Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 2 - 09/09/2013

Lecture 2: Approximating Matrix Multiplication

Lecturer: Michael Mahoney *Scribe: Michael Mahoney*

Warning: these notes are still very rough. They provide more details on what we discussed in class, but there may still be some errors, incomplete/imprecise statements, etc. in them.

2 Approximating Matrix Multiplication

Approximating the product of two matrices with random sampling or random projection methods is a fundamental operation that is of interest in and of itself as well as since it is used in a critical way as a primitive for many RandNLA algorithms. In this class, we will introduce a basic algorithm; and in this and the next few classes, we will discusses several related methods. Here is the reading for today.

- Drineas, Kannan, and Mahoney, "Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication"

2.1 Some notation

Before proceeding, here is some notation that we will use. For a vector $x \in \mathbb{R}^n$ we let $\|x\|_2 = \left(\sum_{i=1}^n |x_i|^2\right)^{1/2}$ denote its Euclidean length. For a matrix $A \in \mathbb{R}^{m \times n}$ we let $A^{(j)}$, $j = 1, \ldots, n$, denote the j-th column of A as a column vector and $A_{(i)}$, $i = 1, \ldots, m$, denote the i-th row of A as a row vector. We denote matrix norms by $\|A\|_\xi$, using subscripts to distinguish between various norms. Of particular interest will be the Frobenius norm which is defined by

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2}, \tag{1}$$

and the spectral norm which is defined by

$$\|A\|_2 = \sup_{x \in \mathbb{R}^n,\ x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}. \tag{2}$$

These norms are related to each other as: $\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\,\|A\|_2$. Both of these norms provide a measure of the "size" of the matrix A. There are several situations in which we will be interested in measuring the size of a matrix—e.g., the error that is incurred by a random sampling or random projection process can be viewed as a matrix, and we will be interested in showing that it is small in an appropriate norm.

2.2 Approximating matrix multiplication by random sampling

We will start by considering a very simple randomized algorithm to approximate the product of two matrices. Matrix multiplication is a fundamental linear algebraic problem, and this randomized algorithm for it is of interest in its own right. In addition, this algorithm is of interest since matrix multiplication is a primitive that is used—often "under the hood" or within the analysis—for many many other randomized algorithms for many many other matrix problems, and thus this algorithm will appear—either explicitly or implicitly, e.g., within the analysis—throughout the course.

The problem is the following: given an arbitrary $m \times n$ matrix A and an arbitrary $n \times p$ matrix B, compute, exactly or approximately, the product AB. As a starting point, the well-known three-loop algorithm to solve this problem is the following.

Algorithm 1 Vanilla three-look matrix multiplication algorithm.

Input: An $m \times n$ matrix A and an $n \times p$ matrix B
Output: The product AB

```
1: for i = 1 to m do
2:    for j = 1 to p do
3:       (AB)_ij = 0
4:       for k = 1 to n do
5:          (AB)_ik += A_ij B_jk
6:       end for
7:    end for
8: end for
9: Return AB
```

The running time of this algorithm is $O(mnp)$ time, which is $O(n^3)$ time if $m = n = p$. Note in particular that this algorithm loops over all pairs of elements in the product matrix and computes that element as a dot product or inner product between the i^{th} row of A and the j^{th} column of B.

The question of interest here is: can we solve this problem more quickly? There has been a lot of work on Strassen-like algorithms, which say that one can use a recursive procedure to decrease the running time to $o(n^3)$ time. For a range of reasons, these algorithms are rarely-used in practice. They will not be our focus; but, since some of our randomized algorithms will call traditional algorithms as black boxes, Strassen-like algorithms can be used to speed up running times of those black boxes, theoretically at least.

Here, we will consider a different approach: a randomized algorithm that randomly samples columns and rows of the matrices A and B. The key insight is that one should *not* think of matrix multiplication as looping over elements in the product matrix and computing, say, the (i, j)th element of the product matrix as the dot product between the ith *row* of A and the jth *column* of B, as is common. That is, the usual perspective is that the elements of the product matrix should be viewed as

$$(AB)_{ik} = \sum_{j=1}^{n} A_{ij}B_{jk} = A_{(i)}B^{(j)},$$

where each $A_{(i)}B^{(j)} \in \mathbb{R}$ is a number, computed as the inner product of two vectors in $\mathbb{R}^n$. Instead of this, one should think of matrix multiplication as returning a matrix that equals the sum of outer products of *columns* of A and the corresponding *rows* of B, i.e., as the sum of rank-one matrices.

Recall that

$$AB = \sum_{i=1}^{n} A^{(i)} B_{(i)}, \tag{3}$$

where each $A^{(i)}B_{(i)} \in \mathbb{R}^{m \times p}$ is an $m \times p$ rank-one matrix, computed as the outer product of two vectors in $\mathbb{R}^n$.

