Data Sparse Matrix Computation - Lecture 11

Dongping Qi, Sujit Rao, Tianyi Shi

October 16, 2017

Contents

1 Krylov Method Review

We have already seen various methods for $Ax = b$, based on Krylov method. The convergence of these methods highly depends on the behavior of A's spectrum.

For example, for Conjugate Gradient method, if A is s.p.d and we try to solve $Ax = b$. Denote P_k as all the polynomials $p(x)$ of degree k with $p(0) = 1$ and x^* the exact solution, then CG is equivalent to the following minimization

$$\min_{p \in P_k} \|p(A)x^*\|_A^2$$

therefore it is implicitly trying to find a polynomial that is small on the spectrum of A with $p(0) = 1$. It is hard if the eigenvalues are spread out, however, it is good if the eigenvalues are clustered. In particular, we have

$$\frac{\|e^{(k)}\|_A}{\|e^0\|_A} = \inf_{p \in P_k} \frac{\|p(A)x^*\|_A}{\|e^{(0)}\|_A} \leq \inf_{p \in P_k} \max_{\lambda \in \Lambda(A)} |p(\lambda)|$$

If we let $K_2 = \lambda_{\max}/\lambda_{min}$, which is the condition number of A, then we will have

$$\frac{\|e^{(k)}\|_A}{\|e^0\|_A} \leq 2\Big(\frac{\sqrt{K_2}-1}{\sqrt{K_2}+1}\Big)^k$$

Therefore a natural idea is to precondition A to improve the convergence.

Another example is the GMRES method. With the same notation above and denote $r_k = b - Ax^{(k)}$ as the kth residue. We will have

$$\frac{\|r_k\|_2}{\|b\|_2} \leq \inf_{p\in P_k} \|p(A)\|_2$$

Assume that A is diagonalizable and $A = X\Lambda X^{-1}$, then

$$\frac{\|r_k\|_2}{\|b\|_2} \leq K_2(X) \inf_{p\in P_k} \{ \sup_{\lambda\in\Lambda(A)} |p(\lambda)|\}$$

Again we can see clearly that the convergence depends on the spectrum.

1.1 Preconditioned Method

Key idea : Pick M s.t.

- We can solve $Mx = d$ cheaply.
- $M \approx A$

Then we can solve

$$M^{-1}Ax = M^{-1}b$$

instead via a Krylov method with $\mathcal{K}(M^{-1}A, M^{-1}b)$.

The convergence now depends on the spectrum of $M^{-1}A$, i.e. $\Lambda(M^{-1}A)$ and at each step of Arnoldi's algorithm, we can apply A and solve a system with M (instead of just applying A).

1.2 How to Pick M?

There are two extreme cases about how to choose M.

- $M = A$: Great convergence in a sense ($M^{-1}A = I$), but no computational gains (still need to solve $Ax = b$).
- $M = I$: More efficiency per iteration ($Ix = d$ is easy to solve), but no improvement to convergence.

We hope that in the middle of these two extreme cases there might be something useful. In reality, this is a bit of "art" and requires problem specificity.

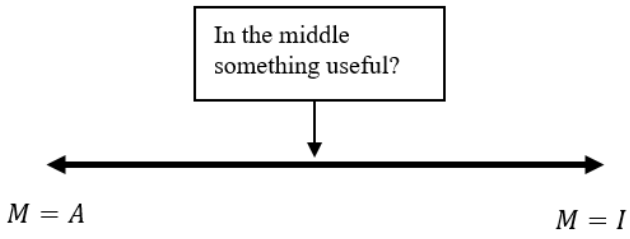

Figure 1:

1.3 Several Examples

1. Simple test: diagonal preconditioner $M = \text{diag}(A)$

   This is widely called the Jacobi preconditioner probably due to the traditional Jacobi iteration method, where we decompose $A = D + R$, D is a diagonal matrix and the iteration is constructed as

   $$x^{k+1} = D^{-1}(b - Rx^k)$$

   This preconditioning is cheap, but often not effective (it is efficient for diagonally dominant matrices).

2. Rank structured factorization.

   If A has some low rank structure, then we can factor A as $M \approx A$ to some accuracy. Solving the systems with respect to M is cheap. In some situations we can quickly compute an approximate inverse of A and apply it to a vector.

