CS 6220 Data-Sparse Matrix Computations September 19, 2017

Lecture 9: Krylov Subspace Methods

Lecturer: Anil Damle

Scribes: David Eriksson, Marc Aurele Gilles, Ariah Klages-Mundt, Sophia Novitzky

1 Introduction

In the last lecture, we discussed two methods for producing an orthogonal basis for the Krylov subspaces $\mathcal{K}_k(A, b)$

$$\mathcal{K}_k(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}$$

of a matrix A and a vector b: the Lanczos and Arnoldi methods. In this lecture, we will use the Lanczos method in an iterative algorithm to solve linear systems $Ax = b$, when A is positive definite. This algorithm is called the Conjugate Gradient (CG) algorithm. We will show that its time complexity in each step k does not depend on the number of steps preceding it, and discuss convergence of the algorithm.

2 Derivation of the Conjugate Gradient Algorithm

Consider the following optimization problem over $\mathcal{K}_k(A, b)$:

$$x_k = \arg \min_{x \in K_k(A,b)} \frac{1}{2}\|x - A^{-1}b\|_A^2.$$

Assuming A is symmetric positive definite, recall that after k steps, the Lanczos method gives us a basis for the kth Krylov $\mathcal{K}_k(A, b)$ space via

$$AV_k = V_{k+1}\tilde{T}_k.$$

Here $\tilde{T}_k$ is a square matrix given by

$$\tilde{T}_k = \begin{bmatrix} & & 0 \\ & T_k & \vdots \\ & & 0 \\ 0 \ldots 0 & & -\beta_k e_k^T \end{bmatrix}.$$

T_k is a tridiagonal matrix, and V_k has orthogonal columns. We will consider Algorithm 1 for solving $Ax = b$.

Naively, it looks like this algorithm would cost one matrix-vector multiplication Aq (to perform an iteration of Lanczos) and a trilinear solve with a matrix of size $k \times k$ thus it looks like the cost of each iteration increases with the number of steps taken k. However, one can show that there is no k dependence because of the 3 term recursion. We will now

Algorithm 1: Conjugate gradient algorithm (Explicit form)

for *step* k **do**
  Run step k of the Lanczos method
  Solve $T_k y^{(k)} = \|b\| e_1$
  Let $x^{(k)} = V_k y^{(k)}$
end

show that the added calculations at each step can also be performed in time that does not depend on k.

Observe that

$$T_{k+1} = \left[\begin{array}{c|c} & 0 \\ T_k & \vdots \\ & 0 \\ & \beta_k \\ \hline 0 \ldots 0 \ \beta_k & \alpha_{k+1} \end{array}\right].$$

Take $T_k = L_k D_k L_k^T$, an LDL^T factorization. Then we can get $T_{k+1} = L_{k+1} D_{k+1} L_{k+1}^T$ in $O(1)$ work. Furthermore, we can go from $z^{(k)} = L_k^T y^{(k)}$ to $z^{(k+1)} = L_{k+1}^T y^{(k+1)}$ in $O(1)$ work since $T_k y^{(k)} = L_k D_k L_k^T y^{(k)} = \|b\| e_1$. We have for some ζ_{k+1},

$$z^{(k+1)} = \begin{bmatrix} z^{(k)} \\ \zeta_{k+1} \end{bmatrix},$$

so we can write

$$\begin{aligned} x^{(k+1)} &= V_{k+1} y^{(k+1)} \\ &= V_{k+1} I y^{(k+1)} \\ &= V_{k+1} L_{k+1}^{-T} L_{k+1}^T y^{(k+1)} \\ &= V_{k+1} L_{k+1}^{-T} z^{(k+1)}. \end{aligned}$$

Now let $C_k = V_k L_k^{-T} = \begin{bmatrix} \mathbf{c_1} & \cdots & \mathbf{c_k} \end{bmatrix}$, where $\mathbf{c_1}, \ldots, \mathbf{c_k}$ are the columns of C_k. Then

$$\begin{aligned} x^{(k+1)} &= C_{k+1} z^{(k+1)} \\ &= C_k z^{(k)} + \mathbf{c_{k+1}} \zeta_{k+1} \\ &= x^{(k)} + \mathbf{c_{k+1}} \zeta_{k+1}. \end{aligned}$$

We can compute $\mathbf{c_{k+1}}$ and ζ_{k+1} in time that does not depend on k (in $O(n)$ in general as it involves vector multiplication). And so we can update $x^{(k)}$ to $x^{(k+1)}$ in time independent of k. I.e., the steps do not become more expensive the further into the algorithm that we go.