Viewing matrix multiplication as the sum of outer products *suggests*, by analogy with the sum of numbers, that we should sample rank-1 components, to minimize their size, according to their size. Recall that, if we were summing numbers, that we could sample (and rescale—see below) according to any probability distribution, and in particular the uniform distribution, and obtain an unbiased estimator of the sum; but that if we want to minimize the variance of the estimator that we should sample (and rescale) according to the size or magnitude of the numbers. Well, the same is true in the case of matrices. Since the role of these probabilities will be important in what follows, we will leave then unspecified as input to this algorithm, and we will return below to what probabilities should or could be used in this algorithm.

Given that background, here is our `BasicMatrixMultiplication` algorithm.

Algorithm 2 The `BasicMatrixMultiplication` algorithm.

Input: An $m \times n$ matrix A, an $n \times p$ matrix B, a positive integer c, and probabilities $\{p_i\}_{i=1}^n$.
Output: Matrices C and R such that $CR \approx AB$

1: **for** $t = 1$ to c **do**
2: Pick $i_t \in \{1, \ldots, n\}$ with probability $\mathbf{Pr}[i_t = k] = p_k$, in i.i.d. trials, with replacement
3: Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$ and $R_{(t)} = B_{(i_t)}/\sqrt{cp_{i_t}}$.
4: **end for**
5: Return C and R.

Basically, what we want to show for this algorithm is that

$$\begin{aligned} AB &= \sum_{i=1}^{n} A^{(i)} B_{(i))} \\ &\approx \frac{1}{c} \sum_{t=1}^{c} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t))} \\ &= CR. \end{aligned}$$

In particular, we will want to show that $\|AB - CR\|_\xi$ is small, for appropriate matrix norms ξ. Not surprisingly, the extent to which we will be able to do this will depend strongly on the sampling probabilities $\{p_i\}_{i=1}^n$ and the number of samples c, in ways that we will describe in detail.

For much of what follows, it will be convenient to express this and related subsequent algorithms in a standardized matrix notation that we will call the *sampling matrix formalism*. To do so, let $S \in \mathbb{R}^{n \times c}$ be a matrix such that

$$S_{ij} = \begin{cases} 1 & \text{if the } i^{th} \text{ column of } A \text{ is chosen in the } j^{th} \text{ independent trial} \\ 0 & \text{otherwise} \end{cases}$$

and let $D \in \mathbb{R}^{c \times c}$ be a diagonal matrix such that

$$D_{tt} = 1/\sqrt{cp_{i_t}}$$

With this notation, we can write the output of the `BasicMatrixMultiplication` algorithm and what it is doing as $C = ASD$, $R = (SD)^T B$, and

$$CR = ASD(SD)^T B = A\mathcal{S}\mathcal{S}B \approx AB.$$

Here, $\mathcal{S} = SD$ is just a way to absorb the diagonal rescaling matrix into the sampling matrix; we will do this often, and we will often refer to it simply as S—since one nearly always rescales in a standard way when one samples, the meaning should be clear from context.

2.3 Sampling probabilities and implementation issues

For approximating the product of two matrices, as well as for all of the other problems we will consider this semester, one can always sample uniformly at random. Unfortunately, as we will show, by doing so one would do very poorly in general, in the sense that it is very easy to construct matrices for which uniform sampling will perform extremely poorly. Thus, a central question in everything we will do will be: how should we construct the random sample?

Since $\{p_i\}_{i=1}^n$ are essentially importance sampling probabilities, informally we would like to choose samples that are more important. Quantifying this is a little subtle for some of the other problems we will consider, but for approximating the product of two matrices, it is quite easy. Recall that we are basically trying to approximate the product of two matrices by sampling randomly rank-one components in the sum Eqn. (3). Thus, by analogy with biasing oneself toward larger terms in the sum of numbers, in order to minimize variance, we would like to bias our random sample toward rank-one components that are larger. The notion of size or magnitude of a rank-one matrix that we will use is the spectral norm of the rank-one components. That is, we will choose columns of A and rows of B according to a probability distribution that is proportional to $\left\|A^{(i)}B_{(i)}\right\|_2$. Since this is a rank-1 matrix, this spectral norm expression takes a particularly simple form:

$$\left\|A^{(i)}B_{(i)}\right\|_2 = \left\|A^{(i)}\right\|_2 \left\|B_{(i)}\right\|_2.$$

Note that the norm on the left is the matrix spectral norm, while the two norms on the right are Euclidean vector norms. This equality is a consequence of the following simple lemma.

