EE 381V: Large Scale Learning **Spring 2013**

Lecture 17 — March 19

Lecturer: Caramanis & Sanghavi *Scribe: Xinyang Yi*

17.1 Quick Review

In the previous lectures, we mainly talked about algorithms and related theorem in machine learning. Linear algebra based methods play an important role in multiple cases. For large scale problems in machine learning, computation cost becomes a critical issue. For instance, in linear methods, suppose $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, cost of multiplication Ax is $\mathcal{O}(m \times n)$; $A \in \mathbb{R}^{nn}$, cost of multiplication AA is $\mathcal{O}(n^3)$. For some other operations like inversion, SVD, eigenvalue decomposition, the cost is generally $\mathcal{O}(n^3)$.

We also introduce some iterative methods such as Lanczos' algorithm in order to get approximate solutions. Note that these algorithms are still deterministic. Characteristics of these methods include:

1. Inexpensive cost in single iteration.
2. Being able to exploit special structure:matrix sparsity.
3. Work less hard but still get theoretical guarantee.

From this lecture, we will start to discuss *Randomized linear Methods* which is able to exploit special structure.

17.2 Randomized Linear Algebra

In particular, we concerned a factorization approximation problem. Given a matrix $A \in \mathbb{R}^{m\times n}$, its SVD could be denoted as $A = U\Sigma V^*$. We are willing to do this approximately. Here, we assume that A is low rank. Suppose there exists a matrix $Q \in \mathbb{R}^{m\times k}$ such that $rank(A) \approx rank(Q)$, $k \ll m, n$. If $A \approx QQ^*A$, then after doing SVD on Q, the left k singular values of Q will be close to left k singular values of A.

Several most common randomized approximation schemes have been introduced in last lecture:

1. Sparsification: $A \to Y = A_\Omega$, where the set Ω is randomly chosen.
2. Column Selection: $A \to Y = [A_{i_1}, A_{i_2}, ..., A_{i_l}]$. The columns are randomly chosen.
3. Dimention Reduction: $A \to Y = A \times \Omega$, where Ω is a low rank matrix with random entries.

4. Approximation by submatrices.

Generally, the peformance of algorithms depends on following points:

1. Structure of data. For instance, the way data is stored and presented/ the sparsity of data.

2. Computing resources available. For instance, the speed of processors w.r.t speed of read and write/computation parallization.

The main reference for this lecture is [1].

17.2.1 Linear Algebra Background

QR-factorization

Any real square matrix A may be decomposed as

$$A = QR \tag{17.1}$$

where Q is an orthogonal matrix ($Q^TQ = I$) and R is an upper triangular matrix (also called right triangular matrix).

Partial Decomposition

For partial decomposition, instead of doing exact QR factorization, we may be interested in approximate solution: $A = QR + E$,where $\|E\|_F$ is small. Note that Lanczos algorithm is in this category.

17.2.2 Randomized algorithm

The basic problem of designing randomized algorithm is that how can we find a l-column matrix Q so that $A \approx QQ^*A$, i.e., $\|(I - QQ^*A)\| \leq \epsilon$. How well can we do for a fixed l? How can a randomized solution do as well by allowing $l = k + p > k$?

A first algorithm: Dimension Reduction

Step 1: Draw a Gaussian random matrix $\Omega \in \mathbb{R}^{n\times l}$.
Step 2: Compute $Y = A\Omega$. $Y \in \mathbb{R}^{m\times l}$
Step 3: Do QR factorization on Y: $Y = QR$.

Note that if $l \ll n$, the computation cost is $\mathcal{O}(mn)$, i.e., the cost of QR factorization could be omitted. Especially, we are interested in the following two types error bound:

1. A priori error bound

2. A posterior error bound

Computing $\|(I - QQ^*)A\|$ is expensive.Note that the operator norm can also be expressed as

$$\sup_{\|w\|=1} \|(I - QQ^*)Aw\| \tag{17.2}$$

We have the following theorem.

