

We can show

$$\|e_k\|_A^2 = \|q(A)e_0\|_A^2 \le \underbrace{\max_j (q(\lambda_j))^2 \|e_0\|_A^2}_{\text{our error bound}}$$

Our error bound comes from minimizing $\max\limits_j (q(\lambda_j))^2$. We need to know how the polynomial behaves on the eigenvalues.

2.1 First CG step

So CG picks $q_1(x)$ so that $q_1(\lambda_1) = -q_1(\lambda_N)$, that is,

$$q_1(x) = 1 - \frac{2x}{\lambda_N + \lambda_1}$$

This is the polynomial of degree 1 through $(0, 1)$ which minimizes the maximum of q on the spectrum (supposing $\lambda_1 < \lambda_2 < \dots < \lambda_N$).

2.2 Second CG step

We can't solve this analytically for arbitrary eigenvalues. If we assume the eigenvalues are uniformly distributed, however, we should get a quadratic where $q_2(\lambda_1) = q_2(\lambda_N) = -q_2\left(\frac{\lambda_1 + \lambda_N}{2}\right)$. This does not exactly solve the problem, but it satisfies

$$\min_{q_i \in \pi_2} \max_{x \in [\lambda_1, \lambda_N]} |q(x)|,$$

that is, it minimizes over the interval between the smallest and largest eigenvalue. Is is an overestimate for clustered eigenvalues.

2.3 kth CG step

These are scaled and shifted Chebyshev polynomials. Use this:

$$\|q_k(A)e_0\|_A \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e_0\|_A$$

where k is the iteration and κ is the condition number.

3 For Discrete Laplacian, what is κ?

The smallest eigenvalue is $\mathcal{O}(1)$ and the largest eigenvalue is $\mathcal{O}\left(\frac{1}{h^2}\right)$ So $\kappa = \mathcal{O}\left(\frac{1}{h^2}\right)$. Doing asymptotics on the above condition gives, for large κ, that the number of iterations to converge to a tolerance of ε (relative) is

$$\sqrt{\kappa}\log(\varepsilon)$$

For the Laplacian, this is $\mathcal{O}\left(\frac{1}{h}\right)$ iterations to converge to a given tolerance. But $\mathcal{O}\left(\frac{1}{h}\right) = \mathcal{O}(n)$ and each iteration costs $\mathcal{O}(N)$. So, in 2D ($n^2 = N$), we expect the total work to be $\mathcal{O}(Nn) = \mathcal{O}\left(N^{3/2}\right)$. This is the same scaling as SOR.

4 Preconditioning CG

Let's try to cluster the eigenvalues!

$Au = f$. Big condition numbers mean slow convergence, small condition numbers mean fast convergence. Let's multiply through by an invertible matrix:

$$M^{-1}Au = M^{-1}f$$

These two problems have the same solution. The spectrum of the matrix $M^{-1}A$ is different than the spectrum of A. The hope is that this is better conditioned. For CG, I need/want M to be

1. s.p.d.
2. $M^{-1}A$ is better conditioned
3. M^{-1} easy to apply, i.e. $Mx = b$ is easy to solve. We don't need the matrix. We just need to apply it.

A good preconditioner M^{-1} approximates A^{-1}.

In general, $M^{-1}A$ is not symmetric.

$$\begin{aligned} Au &= f \\ B^{-1}Au &= B^{-1}f \\ B^{-1}AB^{-T}B^Tu &= B^{-1}f \end{aligned}$$

Define $\tilde{A} = B^{-1}AB^{-T}$, $\tilde{u} = B^Tu$, $\tilde{f} = B^{-1}f$, so

$$\tilde{A}\tilde{u} = \tilde{f}$$

We see $\tilde{A}$ is symmetric. Also,

$$y^T B^{-1} A B^{-T} y = \left(B^{-T} y\right)^T A \left(B^{-T} y\right) \geq 0$$

for $y \neq 0$, i.e. $\tilde{A}$ is s.p.d. Next,

$$\begin{aligned} B^{-T} B^{-1} A B^{-T} B^T &= B^{-T} B^{-1} A \\ &= \left(B B^T\right)^{-1} A \\ &= M^{-1} A \end{aligned}$$

where $M \coloneqq BB^T$. So $\tilde{A}$ has the same eigenvalues as $M^{-1}A$ where $M = BB^T$.

So, we write CG in $\tilde{}$ variables and transform back to original variables. So,

$$u_k = B^{-T} \tilde{u}_k, \qquad p_k = B^{-T} \tilde{p}_k, \qquad r_k = B \tilde{r}_k \tag{1}$$

Then B and B^{-T} drop out of the algorithm because:

$$\begin{aligned} \tilde{p}_{k+1} &= \tilde{r}_{k+1} + \beta_k \tilde{p}_k && (2) \\ B^T p_{k+1} &= B^{-1} r_{k+1} + \beta_k B^T p_k && (3) \\ B^{-T} B^T p_{k+1} &= B^{-T} B^{-1} r_{k+1} + \beta_k B^{-T} B^T p_k && (4) \\ p_{k+1} &= M^{-1} r_{k+1} + \beta_k p_k && (5) \end{aligned}$$

4.1 PCG Algorithm

- Initialize residual $r_0 = f - Au_0$
- Solve $Mz_0 = r_0$ or compute $z_0 = M^{-1} r_0$ (either we have M^{-1} or just apply V-cycle or whatever to $z_0 = Mr_0$)
- loop in k
  - $w_k = Ap_k$
  - $\alpha = \dfrac{z_k^T r_k}{p_k^T w_k}$ $\leftarrow$ different from before
  - $u_{k+1} = u_k + \alpha p_k$
  - $r_{k+1} = r_k - \alpha w_k$
  - check $\|r_{k+1}\|$ for breaking out of the loop
  - compute $z_{k+1} = M^{-1} r_{k+1}$ $\leftarrow$ different from before
  - compute $\beta = \dfrac{k_{k+1}^T r_{k+1}}{z_k^T r_k}$ $\leftarrow$ different from before
  - $p_{k+1} = z_{k+1} + \beta p_k$ $\leftarrow$ different from before

4.2 How do we pick the preconditioner M^{-1}?

- One choice is $M^{-1} = D^{-1}$ whre D is the diagonal matrix of A. For a Poisson equation with constant coefficient, this is ineffective since D^{-1} is a scalar. This brings variable coefficient problems on par with constant coefficient.
- Use other iteration schemes
  - SSOR (Symmetric SOR) Loop through SOR in both directions
  - MG (multigrid) but using a symmetric smoother (like red-black-black-red)
- Approximate factorizations (incomplete LU or incomplete Cholesky) This is just doing a little Gauss-Jordan elimination and just stopping midway. These are nice since they're algebraic, so it's a know-nothing algorithm.
- The sky's the limit.