Claim 1 *Let x and y be column vectors in $\mathbb{R}^n$. Then, $\left\|xy^T\right\|_2 = \|x\|_2 \|y\|_2$.*

Proof: Recall that $\|\Omega\|_2 = \sqrt{\sigma_{max}(\Omega^T\Omega)}$, for a matrix Ω. Thus, $\left\|xy^T\right\|_2 = \sqrt{\sigma_{max}(xy^Tyx^T)} = \sqrt{\|x\|_2^2 \|y\|_2^2} = \|x\|_2 \|y\|_2$.

$\diamond$

Depending on what one is interested in proving, probabilities that depend on A in B in other ways might be appropriate, and in a few cases we will do so. But probabilities that depend on the spectral norm of the rank-one components have proven to be remarkably useful in the general area of randomized numerical linear algebra.

With respect to a few implementation issues, here are some things to note, when probabilities of different forms are used in the `BasicMatrixMultiplication` algorithm.

- Uniform sampling: one can choose which elements to keep before looking at the data, and so one can implement this algorithm in one-pass over the data.

- For nonuniform sampling, if one uses $p_i = \frac{|A^{(i)}|B_{(i)}|}{\sum_{i'=1}^{n} |A^{(i')}|B_{(i')}|}$, then one pass and $O(n)$ additional space is sufficient to compute the sampling probabilities—in that additional space, keep running totals of $|A^{(i)}|^2$ and $|B_{(i)}|^2$, for all i, and $O(n+p)$ space in the second pass can be used to choose the sample.

- For nonuniform sampling, if $B = A^T$ and one uses $p_i = \frac{|A^{(i)}|^2}{||A||_F^2}$ as the sampling probabilities, then by the select lemma one needs only $O(1)$ additional space (multiplied by the number of samples c to be selected) in the first pass to decide which samples to draw (and still need $O(m+p)$ space in the second pass to keep the sample).

Actually, the comments in the last bullet are true even if $B \neq A^T$, i.e., if we sample based on the norms-squared of A and completely ignore information in B. We will see that permitting this flexibility is very helpful in certain situations.

2.4 Initial results for approximating the product of two matrices

Here is our first result for the quality of approximation of the `BasicMatrixMultiplication` algorithm. This lemma holds for any set of sampling probabilities $\{p_i\}_{i=1}^n$. It states that CR is an unbiased estimator for AB, element-wise, and it provides an expression for the variance of that estimator that depends on the probabilities that are used.

Lemma 1 *Given matrices A and B, construct matrices C and R with the* BasicMatrixMultiplication *algorithm. Then,*

$$\mathbf{E}\left[(CR)_{ij}\right] = (AB)_{ij}$$

and

$$\mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2.$$

Proof: Fix i, j. For $t = 1, \ldots, c$, define $X_t = \left(\frac{A^{(i_t)}B_{(i_t)}}{cp_{i_t}}\right)_{ij} = \frac{A_{ii_t}B_{i_tj}}{cp_{i_t}}$. Thus,

$$\mathbf{E}\left[X_t\right] = \sum_{k=1}^{n} p_k \frac{A_{ik}B_{kj}}{cp_k} = \frac{1}{c}(AB)_{ij} \quad \text{and} \quad \mathbf{E}\left[X_t^2\right] = \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{c^2 p_k}.$$

Since by construction $(CR)_{ij} = \sum_{t=1}^{c} X_t$, we have $\mathbf{E}\left[(CR)_{ij}\right] = \sum_{t=1}^{c} \mathbf{E}\left[X_t\right] = (AB)_{ij}$. Since $(CR)_{ij}$ is the sum of c independent random variables, $\mathbf{Var}\left[(CR)_{ij}\right] = \sum_{t=1}^{c} \mathbf{Var}\left[X_t\right]$. Since $\mathbf{Var}\left[X_t\right] = \mathbf{E}\left[X_t^2\right] - \mathbf{E}\left[X_t\right]^2$ we see that

$$\mathbf{Var}\left[X_t\right] = \sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{c^2 p_k} - \frac{1}{c^2}(AB)_{ij}^2$$

and the lemma follows.

◇

Given Lemma 1, we can provide an upper bound on $\mathbf{E}\left[\,\|AB - CR\|_F^2\right]$ and note how this measure of the error depends on the p_i's.

