Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 7 - 09/25/2013

Lecture 7: Sampling/Projections for Least-squares Approximation, Cont.

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.

7 Sampling/Projections for Least-squares Approximation, Cont.

We continue with the disucssion from last time. There is no new reading, just the same as last class.

Recall that last time we provided a brief overview of LS problems and a brief overview of sketching methods for LS problems. For the latter, we provided a lemma that showed that under certain conditions the solution of a sketched LS problem was a good approximation to the solution of the original LS problem, where good is with respect to the objective function value. Today, we will focus on three things.

- Establishing goodness results for the sketched LS problem, where goodness is with respect to the certificate or solution vector.
- Relating these two structural conditions and the satisfaction of the two conditions by random sampling/projection to exact and approximate matrix multiplication algorithms.
- Putting everything together into two basic (but still slow—we'll speed them up soon enough) RandNLA algorithms for the LS problem.

7.1 Deterministic and randomized sketches and LS problems, cont.

Last time we identified two structural conditions, and we proved that if those structural conditions are satisfied by a sketching matrix, then the solution to the subproblem defined by that sketching matrix has a solution that is a relative-error approximation to the original problem, i.e., that the objective function value of the original problem is approximate well. Now, we will prove that the vector itself solving the subproblem is a good approximation of the vector solving the original problem. After that, we will show that random sampling and random projection matrices satisfy those two structural conditions, for appropriate values of the parameter settings.

Lemma 1 *Same setup as the previous lemma. Then*

$$\|x_{opt} - \tilde{x}_{opt}\|_2 \leq \frac{1}{\sigma_{min}(A)}\sqrt{\epsilon}\mathcal{Z}. \tag{1}$$

Proof: If we use the same notation as in the proof of the previous lemma, then $A(x_{opt} - \tilde{x}_{opt}) = U_A z_{opt}$. If we take the norm of both sides of this expression, we have that

$$\begin{aligned} \|x_{opt} - \tilde{x}_{opt}\|_2^2 &\leq \frac{\|U_A z_{opt}\|_2^2}{\sigma_{min}^2(A)} && (2) \\ &\leq \frac{\epsilon \mathcal{Z}^2}{\sigma_{min}^2(A)}, && (3) \end{aligned}$$

where (2) follows since $\sigma_{min}(A)$ is the smallest singular value of A and since the rank of A is d; and (3) follows by a result in the proof of the previous lemma and the orthogonality of U_A. Taking the square root, the second claim of the lemma follows.

$\diamond$

If we make no assumption on b, then (1) from Lemma 1 may provide a weak bound in terms of $\|x_{opt}\|_2$. If, on the other hand, we make the additional assumption that a constant fraction of the norm of b lies in the subspace spanned by the columns of A, then (1) can be strengthened. Such an assumption is reasonable, since most least-squares problems are practically interesting if at least some part of b lies in the subspace spanned by the columns of A.

Lemma 2 *Same setup as the previous lemma, and assume that* $\left\|U_A U_A^T b\right\|_2 \geq \gamma \|b\|_2$, *for some fixed* $\gamma \in (0, 1]$. *Then, it follows that*

$$\|x_{opt} - \tilde{x}_{opt}\|_2 \leq \sqrt{\epsilon}\left(\kappa(A)\sqrt{\gamma^{-2} - 1}\right)\|x_{opt}\|_2 . \qquad (4)$$

Proof: Since $\left\|U_A U_A^T b\right\|_2 \geq \gamma \|b\|_2$, it follows that

$$\begin{aligned} \mathcal{Z}^2 &= \|b\|_2^2 - \left\|U_A U_A^T b\right\|_2^2 \\ &\leq (\gamma^{-2} - 1)\left\|U_A U_A^T b\right\|_2^2 \\ &\leq \sigma_{\max}^2(A)(\gamma^{-2} - 1)\|x_{opt}\|_2^2 . \end{aligned}$$

This last inequality follows from $U_A U_A^T b = A x_{opt}$, which implies

$$\left\|U_A U_A^T b\right\|_2 = \|A x_{opt}\|_2 \leq \|A\|_2 \|x_{opt}\|_2 = \sigma_{\max}(A)\|x_{opt}\|_2 .$$

By combining this with eqn. (1) of Lemma 1, the lemma follows.