1.4 Preconditioned Conjugate Gradient method (PCG)

In this section, we will discuss about the Preconditioned Conjugate Gradient method. If A is s.p.d, choose M also s.p.d, we can still use Conjugate Gradient method with some mild modifications.

Rationale: If M is s.p.d, it has the Cholesky decomposition $M = CC^T$. Then we have

$$Ax = b \quad \Leftrightarrow \quad M^{-1}Ax = M^{-1}b \quad \Leftrightarrow \quad (C^{-1}AC^{-T})C^Tx = C^{-1}b$$

Rewrite the form into $\hat{A}x = \hat{b}$ where $\hat{A} = C^{-1}AC^{-T}$, $\hat{x}_n = C^Tx_n$ and $\hat{b} = C^{-1}b$. We can write out the Preconditioned Conjugate Gradient method explicitly. The residual will be

$$\hat{r}_n = \hat{b} - \hat{A}\hat{x}_n = C^{-1}b - C^{-1}Ax_n = C^{-1}r_n$$

Suppose that in the nth iteration the searching direction vector is d_n and let $\hat{d}_n = C^T d_n$. The Conjugate Gradient method has two important value to compute in each step. Due to definition they are

$$\hat{\alpha}_n = \frac{\hat{r}_{n-1}^T \hat{r}_{n-1}}{\hat{d}_{n-1}^T \hat{A} \hat{d}_{n-1}} = \frac{r_{n-1}^T C^{-T} C^{-1} r_{n-1}}{d_{n-1}^T A d_{n-1}^T} = \frac{r_{n-1}^T M^{-1} r_{n-1}}{d_{n-1}^T A d_{n-1}^T}$$

$$\hat{\beta}_n = \frac{\hat{r}_n^T \hat{r}_n}{\hat{r}_{n-1}^T \hat{r}_{n-1}} = \frac{\hat{r}_n^T C^{-T} C^{-1} \hat{r}_n}{r_{n-1}^T C^{-T} C^{-1} r_{n-1}} = \frac{\hat{r}_n^T M^{-1} \hat{r}_n}{r_{n-1}^T M^{-1} r_{n-1}}$$

and the new $\hat{x}_n$ and $\hat{d}_n$ can be generated. Therefore we only need to know M^{-1} during each iteration. This forms into the general upstate step for PCG. There are also several different preconditioning method and readers can refer to [1].

2 Randomized algorithms

We will focus on the construction of low-rank approximations, since this is where randomized algorithms were first used in numerical linear algebra. These can often be building blocks for larger algorithms.

Why use randomization if we have existing algorithms to find low-rank approximations? One reason is to deal with very large matrices. Also, some of the tradeoffs can make more sense for certain problems – if we have a matrix whose entries are collected from empirical data, then why should we compute a factorization to machine precision if our matrix isn't even known to machine precision? Additionally, the data flow and ordering of the matrices used in randomized algorithms can be more suitable for modern platforms and parallelism.

Randomized algorithms are often fairly new, but have close connections to classical algorithms like Krylov methods and subspace iterations. Randomization yields computational gains and sometimes robustness (which partly comes from how we analyze the algorithms).[2]

2.1 Randomized low-rank factorization

Given $A \in \mathbb{R}^{m \times n}$, we want to compute $B \in \mathbb{R}^{m \times k}$ and $C \in \mathbb{R}^{k \times n}$ such that $A \approx BC$. Classically, this costs $O(mnk)$.

We will solve this in two steps.

1. Construct a low-dimensional subspace that captures the essential information of A (using randomness).

2. Restrict A to this new subspace (resulting in a much smaller matrix) and then apply classical techniques and factorizations to this restriction. After this, we can get a factorization of A easily.

In (1), we will find a matrix Q with orthonormal columns such that $A \approx QQ^T A$. We want Q to have as few columns as possible to achieve some given accuracy.

Then given Q, we compute a truncated SVD of A. Set $B = Q^T A$, so $A \approx QB$. If Q is $n \times j$ and B is $k \times n$, and we already have what looks like a

low-rank approximation. However, we don't have an idea of what the singular values look like. The second step is to compute an SVD of $B = \widetilde{U}\Sigma V^T$. Then $\tilde{U}$ is unitary and $k \times k$, Σ is $k \times k$, and V is $n \times k$. If we set $U = Q\widetilde{U}$, then $A \simeq U\Sigma V^T$ is the desired factorization.