Lemma 2 *Given matrices A and B, construct matrices C and R with the* BasicMatrixMultiplication *algorithm. Then,*

$$\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right] = \sum_{k=1}^{n} \frac{\left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2}{cp_k} - \frac{1}{c}\left\|AB\right\|_F^2. \tag{4}$$

Furthermore, if

$$p_k = \frac{\left\|A^{(k)}\right\|_2 \left\|B_{(k)}\right\|_2}{\sum_{k'=1}^{n} \left\|A^{(k')}\right\|_2 \left\|B_{(k')}\right\|_2}, \tag{5}$$

then

$$\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right] = \frac{1}{c}\left(\sum_{k=1}^{n} \left\|A^{(k)}\right\|_2 \left\|B_{(k)}\right\|_2\right)^2 - \frac{1}{c}\left\|AB\right\|_F^2. \tag{6}$$

Proof: First, note that

$$\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{p} \mathbf{E}\left[(AB-CR)_{ij}^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{p} \mathbf{Var}\left[(CR)_{ij}\right].$$

Thus, from Lemma 1 it follows that

$$\begin{aligned}
\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right] &= \frac{1}{c}\sum_{k=1}^{n}\frac{1}{p_k}\left(\sum_i A_{ik}^2\right)\left(\sum_j B_{kj}^2\right) - \frac{1}{c}\left\|AB\right\|_F^2 \\
&= \frac{1}{c}\sum_{k=1}^{n}\frac{1}{p_k}\left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2 - \frac{1}{c}\left\|AB\right\|_F^2.
\end{aligned}$$

If the value $p_k = \frac{\left\|A^{(k)}\right\|_2\left\|B_{(k)}\right\|_2}{\sum_{k'=1}^{n}\left\|A^{(k')}\right\|_2\left\|B_{(k')}\right\|_2}$ is used in this expression, then

$$\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right] = \frac{1}{c}\left(\sum_{k=1}^{n} \left\|A^{(k)}\right\|_2 \left\|B_{(k)}\right\|_2\right)^2 - \frac{1}{c}\left\|AB\right\|_F^2.$$

◇

Finally, we can provide the following statement of the manner in which the sampling probabilities of the form Eqn. (5) are optimal. Basically, they minimize the expectation of the Frobenius norm of the error, and this is equal to the sum of the variances of all of the elements of the product matrix.

Lemma 3 *Sampling probabilities $\{p_i\}_{i=1}^n$ of the form Eqn. (5) minimize $\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right]$.*

Proof: To prove that this choice for the p_k's minimizes $\mathbf{E}\left[\left\|AB-CR\right\|_F^2\right]$ define the function

$$f(p_1,\ldots p_n) = \sum_{k=1}^{n}\frac{1}{p_k}\left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2,$$

which characterizes the dependence of $\mathbf{E}\left[\|AB-CR\|_F^2\right]$ on the p_k's. To minimize f subject to $\sum_{k=1}^n p_k = 1$, introduce the Lagrange multiplier λ and define the function

$$g(p_1,\ldots p_n) = f(p_1,\ldots p_n) + \lambda\left(\sum_{k=1}^n p_k - 1\right).$$

We then have at the minimum that

$$0 = \frac{\partial g}{\partial p_i} = \frac{-1}{p_i^2}\left\|A^{(i)}\right\|_2^2 \left\|B_{(i)}\right\|_2^2 + \lambda.$$

Thus,

$$p_i = \frac{\left\|A^{(i)}\right\|_2 \left\|B_{(i)}\right\|_2}{\sqrt{\lambda}} = \frac{\left\|A^{(i)}\right\|_2 \left\|B_{(i)}\right\|_2}{\sum_{i'=1}^n \left\|A^{(i')}\right\|_2 \left\|B_{(i')}\right\|_2},$$

where the second equality comes from solving for $\sqrt{\lambda}$ in $\sum_{k=1}^{n-1} p_k = 1$. That these probabilities are a minimum follows since $\frac{\partial^2 g}{\partial {p_i}^2} > 0\ \forall i$ s.t. $\left\|A^{(i)}\right\|_2^2 \left\|B_{(i)}\right\|_2^2 > 0$.

$\diamond$

A few comments about this result.

- Many of these results are quite robust to the exact form of the sampling probabilities. For example, if they are a factor of 2 from optimal, then we get similar results if we sample a factor or 2 or so more. This flexibility will be important for what we do, and so we will provide a precise form of this result next time.
- We can use Markov's inequality to "remove the expectation" from this bound, and in some cases this will be good enough. In general, however, the sampling complexity will be bad with respect to δ, the failure probability parameter. This will be important for what we do, and so we will spend time to develop heavier-duty methods to do much better with respect to δ.
- This bound is for the Frobenius norm; we will get to the spectral norm later. Of course, the spectral norm is upper bounded by the Frobenius norm. Although we loose something in the process, in some cases this won't matter so much, in the sense that we still get somewhat meaningful results, but in other cases it will matter a lot. For this reason, we will spend time to develop heavier-duty methods to get spectral norm bounds.
- We can obtain similar but slightly weaker (typically, in terms of the concentration) if we perform the sampling with respect to one or the other matrix, but not both, or if we consider other variants of this basic algorithm. These will be important for some of our subsequent developments, and so we will revisit them later.