Notes for CS 6220 Lecture 15

Heather Wilber and Lifan Wu

Oct. 12 2017

A single pass randomized algorithm for the approximate eigenvalue decomposition

Recall that in Lecture 13, a randomized algorithm was described for computing a low rank approximation to the eigendecomposition of a matrix A. A drawback to this method is that the matrix A must be accessed multiple times (twice), which may not be possible in streaming models where A cannot be stored in memory [1]. For the streaming model, we require a *single pass* algorithm, where A is accessed only once. To derive this algorithm, we begin with a brief review of the multipass method, and work under the assumption that $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix.

Algorithm 1 The multipass method

1: Using the range-finding algorithm described in Lecture 13, we first find $Q \in \mathbb{C}^{n\times k}$, $k \ll n$, so that $A \approx QQ^TA$. This is the first time A is accessed.
2: Set $B = Q^TAQ$. This is the second time A is accessed.
3: Find the eigendecomposition $B = V\Lambda V^T$.
4: Set $X = QV$.
5: $A \approx X\Lambda X^T$.

By eliminating multiplication by A in Step 2 of the algorithm, we can develop a *single pass* method of computing the eigendecomposition, where access to A is only required once.

The single pass method.

To eliminate A in Step 2, we examine the matrix Q more closely. The range-finding algorithm in Step 1 proceeds by selecting a Gaussian random matrix $\Omega \in \mathbb{R}^{n\times k}$. Then, the matrix $Y = A\Omega$ is formed, and Q is found by computing the thin QR factorization $Y = QR$.

Since $B = Q^TAQ$, we have that $BQ^T\Omega = Q^TAQQ^T\Omega$. Using the fact that A is symmetric, it follows that $A \approx AQQ^T$, so $BQ^T \approx Q^TA\Omega = Q^TY$. We can replace Step 2 in the above algorithm with the following least squares problem:

$$\tilde{B} = \operatorname{argmin} \left\| BQ^T\Omega - Q^T\Omega Y \right\|_F, \qquad \text{subject to } \tilde{B} = \tilde{B}^T, \tag{1}$$

which has no explicit dependence on A. This results in the following algorithm:

Algorithm 2 The single pass method

1: Select the Gaussian random matrix $\Omega \in \mathbb{C}^{n\times k}$. Set $Y = A\Omega$ and find $Y = QR$, so that $A \approx QQ^TA$.
2: Solve $\tilde{B} = \operatorname{argmin}\|BQ^T\Omega - Q^T\Omega Y\|_2$, with $\tilde{B} = \tilde{B}^T$.
3: Find the eigendecomposition $\tilde{B} = \tilde{V}\tilde{\Lambda}\tilde{V}^T$.
4: Set $\tilde{X} = Q\tilde{V}$.
5: $A \approx \tilde{X}\tilde{\Lambda}\tilde{X}^T$.

Improving the accuracy.

In Algorithm 2, we have traded the exact (in infinite precision) eigendecomposition of $B = Q^T AQ$ for an approximate eigendecomposition satisfying the least squares problem, and this results in some loss of accuracy. The situation is particularly bad if $Q^T\Omega$ is ill-conditioned. Specifically, the error $\|A - \tilde{X}\tilde{\Lambda}\tilde{X}^T\|_2$ can be worse than the error produced via Algorithm 1 by a factor of $1/\sigma_{min}$, where σ_{min} is the smallest singular value of $Q^T\Omega$. This issue can be alleviated in the following way: If k is the target rank, choose a small oversampling factor p and draw a Gaussian random matrix $\Omega \in \mathbb{R}^{n\times(k+p)}$. Form $Y = A\Omega$, as before, but find $Y = QR$ and set $\tilde{Q} = Q_{(\,:\,,1:k)}$, i.e., choose $\tilde{Q}$ as the first k columns of Q. Then, the matrix $Q^T\Omega$ is of size $k \times (k+p)$, so that the linear system (1) is overdetermined [2].

The non-symmetric case

A single pass eigendecomposition algorithm can also be devised when A is not symmetric. In this setting, we require bases for the ranges of A and A^T. Let $Y = A\Omega_1$, and select $W = A^T\Omega_2$, where Ω_1 and Ω_2 are each Gaussian random matrices. Then, find the thin QR factorizations $Y = Q_yR_y$ and $W = Q_wR_w$, so that $A \approx Q_yQ_y^TAQ_wQ_w^T$. We now require an approximation to the matrix $B = Q_y^TAQ_w$, and observe that

$$Q_y^TY \approx BQ_w^T\Omega, \qquad Q_w^TW \approx B^TQ_y^T\Omega_w. \tag{2}$$

The least squares problem in (1) must be replaced by the system of equations in (2) for which a minimum residual solution $B^* \approx B$ is sought [2].

An introduction to the LSRN (least squares random normal) method

The LSRN is another application of randomized linear algebra. It is used to solve linear least squares problems which are over or underdetermined. The LSRN method was proposed in [3] in 2014 as a fast and accurate way to solve these kinds of problems. Code and related papers are available at `http://web.stanford.edu/group/SOL/software/lsqr/`

Problem setting.