After some algebra, the CG can be rewritten as Algorithm 2.

Algorithm 2: Conjugate gradient algorithm (Explicit form)

$x_0 = 0$
$r_0 = b$
$p_0 = b$
for $k = 1$ *to* K **do**
  $\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle p_k, Ap_k \rangle}$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  $\beta_k = \frac{\langle r_{k+1}, r_{k+1} \rangle}{\langle r_k, r_k \rangle}$
  $p_{k+1} = r_{k+1} + \beta_k p_k$
end

This form makes it clear that the computational cost of one CG iterate is one matrix multiply, 3 vector inner products and 3 scalar vector product and 3 vector addition. It also only requires storage for 3 vectors at any iteration.

A completely equivalent way to derive CG is through the minimization of the following quadratic:

$$x_k = \arg \min_{x \in K_k(A,b)} \frac{1}{2} \|x - A^{-1}b\|_A^2. \tag{1}$$

The derivation from this point of view is included in most textbooks, e.g., see [1].

3 Convergence and stopping criteria

3.1 Monitoring residual

We now consider when to stop the conjugate gradient method. One stopping criterion is to monitor the residual

$$r_k := b - Ax^{(k)}$$

and stop when $\|r_k\|_2$ is smaller than some threshold. Notice we can't use the error metric defined in 1 as it requires to know the true solution x^*.

Observe that via Lanczos

$$\begin{aligned}
\|r_k\|_2 &= \|b - AV_k y^{(k)}\|_2 \\
&= \|V_{k+1}(\|b\|e_1) - V_{k+1}\tilde{T}_k y^{(k)}\|_2 \\
&= \left\| V_{k+1} \left(\begin{bmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} T_k \\ \beta_k e_k^T \end{bmatrix} y^{(k)} \right) \right\|_2 \\
&= \|V_{k+1}(\beta_k (e_k^T y^{(k)}) e_{k+1})\|_2 \propto \|V_{k+1}\|_2,
\end{aligned}$$

where the simplification of the vector from the second-to-last line to the last line comes from the fact that we've constructed $y^{(k)}$ in the Lanczos algorithm so that it satisfies this

(i.e., $T_k y^{(k)} = \|b\| e_1$). This shows that the kth residual is proportional to the $(k+1)$st Krylov vector.

If V_{k+1} is orthonormal, then it drops out of the 2 norm and we just get the norm of the vector. However, in floating point arithmetic, V_k may not have orthogonal columns. Even in this case we have

$$\|r_k\|_2 \leq \|V_{k+1}\|_2 |\beta_k (e_k^T y^{(k)})|.$$

and since the columns of V_k have norm one, $\|V_{k+1}\|_2$ is typically close to one even in floating point arithmetic.

3.2 Convergence

Recall that the k-th Krylov subspace of the matrix A and vector b is:

$$\mathcal{K}_k(A, b) = \text{span}\{b, Ab, A^2 b, \ldots, A^{k-1} b\}$$

Therefore,

$$x \in \mathcal{K}_k(A, b) \iff x = \sum_{j=0}^{k-1} \alpha_j A^j b = \left(\sum_{j=0}^{k-1} \alpha_j A^j \right) b = p(A) b$$

where $p(x)$ is a 1-dimensional polynomial of degree $k-1$ with coefficients $\{\alpha_j\}_{j=0}^{k-1}$. Let $x^* = A^{-1} b$, recall that the x_k, the k-th iterate of CG is chosen such that it minimizes the error in the A-norm over the k-th Krylov subspace, i.e.:

$$x_k = \arg \min_{x \in \mathcal{K}_k(A,b)} \|x - x^*\|_A.$$

Using the observation above, we can rewrite this error metric as a polynomial approximation problem:

$$\begin{aligned} \|x - x^*\|_A &= \|p(A)b - A^{-1}b\|_A \\ &= \|p(A)AA^{-1}b - A^{-1}b\|_A \\ &= \|(p(A)A - I)A^{-1}b\|_A \\ &\leq \|(I - p(A)A)\|_2 \|A^{-1}b\|_A. \end{aligned}$$