$\diamond$

7.2 Connections with exact and approximate matrix multiplication

7.2.1 An aside: approximating matrix multiplication for vector inputs

Before continuing with our discussion of LS regression, here is a simple example of applying the matrix multiplication ideas that might help shed some light on the form of the bounds as well as when the bounds are tight and when they are not.

Let's say that we have two vectors $x, y \in \mathbb{R}^n$ and we want to approximate their product by random sampling. In this case, we are approximating $x^T y$ as $x^T S S^T y$, where S is a random sampling matrix that, let's assume, is constructed with nearly optimal sampling probabilities. Then, our main bound says that, under appropriate assumptions on the number $c \ll n$ of random samples we draw, then we get a bound of the form

$$\left\|x^T y - x^T S S^T y\right\|_F \leq \epsilon \left\|x\right\|_F \left\|y\right\|_F ,$$

which, since we are dealing with the product of two vectors simplifies to

$$\left|x^T y - x^T S S^T y\right| \leq \epsilon \left\|x\right\|_2 \left\|y\right\|_2 .$$

The question is: when is this bound tight, and when is this bound loose, as a function of the input data? Clearly, if $x \perp y$, i.e., if $x^T y = 0$, then this bound will be weak. In the other hand if $y = x$, then $x^T y = x^T x = \left\|x\right\|_2^2$, in which case this bound says that

$$\left|x^T x - x^T S S^T x\right| \leq \epsilon \left\|x\right\|_2^2 ,$$

meaning that the algorithm provides a relative error guarantee on $x^T x = \left\|x\right\|_2^2$. (We can make similar statements more generally if we are multiplying two rectangular orthogonal matrices to form a low-dimensional identity, and this is important for providing subspace-preserving sketches.)

The lesson here is that when there is cancellation the bound is weak, and that the scales set by the norms of the component matrices are in some sense real. For general matrices, the situation is more complex, since subspace can interact in more complicated ways, but the similar ideas goes through.

7.2.2 Understanding and exploiting these structural conditions via approximate matrix multiplication

These lemmas say that if our sketching matrix X satisfies Condition I and Condition II, then we have relative-error approximation on both the solution vector/certificate and on the value of the objective at the optimum. There are a number of things we can do with this, and here we will focus on establishing a prioi running time guarantees for any, i.e., worst-case, input. But, before we get into the algorithmic details, however, we will outline how these structural conditions relate to our previous approximate matrix multiplication results, and how we will use the latter to prove our results.

The main point to note is that both Condition I and Condition II can be expressed as approximate matrix multiplications and thus bounded by our approximate matrix multiplication results from a few classes ago. To see this, observe that a slightly stronger condition than Condition I is that

$$|1 - \sigma_1(XU_A)| \leq 2^{-1/2} \quad \forall i,$$

and that U_A is an $n \times d$ orthogonal matrix, with $n \gg d$, we have that $U_A^T U_A = I_d$, and so this latter condition says that $U_A^T U_A \approx U_A^T X^T X U_A$ in the spectral norm, i.e., that

$$\left\|I - (XU_A)^T XU_A\right\|_2 \leq 2^{-1/2}.$$

Similarly, since $U_A^T b^{\perp} = 0$, Condition II says that $0 \approx U_A^T X^T X b^{\perp}$ with respect to the Frobenius norm, i.e., that

$$\left\|U_A^T X^T X b^{\perp}\right\|_F^2 \leq \frac{\epsilon}{2} \mathcal{Z}^2$$

Of course, this is simply the Euclidean norm, since $b^{\perp}$ is simply a vector. For general matrices, for the Frobenius norm, the scale of the right hand side error, i.e., the quantity that is multiplied by the ϵ, depends on the norm of the matrices entering into the product. But, the norm of $b^{\perp}$ is simply the residual value $\mathcal{Z}$, which sets the scale of the error and of the solution. And, for general matrices, for the spectral norm, there were quantities that depended on the spectral and Frobenius norm of the input matrices, but for orthogonal matrices like U_A, those are 1 or the low dimension d, and so they can be absorbed into the sampling complexity.