For this part, we're only doing dense linear algebra on B, which is $k \times n$. Thus the complexity of this part is $O(k^2 n)$, which is cheaper. (The algorithms which only make one pass over the data avoid computing $B = Q^T A$ explicitly.)

2.2 How to find such a Q

Our goal is to solve a Fixed-precision Approximation Problem here:
Given A $\in \mathbb{R}^{m\times n}$ and accuracy ϵ, find Q with $k = k(\epsilon)$ orthonormal columns such that $||(I - QQ^T)A||_2 \leq \epsilon$. That means, the range of Q is a k-dimensional subspace that captures most of the action of A, with k as small as possible.

We know from previous exposure to singular values of matrix that if A have singular values σ_1, σ_2, ..., $\sigma_{\min(m,n)}$, then
$\min_{rank(X)\leq j} ||A - X||_2 = \sigma_{j+1}$.
Therefore the optimal solution is $X = QQ^T A$ with Q as first j left singular vectors, and we can choose the minimal $k(\epsilon)$ to be the number of σ's greater than ϵ.

Since it is hard to solve a Fixed-precision Approximation Problem directly, we first solve an easier problem, the Fixed-rank approximation problem:
Given k and oversampling parameter p, find $Q \in \mathbb{R}^{m\times(K+P)}$ with orthonormal columns such that
$||A - QQ^T A||_2 \approx \min_{rank(X)\leq k} ||A - X||_2 \leq c_1 \cdot \sigma_{K+1} + c_2 \cdot (\sum_{j>K} \sigma_j^2)^{1/2}$.
When $p > 0$, we have an opportunity to use a small number of additional columns that provide a flexibility that is crucial for the effectiveness of the computational methods we discuss.

The Fixed-rank approximation can be adapted to solve Fixed-accuracy approximation problem. We can build the basis matrix Q incrementally and at any point of the computation, inexpensively estimate $||A - QQ^T||_2$. We will show one adaptive algorithm later in Section 2.4.

2.3 How to construct Q with randomness

The remarks in this section mainly come from [2, section 1 and section 4].
We can follow the following steps to construct Q with randomness:

1. Draw a random vector $w \in \mathbb{R}^{n\times 1}$, then $y = Aw$ is a sample from the range of A.

2. Repeat this k times $y^{(i)} = Aw^{(i)}$.

   There are some things we note in this second step:

   - Generically, $y^{(i)}$ are linearly independent.

- If A were exactly rank k, then an orthonormal basis for $y^{(1)}$, $y^{(2)}$, $\ldots$, $y^{(k)}$ gives us an orthonormal basis for the range of A.

- More realistically, $A = B + E$, where A is of rank k and E is a small perturbation; now $y^{(i)}$ do not span the range of A, but using $k + p$ samples suffices to capture the range of A, for a fixed small number of p.

- We also have some rules of choosing p: "very large matrices need large p; the more rapid the decay of the singular values, smaller p will be needed; Gaussian matrices succeed with very small p, but are not always the most cost-effective option" [2].

- Remarkably, "for certain types of random sampling schemes, the failure probability decreases superexponentially with the oversampling parameter p" [2]; in practice, setting $p = 5$ or $p = 10$ often gives superb results.

3. Let the $y^{(i)}$'s above form a random matrix Ω, form a matrix $Y = A\Omega$ and construct a matrix Q whose columns form an orthonormal basis for the range of Y, say doing a QR factorization $Y = QR$.

The third step has a bottleneck that it takes $O(mnl)$ to compute the matrix product $A\Omega$ when Ω is standard Gaussian (which is our basic choice for this randomness method). To make it quicker, we can use a structured random matrix that allows us to compute the product in $O(mn\log(l))$ flops. One of the simplest structured random matrix is the Subsampled Random Fourier Transform, and readers can refer to [2] on how to construct a SRFT Ω. In this way, we can use subsampled FFT to make our range finder algorithm in $O(mn\log(l))$.

Now we state a Lemma that can help with understanding the error bound of adaptive methods with Gaussian random vectors.