We want to solve find the min-length solution to

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2\,, \tag{3}$$

given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, with $m \gg n$ or $m \ll n$, where $\|\cdot\|_2$ is the Euclidean norm. Notice that the minimization problem (3) may have many solutions. What we are interested is the one with minimum length, i.e., with the smallest 2 norm. Therefore, we may view this optimization problem as

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & ||x||_2 \\ \text{subject to} \quad & x \in \arg\min ||Ax - b||_2 \end{aligned} \tag{4}$$

In the following discussion, we consider the case where $m \gg n$ (so the linear system is over-determined), and assume the matrix A has full column rank (i.e., $\text{rank}(A) = n$).

Deterministic ways to solve the least squares problem.

In this section, we review two ways to solve the least squares problem (4).

The SVD approach.

One traditional approach to solve (4) is to use the singular value decomposition (SVD) of A. If $A = U\Sigma V^T$ is the reduced (truncated) SVD of the matrix A, where $U \in \mathbb{R}^{m\times n}$, $\Sigma \in \mathbb{R}^{n\times n}$, and $V \in \mathbb{R}^{n\times n}$, then $x^* = V\Sigma^{-1}U^Tb = A^+b$ solves the least squares problem, where $A^+ := V\Sigma^{-1}U^T$ is the Moore-Penrose pseudo-inverse of A.
More generally, we can use any complete orthogonal factorization of A to compute x^*; the SVD is not the only option.

This approach is accurate, but if A is sparse, then we may not want to compute the SVD, and it may be beneficial to reduce the size of our problem first. This leads to our second method.

The normal equations approach.

Instead of solving the original least squares problem, we look for the min-length solution to the normal equations

$$A^TAx = A^Tb. \tag{5}$$

The solution x^* also solves the original least squares problem.
We could solve the problem directly, observing that $x^* = (A^TA)^+A^Tb$. The Cholesky factorization of A^TA yields the desired result. Though less expensive than SVD, this approach is less accurate. For example, the ill-conditioning of A is amplified when we deal with A^TA.
Alternatively, we can use iterative methods to solve (5). Since we assumed $\text{rank}(A) = n$, it follows that the matrix A^TA is positive definite, and the conjugate gradient (CG) method can be applied. Given A and b, the LSQR method is mathematically equivalent to applying CG on A^TA and A^Tb, but it improves on this idea by eliminating the explicit computation of A^TA.

Preconditioning for least squares systems.

Loosely speaking, the rate of convergence for an iterative method solving the normal equations depends on the spectrum of A^TA. Recall from previous lectures that we have

$$\frac{\|x^{(k)} - x^*\|_{A^TA}}{\|x^{(0)} - x^*\|_{A^TA}} \leq 2\left(\frac{\kappa(A^TA) - 1}{\kappa(A^TA) + 1}\right)^k, \tag{6}$$

where $x^* = A^+b$ is the true solution, and $\kappa(A^TA) = \|A^TA\|_2 \, \|(A^TA)^{-1}\|_2$ is the condition number of A^TA. Unless A^TA is well–conditioned, the performance of iterative method is uncontrollable. If, by any means, we are able control the condition number, then we can find a guarantee on the number of iterations required for convergence. In the LSRN scheme, a preconditioner is selected in a manner that controls the condition number of the normal equations. This preconditioner is built using randomness, and is discussed in greater detail in Lecture 16. Here, we give an outline of the method.

Right preconditioning.

For tall, skinny matrices ($n \ll m$), we precondition from the right. For the case where $m \ll n$, left preconditioning is used instead. In general, the idea behind a right-preconditioner is to choose a matrix $N \in \mathbb{R}^{n \times s}$ and solve the (better conditioned) least squares problem $ANy = b$ for y, and then set $x = Ny$.
In addition to selecting N so that the resulting system is better conditioned, we also require that range(N) = range(A^T). This ensures that for the true least squares solution x^*_{right}, we have

$$x^*_{right} = N(AN)^+ b = x^* = A^+ b.$$

We include pseudocode for the LSRN algorithm below; additional details are available in Lecture 16 and in [3].

Algorithm 3 The LSRN algorithm for computing $\hat{x} \approx A^+ b$ when $m \gg n$

1: Choose the oversampling parameter $\gamma > 1$, and set $s = \lceil \gamma n \rceil$.
2: Generate $s \times m$ matrix G with iid $N(0, 1)$ entries.
3: Set $\tilde{A} = GA$.
4: Compute the reduced SVD of $\tilde{A}$: $\tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^T$.
5: Let $N = \tilde{V}\tilde{\Sigma}^{-1}$.
6: Compute the min-length solution to $\min_y ||ANy - b||_2$, via, eg. LSQR.
Denote the solution as $\hat{y}$.
7: Return $\hat{x} = N\hat{y}$.

References

[1] Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.

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

[3] Xiangrui Meng, Michael A Saunders, and Michael W Mahoney. LSRN: a parallel iterative solver for strongly over-or underdetermined systems. *SIAM J. Sci. Comput.*, 36(2):C95–C118, 2014.