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

Lecture 25 — April 16

Lecturer: Caramanis & Sanghavi *Scribe: Jeff Manning*

25.1 Data Streaming Model

This model is applied to the same kinds of problems as we have covered already in class, but when the data objects (matrices and matrix elements) are too big to fit in memory, or are otherwise presented to us serially.

25.2 Two Models

25.2.1 Model A: Multi-pass

Data fits on disk (e.g. server on a network) but does not fit in local RAM. Disk (or network) access is expensive in comparison with processing power, and becomes the throughput bottleneck.

Model:

- Small main memory
- Infinite tape but expensive to access
- Must read the data left-to-right in sequential order. The goal herein is to read through the data as few times as possible, ideally executing O(1) operations per datum.

Metrics:

- Number of passes through the data
- Amount of memory required
- Computation time

25.2.2 Model B: Single-pass

Data must be processed sequentially, as if coming directly from a sensor, e.g. a camera.

Metrics:

- Amount of memory required
- Computation time
- Sample Complexity (especially relevant when we have a generative model, e.g. PCA)

If x, w are Gaussian, then $y_i = \mathcal{N}(0, AA^T + I)$, where $y_i \in \mathbb{R}^p$. We need to estimate range(A). The goal here is to determine how many points are needed to do so. Produce the covariance matrix of y, given by

$$\frac{1}{n}\sum y_i y_i^T,$$

(which is (p x p), requiring storage $O(p^2)$), then compute its SVD and project it into k principal components, which will require $O(kp)$ operations. Therefore, an example image of 10Mb, where each y_i is a (1 x 10^7) vector, has $p^2 = 10^{14}$. A computer with 100Tb of memory looks much different (and costs a bit more) than one with 10Mb! We are consequently compelled to seek a lower-complexity approximate solution.

25.3 Efficient Approximate Matrix Multiplication

Given two matrices A and B, our goal is to produce the matrix C, such that

$$\left\|A^T B - C\right\|_F < \epsilon \left\|A\right\|_F \left\|B\right\|_F.$$

How much memory is required to compute the approximation C? Henceforth, we will derive an upper bound for the amount of memory required, and then use the principle of Communication Complexity to find a lower bound as well.

25.3.1 Data Acquisition

1. Turnstile Model (like database access) - Over time, the algorithm receives arbitrary updates to the individual elements of a matrix:

$$M_{ij} \leftarrow M_{ij} + x$$

2. Another model - Algorithm receives updates to an entire row or column at a time.

25.3.2 Matrix Product using Sketches

Recall that for a large mxn matrix A, we can form

$$Y = A\Omega$$

where Ω = n x k Gaussian, which randomly projects every row of A onto a k-dimensional subspace. This works, because random projections have similar properties as Johnson-Lindenstrauss projections onto a lower dimensional subspace.

Here, we will use a "sign matrix" instead of a Gaussian random matrix, to form a "sketch" of the product of two large matrices $A^T B$.

Define S, (p x m), with

$$s_{ij} = \begin{cases} +1, & \text{prob} = 1/2 \\ -1, & \text{prob} = 1/2 \end{cases} \tag{25.1}$$

So the sketch of $A^T B$ is $S^T A S^T B$. We therefore have

$$\frac{\mathbb{E}[A^T S S^T B]}{m} = \frac{A^T \mathbb{E}[S S^T] B}{m} = \frac{A^T (mI) B}{m} = A^T B$$

The variance is given by

$$\mathbb{E}[\| \frac{A^T S S^T B}{m} - A^T B \|_F^2] \leq \frac{2}{m} \| A \|^2 \| B \|^2 \tag{2}$$

Exercise: Use the Chebyshev Inequality to show that this bound implies that

$$\forall \epsilon > 0, \exists m = O(\frac{1}{\epsilon^2})$$

such that

$$\mathbb{P}(\| [A^T S S^T B - A^T B \|^2 \leq \epsilon \| A \|^2 \| B \|^2) \geq \frac{3}{4}$$

By the Chebyshev Inequality, the probability that a value of an rv is less than $k\sigma$ from the mean is $\geq 1 - \frac{1}{k^2}$. Here we have a sketch whose expected value is $A^T B$ and variance bounded as given above. Thus we have

$$\sqrt{\epsilon} \| A \| \| B \| = k\sigma$$

but by the bound on σ,

$$\frac{c}{\epsilon} \leq 2k^2$$

for some constant c, and $k = \frac{2}{\sqrt{3}}$.

The algorithm is

1. Generate $r = O(log \frac{1}{\delta})$. Keep r pairs of sketches: $S^T A, S^T B$.
2. For each pair, store

$$p^i = \frac{(S^{(i)T} A)^T (S^{(i)T} B)^T}{m}$$

3. See if
$$\| p_j - p_i \|_F \leq \frac{\epsilon}{2} \| A \| \| B \|$$
for $> \frac{1}{2}$ of the other p_i.

One of the p_i is "good" (best). That is, pick j=1 to start, then go through and find p_j that is close to lots of the others. That p_j is the best one, so output it.

Proof: (of correctness)

1. $\exists$ a "good" p_j, and
2. the "good" p_j is *good*.

By (2), we have
$$\mathbb{P}(EventE_i) = \mathbb{P}(\| p_i - A^T B \| \leq \frac{\epsilon}{4} \| A \| \| B \|) \geq \frac{3}{4}$$
when p_i is close to $A^T B$.