Lemma 1 *Let B be a real $m \times n$ matrix. Fix a positive integer r and a real number $\alpha > 1$. Draw an independent family $\{w^{(i)} : i = 1, 2, \ldots, r\}$ of standard Gaussian vectors. Then*
$||B|| \leq \alpha\sqrt{\frac{2}{\pi}}\max_{i=1\ldots r}||Bw^{(i)}||_2$ *except with probability α^{-r}.*

Then setting $B = (I - QQ^T)A$ and $\alpha = 10$ we can get an error estimation that will be covered in future lectures:
$||(I - QQ^T)A||_2 \leq 10\sqrt{\frac{2}{\pi}}\max_{i=1\ldots r}||(I - QQ^T)Aw^{(i)}||_2$. with probability at least $1 - 10^{-r}$

2.4 An adaptive randomized range finder algorithm

We introduce here an adaptive randomized range finder algorithm that solves the Fixed-precision Approximation Problem with the help of an algorithm that

solves the simpler Fixed-rank approximation problem. This is Algorithm 4.2 in [2].

1. Draw standard Gaussian vectors $w^{(1)}, \ldots, w^{(r)}$ of length n.
2. For $i = 1, 2, \ldots, r$, compute $y^{(i)} = Aw^{(i)}$.
3. $j = 0$.
4. $Q^{(0)} = []$, the $m \times 0$ empty matrix.
5. while $\max(||y^{(j+1)}||, ||y^{(j+2)}||, \ldots, ||y^{(j+r)}||) > \epsilon/(10\sqrt{2/\pi})$,
   - $j = j + 1$
   - Overwrite $y^{(j)}$ by $(I - Q^{(j-1)}(Q^{(j-1)})^{(T)})y^{(j)}$.
   - $q^{(j)} = y^{(j)}/||y^{(j)}||$.
   - $Q^{(j)} = [Q^{(j-1)}q^{(j)}]$.
   - Draw a standard Gaussian vector $w^{(j+r)}$ of length n
   - $y^{(j+r)} = (I - Q^{(j)}(Q^{(j)})^{(T)})Aw^{(j+r)}$
   - for $i = (j+1), (j+2), \ldots, (j+r-1)$: overwrite $y^{(i)}$ by $y^{(i)} - q^{(j)}\langle q^{(j)}, y^{(i)}\rangle$.
6. $Q = Q^{(j)}$.

2.5 Example of implementation of the adaptive range approximation method

In [2, section 7.1], the authors perform an experiment to test the behavior of the adaptive method shown in Section 2.4.

The writers consider a 200×200 matrix A that results from discretizing the following single-layer operator associated with the Laplace equation:
$[S\sigma](x) = const \cdot \int_{\Gamma_1} \log|x - y|\sigma(y)dA(y), x \in \Gamma_2$ where Γ_1 and Γ_2 are the two contours in $\mathbb{R}^2$.

Then they approximate the integral with the trapezoidal rule and estimate that the discretization error would be less than 10^{-20} for a smooth source σ and they implement the Algorithm in Matlab v6.5. Quantities, observations

and conclusions below are all direct quotes from [2].

For each number l of samples, they compare the following three quantities:

1. The minimum rank-l approximation error σ_{l+1}.

2. The actual error $e_l = ||(I - Q^{(l)}(Q^{(l)})^{(T)})A||$.

3. A random estimator f_l for the actual error e_l obtained from the error bound given above with the parameter r set to 5.

From the experiment they make three observations:

1. The error e_l incurred by the algorithm is remarkably close to the theoretical minimum σ_{l+1}.

2. The error estimate always produces an upper bound for the actual error. Without the built-in $10\times$ safety margin, the estimate would track the actual error almost exactly.

3. The basis constructed by the algorithm essentially reaches full double-precision accuracy.

The authors' conclusion is that the trial above is typical. They examine the empirical performance of the algorithm over 2000 independent trials and offer four observations: (i) The initial run detailed above is entirely typical. (ii) Both the actual and estimated error concentrate about their mean value. (iii) The actual error drifts slowly away from the optimal error as the number l of samples increases. (iv) The error estimator is always pessimistic by a factor of about ten, which means that the algorithm never produces a basis with lower accuracy than requested. The only effect of selecting an unlucky sample matrix Ω is that the algorithm proceeds for a few additional steps.

References

[1] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Matrix Computations. Johns Hopkins University Press, 2012.

[2] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.