Note that if $p(x)$ is a $k-1$ degree polynomial, then $q(x) = 1 - xp(x)$ is a k degree polynomial with $q(0) = 1$. Let $\mathcal{P}_{k,0}$ be the subspace of such polynomial q, that is

$$\mathcal{P}_{k,0} = \{q(x) \mid q(x) \in \mathcal{P}_{k,0}, q(0) = 1\}$$

Furthermore, by the argument above, for all $x \in p_k$, there exist a polynomial q such that $\|x - x^*\|_A \leq \|q(A)\|_2 \|A^{-1}b\|_A$. To summarize, we have shown that:

$$\frac{\|x_k - x^*\|_A}{\|x^*\|_A} \leq \min_{q \in \mathcal{P}_{k,0}} \|(q(A))\|_2. \tag{2}$$

Consider the eigenvalue decomposition of A, $A = V\Sigma V^{-1}$. A is SPD, therefore its eigenvalues are real and positive, and $V^{-1} = V^T$. Note that

$$A^i = AA\cdots A = V\Sigma V^{-1}V\Sigma V^{-1}\cdots V\Sigma V^{-1} = V\Sigma^i V^T.$$

Therefore for any polynomial A

$$\begin{aligned} p(A) &= \sum_{i=0}^{k} \alpha_i A^i \\ &= \sum_{i=0}^{k} \alpha_i V\Sigma^i V^T \\ &= V\sum_{i=0}^{k} \alpha_i \Sigma_i V^T \\ &= Vp(\Sigma)^i V^T \end{aligned}$$

(This is a special case of the spectral theorem). Let $\Lambda(A)$ be the set of eigenvalues of A, then using this observation we can rewrite 2 as:

$$\begin{aligned} \frac{\|x_k - x^*\|_A}{\|x^*\|_A} &\leq \min_{q\in\mathcal{P}_{k,0}} \|q(A)\|_2 \\ &= \min_{q\in\mathcal{P}_{k,0}} \|Vq(\sigma)V^T\|_2 \\ &\leq \min_{q\in\mathcal{P}_{k,0}} \|V\|\|V\|^T\|q(\sigma)\|_2 \\ &= \min_{q\in\mathcal{P}_{k,0}} \|q(\Sigma)\|_2 \end{aligned}$$

where we have used that V is orthogonal, thus $\|V\|_2 = \|V^T\|_2 = 1$. Recall the definition of the spectral norm: $\|A\|_2 = \max_i \sigma_i(A)$, where σ_i are the singular values of A which coincide with the eigenvalues as A is SPD. This allows us to rewrite the bound one more time as:

$$\frac{\|x_k - x^*\|_A}{\|x^*\|_A} \leq \min_{q\in\mathcal{P}_{k,0}} \max_{\lambda\in\Lambda(A)} |q(\lambda)| \tag{3}$$

This is a remarkable result, which lets us reason about the convergence of CG in terms of the minimization of a polynomial over a set of discrete points. Much of the theory and intuition we have about the convergence of CG comes from this inequality.

This bound implies that the convergence of CG on a given problem depends strongly on the eigenvalues of A. When the eigenvalues of A are spread out it is hard to find such p, and the degree of the polynomial ($k \in \mathbb{N}$) has to be greater, which means that the algorithm requires more iterations. The figure 1 illustrates the challenge. Neither the first nor the second degree polynomial give desired results. When the eigenvalues don't form any groups, the polynomial degree may have to be as large as the number of eigenvalues of A.

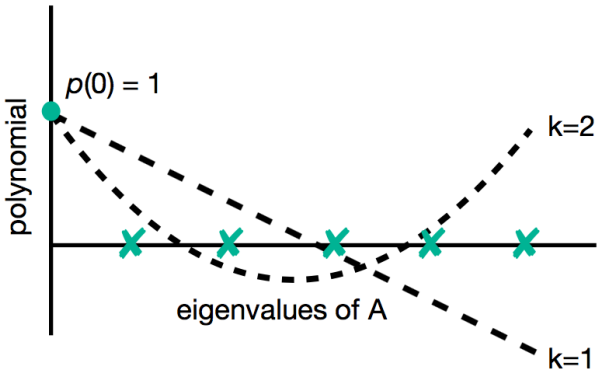

Figure 1: Polynomial fitting for A with eigenvalues spread out.