7.2.3 Bounds on approximate matrix multiplication when information about both matrices is unavailable

The situation about bounding the error incurred in the two structural conditions is actually somewhat more subtle than the previous discussion would imply. The reason is that, although we might have access of information such as the leverage scores (row norms of one matrix) that depend on U_A, we in general don't have access to any information in $b^{\perp}$ (and thus the row norms of it). Nevertheless, an extension of our previous discussion still holds, and we will describe it now.

Observe that the nearly optimal probabilities

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

for approximating the product of two general matrices A and B use information from both matrices A and B in a very particular form. In some cases, such detailed information about both matrices may not be available. In particular, in some cases, we will be interested in approximating the product of two different matrices, A and B, when only information about A (or, equivalently, only B) is available. Somewhat surprisingly, in this case, we can still obtain partial bounds (i.e., similar form, but with slightly weaker concentration) of the form we saw above. Here, we present results for the BasicMatrixMultiplication algorithm for two other sets of probabilities.

In the first case, to estimate the product AB one could use the probabilities (5) which use information from the matrix A only. In this case $\|AB - CR\|_F$ can still be shown to be small in expectation; the proof of this lemma is similar to that of our theorem for the nearly-optimal probabilities from a few classes ago, except that the indicated probabilities are used.

Lemma 3 *Suppose $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $c \in \mathbb{Z}^+$ such that $1 \leq c \leq n$, and $\{p_i\}_{i=1}^{n}$ are such that $\sum_{i=1}^{n} p_i = 1$ and such that*

$$p_k \geq \frac{\beta \left\|A^{(k)}\right\|_2^2}{\|A\|_F^2} \tag{5}$$

for some positive constant $\beta \leq 1$. Construct C and R with the BasicMatrixMultiplication *algorithm, and let CR be an approximation to AB. Then:*

$$\mathbf{E}\left[\|AB - CR\|_F^2\right] \leq \frac{1}{\beta c} \|A\|_F^2 \, \|B\|_F^2 . \tag{6}$$

Following the analysis of our theorem for the nearly-optimal probabilities from a few classes ago, we can let $\mathcal{M} = \max_\alpha \frac{\left\|B_{(\alpha)}\right\|_2}{\left\|A^{(\alpha)}\right\|_2}$, let $\delta \in (0,1)$ and let $\eta = 1 + \frac{\|A\|_F}{\|B\|_F}\mathcal{M}\sqrt{(8/\beta)\log(1/\delta)}$, in which case it can be shown that, with probability at least $1-\delta$:

$$\|AB - CR\|_F^2 \leq \frac{\eta^2}{\beta c} \|A\|_F^2 \, \|B\|_F^2 \, .$$

Unfortunately, the assumption on $\mathcal{M}$, which depends on the maximum ratio of two vector norms, is sufficiently awkward that this result is not useful. Nevertheless, we can still remove the expectation from Eqn. (6) with Markov's inequality, paying the factor of $1/\delta$, but without any awkward assumptions, assuming that we are willing to live with a result that holds with constant probability. This will be fine for several applications we will encounter, and when we use Lemma 3, this is how we will use it.

We should emphasize that for most probabilities, e.g., even simple probabilities that are proportional to (say) the Euclidean norm of the columns of A (as opposed to the norm-squared of the columns of A or the product of the norms of the columns of A and the corresponding rows of B), we obtain much uglier and unusable expressions, e.g., we get awkward factors such as $\mathcal{M}$ above. Lest the reader thing that any sampling probabilities will yield interesting results, even for the expectation, here are the analogous results if sampling is performed u.a.r. Note that the scaling factor of $\sqrt{\frac{n}{c}}$ is *much* worse than anything we have seen so far—and it means that we would have to choose c to be *larger* than n to obtain nontrivial results, clearly defeating the point of random sampling in the first place.