Exercise: Use the Chernoff Bound: $\exists$ some r such that with probability $\geq 1 - \delta$, at least $\frac{5}{8}$ of the events E_i occur. The Chernoff bound states that for a set of r Bernoulli random variables, each with probability $> \frac{1}{2}$, the probability of more than $\frac{r}{2}$ of them having a value of 1 is
$$S = \sum_{i=\lfloor \frac{r}{2} \rfloor + 1}^{r} \binom{n}{i} p^i (1-p)^{r-i}$$
or,
$$S \geq 1 - exp(-\frac{1}{2p} r (p - \frac{1}{2})^2)$$
so, given the above,
$$r = -24 log(\delta)$$

Suppose $\geq \frac{5}{8}$ of the $E_i s$ have occurred. Then for at least $\frac{1}{2}$ of all of the $\{p_1, ..., p_r\}$, by the triangle inequality,
$$\| p_i - p_j \| \leq \| p_i - A^T B \| + \| p_j - A^T B \| \leq \frac{\epsilon}{2} \| A \| \| B \|$$

Now, we must show that we can estimate $\| A \|$and$\| B \|$ from the sketches. $\frac{1}{2}$ are close, and $\frac{5}{8}$ are good, so there are some that are both close and good ("close" means closer than the others, while "good" means within a specified bound). Here we show that our test will indeed find the "good" ones:

Suppose p_1 satisfies

$$\| p_1 - p_j \| \leq \frac{\epsilon}{2} \| A \| \| B \|$$

for more than half of the p_2, ..., p_r. Then we want to show that p_1 is also *good*:

$$\begin{aligned} \| p_1 - A^T B \| &\leq \| p_1 - p_j \| + \| p_j - A^T B \| \\ &\leq \frac{\epsilon}{2} \| A \| \| B \| + \frac{\epsilon}{4} \| A \| \| B \| \\ &= \frac{3\epsilon}{4} \| A \| \| B \| \end{aligned}$$

□

25.3.3 Algorithm Performance

We claim that the memory required for storing the random data to generate S is $O(r)$.

Theorem 25.1. *Given two (p x n) matrices A and B,*

$$m = O(\frac{1}{\epsilon^2})$$

$$r = O(log\frac{1}{\delta})$$

The algorithm succeeds with probability ≥ 1-δ and outputs a p such that

$$\| p - A^T B \| \leq \epsilon \| A \| \| B \|$$

with space requirement O(mpr), where r = the number of sketches used.

25.3.4 Finding a Lower error Bound

The foregoing establishes an upper bound on the error of our sketch-based matrix multiplication algorithm. We now wish to determine a lower bound for that error, and will use the notion of communication complexity to do so. The motivation for finding a lower bound is that it acts as a quality measure on the upper bound. That is, if we can find lower and upper

bounds which are very close to each other (or *equal*), then our algorithm must be close to (or identically) optimal. If not, then perhaps improvements can be made to it. In any case, the only way to know is to bound the error from both directions.

Communication complexity is a measure, given a certain task to be performed between two remote participants, of how much information must be exchanged between them in order to perform the task. For example, considering the two parties, Alice and Bob, presented in class and also in [2], we have the following:

Alice posesses the bit string x, of length n bits. Bob posesses a different bit string y, also n bits long. The task is for either of them to produce the output of a function f(x,y) which requires both x and y as input. Therefore, information will have to move from one of them to the other (or in the general sense, in either direction between them).

To define the communication complexity, consider that we choose some protocol P that results in a sequence of messages m_i between Alice and Bob, denoted by $s_p(x,y) = (m_1, m_2, ..., m_r)$. Further, if $|m_i|$ is the bit length of m_i, then $|s_p(x,y)| = \sum_{i=1}^{r} |m_i|$. The deterministic communication complexity is then given by

$$D(P) \triangleq \max_{(x,y)\in\{0,1\}^n \mathrm{x} \{0,1\}^n} |s_p(x,y)|$$

.

As an example, if we choose the equality function EQ(x,y) such that

$$EQ(x,y) = \begin{cases} 1, & \mathrm{x} = \mathrm{y} \\ 0, & \text{else} \end{cases} \tag{25.2}$$

The communication complexity D(EQ(x,y)) = n, for one-way communication only.

Another case, the randomized communication complexity, is where Alice transmits the parity of her message, $\langle x, z_i \rangle$, where z_i is a random vector. In this case, $\mathbb{P}(x \cdot z_i = y \cdot z_i) = \frac{1}{2}$. For a sequence of k bits, the probability is $\mathbb{P} = 2^{-k}$, and the communication complexity D $= log(n)$.

In the context of reduction, recall the turnstile model, in which Alice feeds the data x into the algorithm in some arbitrary order, that is,

$$f(x,y) = Algo(x,y)$$

which implies that

$$Storage(Algo) \geq D(f)$$

so the communication complexity forms a lower bound, and this bound can be mapped back to our original matrix multiplication problem, which will be covered in the next lecture.

Bibliography

[1] Eyal Kushilevitz, *Communication Complexity*, Dept. of Computer Science, Technion, Haifa 32000, Israel, 1997.

[2] Tamas Sarlos, *Improved approximation algorithms for large matrices via random projections*, 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 143-152, 2006.