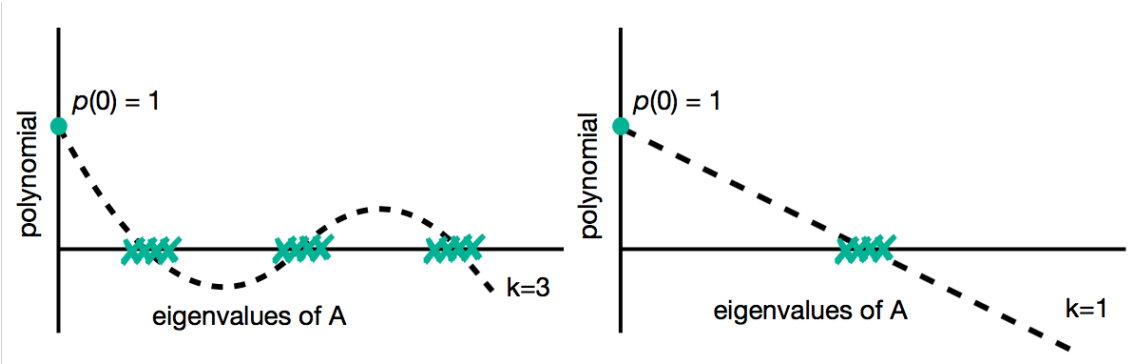

Figure 2: Polynomial fitting for clustered eigenvalues. The less clusters are present, the smaller polynomial degree k is required.

The task is easier when the eigenvalues are clustered. The fewer and tighter the clusters are, the smaller k (degree of the polynomial) is required to attain a good upper bound on algorithm error. Figure 2 shows two different cases of clustered eigenvalues. If there are three clusters as on the left plot, a third degree polynomial performs well. When all eigenvalues are in one cluster like in the plot on the right, a first degree polynomial is sufficient. This dependence on the eigenvalues of A is the true reason why the algorithm works so well in practice.

Equation (3) is a strong statement, but one that is difficult to apply. Indeed, we usually don't know the distribution of eigenvalues of our matrix A, and even if we did, the polynomial minimization problem on a discrete set is very difficult to solve. We now give another bound which is must weaker but is something we can usually estimate.

Let λ_1, λ_n be the smallest and largest eigenvalue of A respectively. Clearly, $\Lambda(A) \subset$

$[\lambda_1, \lambda_n]$. This implies that

$$\min_{q \in \mathcal{P}_{k,0}} \max_{\lambda \in \Lambda(A)} |q(\lambda)| \leq \min_{q \in \mathcal{P}_{k,0}} \max_{\lambda \in [\lambda_1, \lambda_n]} |q(\lambda)|. \tag{4}$$

The right hand side of the equation is much easier to solve than the left hand side because it involves a continuous interval. It is related to the Chebyshev polynomials $T_n(x)$ characterized by:

$$T_n(x) = \cos\left(n \arccos\left(x\right)\right) \quad \text{for } \|x\| \leq 1.$$

Lemma 1 *The minimax problem:*

$$\min_{q \in \mathcal{P}_{k,0}} \max_{\lambda \in [\lambda_1, \lambda_n]} |q(\lambda)| \tag{5}$$

is solved by the polynomial

$$P_i(x) = \frac{T_i(\frac{\lambda_n + \lambda_1 - 2x}{\lambda_n - \lambda_1})}{T_i\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right)}$$

Furthermore,

$$P_i(x) \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^i$$

where $\kappa = \frac{\lambda_n}{\lambda_1}$ (called the condition number of A).

The proof of this lemma can be found in [2]. This implies directly the following theorem:

Theorem 2 *Let x_k be the kth iterate of CG, and κ be the condition number of the matrix A, then:*

$$\frac{\|x_k - x^*\|_A}{\|x^*\|_A} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \tag{6}$$

References

[1] Gene H. Golub and Van Loan Charles F. *Matrix computation.* Johns Hopkins University Press, 1990.

[2] Jonathan Richard Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain.* 1994.