Lemma 4 *Suppose $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$, $c \in \mathbb{Z}^+$ such that $1 \leq c \leq n$, and $\{p_i\}_{i=1}^n$ are such that*

$$p_k = \frac{1}{n}. \tag{7}$$

Construct C and R with the BasicMatrixMultiplication *algorithm, and let CR be an approximation to AB. Then:*

$$\mathbf{E}\left[\, \|AB - CR\|_F \right] \leq \sqrt{\frac{n}{c}} \left(\sum_{k=1}^n \left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2 \right)^{1/2} . \tag{8}$$

Furthermore, let $\delta \in (0,1)$ and $\gamma = \frac{n}{\sqrt{c}}\sqrt{8\log(1/\delta)}\max_\alpha \left\|A^{(\alpha)}\right\|_2 \left\|B_{(\alpha)}\right\|_2$; then with probability at least $1-\delta$:

$$\|AB - CR\|_F \leq \sqrt{\frac{n}{c}} \left(\sum_{k=1}^n \left\|A^{(k)}\right\|_2^2 \left\|B_{(k)}\right\|_2^2 \right)^{1/2} + \gamma. \tag{9}$$

7.3 Random sampling and random projection for LS approximation

So, to take advantage of the above two structural results and bound them with our matrix multplication bounds, we need to perform the random sampling with respect to the so-called *statistical leverage socres*, which are defined as $\left\|U_{(i)}\right\|_2$, where $U_{(i)}$ is the i^{th} row of any orthogonal matrix for span(A). If we normalize them, then we get the *leverage score probabilities*:

$$p_i = \frac{1}{d} \left\|U_{(i)}\right\|_2^2 . \tag{10}$$

These will be important for our subsequent discussion, and so there are several things we should note about them.

- Since U is an $n \times d$ orthogonal matrix, the normalization is just the lower dimension d, i.e., $d = \|U\|_F^2$.
- Although we have defined these scores i.t.o. a particular basis U, they don't depend on that particular basis, but instead they depend on A, or actually on span(A). To see this, let $P_A = AA^+$ be a projection onto the span of A, and note that $P_A = QRR^{-1}Q = QQ^T$, where R is any square non-singular orthogonal transformation between orthogonal matrices for span(A). So, in particular, up to the scaling factor of $\frac{1}{d}$, the leverage scores equal the diagonal elements of the projection matrix P_A:
$$\begin{aligned} (P_A)_{ii} &= \left(U_A U_A^T\right)_{ii} = \left\|U_{A(i)}\right\|_2^2 \\ &= \left(Q_A Q_A^T\right)_{ii} = \left\|Q_{A(i)}\right\|_2^2 \end{aligned}$$
Thus, they are equal to the diagonal elements of the so-called hat matrix.
- These are scores that quantify *where* in the high-dimensional space $\mathbb{R}^n$ the (singular value) information in A is being sent (independent of what that information is).
- They capture a notion of leverage or influence that the i^{th} constraint has on the LS fit.
- They can be very uniform or very nonuniform. E.g., if $U_A = \begin{bmatrix} I \\ 0 \end{bmatrix}$, then they are clearly very nonuniform, but if U_A consists of a small number of columns from a truncated Hadamard matrix or a dense Gaussian matrix, then they are uniform or nearly uniform.

With that in place, here we will present two algorithms that compute relative-error approximations to the LS problem.

First, we start with a random sampling algorithm, given as Algorithm 1.

Algorithm 1 A "slow" random sampling algorithm for the LS problem.

```
Input: An $n \times d$ matrix $A$, with $n \gg d$, an $n$-vector $b$
Output: A $d$-vector $\tilde{x}_{opt}$
1: Compute $p_i = \frac{1}{d}\left\|U_{(i)}\right\|_2^2$, for all $i \in [n]$, from the QR or the SVD.
2: Randomly sample $r \gtrsim O(\frac{d \log d}{\epsilon})$ rows of $A$ and elements of $b$, rescaling each by $\frac{1}{rp_{i_t}}$, i.e., form
   $SA$ and $Sb$
3: Solve $\min_{x \in \mathbb{R}^d} \|SAx - Sb\|_2$ with a black box to get $\tilde{x}_{opt}$
4: Return $\tilde{x}_{opt}$
```

For this algorithm, one can prove the following theorem. The idea of the proof is to combine the structural lemma with matrix multiplication bounds that show that under appropriate assumptions on the size of the sample, etc., that the two structural conditions are satisfied.