Theorem 17.1.

$$\|(I - QQ^*A)\| \leq 10\sqrt{\frac{2}{\pi}} \underset{i=1,2,...,r}{max} \|(I - QQ^*Aw^{(i)})\| \tag{17.3}$$

with probability at least $1 - 10^{-r}$. *Where* $w^{(1)}, w^{(2)}, ..., w^{(r)} \sim N(0, 1)$

Priori bound: Let Ω be random matrix

$$A = U \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} \tag{17.4}$$

$\Omega_1 = V_1^*\Omega$, $\Omega_2 = V_2^*\Omega$

Theorem 17.2.

$$\|A - QQ^*A\|_*^2 \leq \|\Sigma_2\|_*^2 + \|\Sigma_2\Omega_2\Omega_1^*\|_*^2 \tag{17.5}$$

where $\|\cdot\|_* = \|\cdot\| or \|\cdot\|_F$

Further, we assume $\Omega \sim Guassian$ with $l = k + p$ columns, we have:

1.
$$\mathbb{E}\|A - QQ^*A\|_F \leq (1 + \frac{k}{p-1})^{\frac{1}{2}} (\sum_{j>k} \sigma_j^2)^{\frac{1}{2}} \tag{17.6}$$

2.
$$\mathbb{E}\|A - QQ^*A\| \leq (1 + \sqrt{\frac{k}{p-1}})\sigma_{k+1} + \frac{c\sqrt{k+p}}{p} (\sum_{j>k} \sigma_j^2)^{\frac{1}{2}} \tag{17.7}$$

Another approach: Matrix Subsampling

Instead of using $Y = A\Omega$, we choose $Y = samples\ of\ A's\ columns$, i.e.,

$$Y = [\overline{A_{i_1}}, \overline{A_{i_2}}, ..., \overline{A_{i_l}}] \tag{17.8}$$

$\overline{A_{i_k}}$ is the rescaled A_{i_k}. Roughly, we will show how to rescale A's columns. Let $P_i = \frac{\|A_i\|^2}{\|A\|_F^2}$ which is the probability that *ith* column of A is chosen.

$$\|Y_i\|^2 = \frac{\|A_i\|^2}{l \times P_i} = \frac{\|A\|_F^2}{l} \tag{17.9}$$

We can show that $\mathbb{E}[YY^*] = AA^*$. Note that $YY^* = \sum_{i=1}^{l} Y_i Y_i^*$,

$$\mathbb{E}[YY^*] = l\mathbb{E}[Y_i Y_i^*] = l \sum P_i \frac{A_i A_i^*}{l P_i} = \sum A_i A_i^* = AA^* \tag{17.10}$$

Sparsification: another way to do sampling

Algorithm:
Input: A
For $i = 1, 2, ..., m$, $j = 1, 2, ..., n$

$$Y_{ij} = \begin{cases} A_{ij}/P_{ij}, & with\ probability\ P_{ij} \\ 0, & otherwise \end{cases} \tag{17.11}$$

Output:top singular value of Y.
Remark:

1. Y is sparse which means faster performance of iterative algorithm(Power methods, Lancoz).
2. Note $Y = \sum_{i,j} Y_{ij} e_i e_j^*$, Y is sum of random matrices.

$$\mathbb{E}[Y] = \sum_{i,j} P_{ij}(\frac{A_{ij}}{P_{ij}}) e_i e_j^T = A \tag{17.12}$$

P_{ij} **Selection**

$$Var(Y_{ij}) = \frac{1 - P_{ij}}{P_{ij}} A_{ij}^2 \approx \frac{1}{P_{ij}} A_{ij}^2 \tag{17.13}$$

One idea is based on minimization of the max variation. $P_{ij} \propto A_{ij}$. Thus if $|A_{ij}| \approx |A_{i'j'}|$, then $P_{ij} \approx P_{i'j'}$. The actual choice could be $\tau_{ij} = (\frac{A_{ij}}{A_{max}})c$, $P_{ij} = max\tau_{ij}, \sqrt{c(n)\tau_{ij}}$, where $c(n) = \frac{(8 \log n)^4}{n}$. Regarding the performance of sparsification, we have the following theorem:

Theorem 17.3. *With probability at least* $1 - \delta$,

$$\|A - QQ^*A\|_2^2 \leq \|A - A_k\|_2^2 + 16\sqrt{\frac{\log(2/\delta)}{l}} \|A\|_F^2 \tag{17.14}$$

Note that the first term on the right hand side is the best possible error.

17.3 Reference

[1]Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." arXiv preprint arXiv:0909.4061 (2009).