Theorem 1 *Algorithm 1 returns a $(1\pm\epsilon)$-approximation to the LS objective and an ϵ-approximation to the solution vector.*

Next, we start with a random projection algorithm, given as Algorithm 2.

For this algorithm, one can prove the following theorem. As before, the idea of the proof is to combine the structural lemma with the random projection version of matrix multiplication bounds

Algorithm 2 A "slow" random projection algorithm for the LS problem.

```
Input: An $n \times d$ matrix $A$, with $n \gg d$, an $n$-vector $b$
Output: A $d$-vector $\tilde{x}_{opt}$
1: Let $S$ be a random projection matrix consisting of scaled i.i.d. Gaussians, $\{\pm 1\}$, etc., random variables.
2: Randomly project onto $r \gtrsim O(\frac{d \log d}{\epsilon})$ rows, i.e., linear combination of rows of $A$ and elements of $b$
3: Solve $\min_{x \in \mathbb{R}^d} \|SAx - Sb\|_2$ with a black box to get $\tilde{x}_{opt}$
4: Return $\tilde{x}_{opt}$
```

that are in the first homework to show that under appropriate assumptions on the size of the sample, etc., that the two structural conditions are satisfied.

Theorem 2 *Algorithm 2 returns a* $(1 \pm \epsilon)$*-approximation to the LS objective and an* ϵ*-approximation to the solution vector.*

We are not going to go into the details of the proofs of these two theorems, basically since they will parallel proofs of "fast" versions of these two results that we will discuss in the next few classes. But, it worth pointing out that you do get good quality-of-approximation bounds for the LS problem with these algorithms. The problem is the running time. Both of these algorithms take at least as long to run (at least in terms of worst-case FLOPS in the RAM model) as the time to solve the problem exactly with traditional deterministic algorithms, i.e., $\Theta(nd^2)$ time. For Algorithm 1, the bottleneck in running time is the time to compute the leverage score importance sampling distribution exactly. For Algorithm 2, the bottleneck in running time is the time to implement the random projection, i.e., to do the matrix-matrixcmultiplication associated with the random projection, and since we are projecting onto roughly $d \log d$ dimensions the running time is actually $\Omega(nd^2)$. Thus, they are "slow" since they are slower than a traditional algorithms—at least in terms of FLOPs in an idealized RAM model, but note that they may, and in some cases are, faster on real machines, basically for communication reasons, and similarly they might be faster in parallel-distributed environments. In particular, the random projection is just matrix-matrix multiplication, and this can be faster than doing things like QR or the SVD, even if the FLOP count is the same. But, we will focus on FLOPS and so we want algorithms to runs in $o(nd^2)$ time. We will use structured or Hadamard-based random projections, which can be implemented with Fast Fourier methods, so that the overall running time will be $o(nd^2)$. There will be two ways to do this: first, call a black box (the running time bottleneck of which is a Hadamard-based random projection) to approximate the leverage scores, and use them as the importance sampling distribution; and second, do a Hadamard-based random projection to uniformize the leverage scores and sample uniformly.

In the next few classes, we will get into these issues. Why random projections satisfy matrix multiplication bounds might be a bit of a mystery, partly since we have focused less on it, so we get into the details of two related forms of the random projection. Also, the black box to approximate the leverage scores might be surprising, since it isn't obvious that they can be computed quickly, so we will get into that. All of the results we will describe will also hold for general random sampling with exact leverage scores and general random projections, but we will get into the details for the fast versions, so we can make running time claims for analogues of the fast sampling and projection versions of above two algorithms.