Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 12 - 10/14/2013

Lecture 12: Randomized Least-squares Approximation in Practice, 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.

12 Randomized Least-squares Approximation in Practice, Cont.

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

Today, we will focus on three things.

- We will describe condition numbers and how RandNLA algorithms can lead to good preconditioning.
- We will describe two different ways that randomness can enter into the parameterization of RandNLA problems.
- We will describe the Blendenpik RandNLA LS solver.

12.1 Condition numbers and preconditioning in RandNLA algorithms

Recall that we are interested in the conditioning quality of randomized sketches constructed by RandNLA sampling and projection algorithms.

For simplicity of comparison with the Blendenpik paper, I'll state the results as they are stated in that paper, i.e., with a Hadamard-based projection, and then I'll point out the generalization (e.g., to leverage score sampling, to other types of projections, etc.) to other related RandNLA sketching methods.

Recall that after pre-multiplying by the randomized Hadamard transform H, the leverage scores of HA are roughly uniform, and so one can sample uniformly. Here is a definition we have mentioned before.

Definition 1 *Let $A \in \mathbb{R}^{n\times d}$ be a full rank matrix, with $n > d$ and let $U \in \mathbb{R}^{n\times d}$ be an orthogonal matrix for span(A). Then, if $U_{(i)}$ is the i^{th} row of U, the* coherence *of A is*

$$\mu(A) = \max_{i\in[n]} \left\|U_{(i)}\right\|_2^2$$

That is, the coherence is—up to a scaling that isn't standardized in the literature—equal to the largest leverage score. Equivalently, up to the same scaling, it equals the largest diagonal element of $P_A = A\left(A^T A\right)^\dagger A^T$, the projection matrix onto the column span of A. Defined this way, i.e., not normalized to be a probability distribution, possible values for the coherence are

$$\frac{d}{n} \leq \mu(A) \leq 1.$$

Thus, with this normalization:

- If $\mu(A) \gtrsim \frac{d}{n}$, then the coherence is small, and all the leverage scores are roughly uniform.
- If $\mu(A) \lesssim 1$, then the coherence is large, and the leverage score distribution is very nonuniform (in that there is at least one very large leverage score).

The following result (which is parameterized for a uniform sampling process) describes the relationship between the coherence $\mu(A)$, the sample size r, and the condition number of the preconditioned system.

Lemma 1 *Let $A \in \mathbb{R}^{n\times d}$ be a full rank matrix, and let $S \in \mathbb{R}^{r\times n}$ be a uniform sampling operator. Let*

$$\tau = C\sqrt{m\mu(A)\log(r)/r},$$

where C is a constant in the proof. Assume that $\delta^{-1}\tau < 1$, where δ is a failure probability. Then, with probability $\geq 1-\delta$, we have that

$$\textit{rank}(SA) = d,$$

and is the QR decomposition of SA is $SA = \tilde{Q}\tilde{R}$, then

$$\kappa\left(A\tilde{R}^{-1}\right) = \frac{1+\delta^{-1}\tau}{1-\delta^{-1}\tau}.$$

Before proceeding with the proof, here are several things to note about this result.

- We can obtain a similar result on the condition number if we sample non-uniformly based on the leverage scores, and in this case the coherence $\mu(A)$ (which could be very large, rendering the results as stated trivial) does *not* enter into the expression. This is of interest more generally, but we'll state the result for uniform sampling for now. The reason is that the Blendenpik solver does a random projection which uniformizes (approximately) the leverage scores, i.e., it preprocesses the input matrix to have a small coherence.
- Also, $\kappa(A)$, i.e., the condition number of the original problem instance, does *not* enter into the bound on $\kappa\left(A\tilde{R}^{-1}\right)$.
- If we are willing to be very aggressive in downsampling, then the condition number of the preconditioned system might *not* be small enough. In this case, all is not lost—a high condition number might lead to a large number of iterations of LSQR, but we might have a distribution of eigenvalues that is not too bad and leads to a number of iterations that is not too bad. In particular, the convergence of LSQR depends on the full distribution of the singular value of $\kappa\left(AR^{-1}\right)$ and not just the ratio of the largest to smallest (considering just that ratio leads to sufficient but not necessary conditions), and if only a few singular values are bad then this can be dealt with. We will return to this topic below.

- As we will see, the proof of this lemma directly uses ideas, e.g., subspace preserving embeddings, that were introduced in the context of low-precision RandNLA solvers, but it uses them toward a somewhat different aim.

Proof:[of Lemma 1] To prove the lemma, we need the following specialization of a result we stated toward the beginning of the semester. Again, since the lemma is formulated in terms of a uniform sampling process, we state the following lemma as having the coherence factor $(\mu(U))$, which is necessary when uniform sampling is used. We saw this approximate matrix multiplication result before when the uniform sampling operator was replaced with a random projection operator or a non-uniform sampling operator, and in both cases the coherence factor did not appear.

Lemma 2 *Let $U \in \mathbb{R}^{n\times d}$ be an orthogonal matrix, and let $S \in \mathbb{R}^{n\times n}$ be a uniform sampling-and-resacling operator. Then*

$$\mathbf{E}\left[\left\|I - U^T S^T S U\right\|_2\right] \le C\sqrt{\frac{m\mu(U)\log(r)}{r}}.$$

Since SU is full rank, so too is SA full rank. Then, we can claim that

$$\kappa(SU) = \kappa\left(A\tilde{R}^{-1}\right).$$

To prove the claim, recall that

$$\begin{aligned} SU &= U_{SU}\Sigma_{SU}V_{SU}^T \quad \text{by definition} \\ SA &= \tilde{Q}\tilde{R} \quad \text{by definition} \\ U_{SU} &= \tilde{Q}W, \quad \text{for a } d\times d \text{ unitary matrix } W\text{, since they span the same space.} \end{aligned}$$

In this case, it follows that

$$\begin{aligned} \tilde{R} &= \tilde{Q}^T S A \\ &= \tilde{Q}^T S U \Sigma V^T \\ &= \tilde{Q}^T U_{SU}\Sigma_{SU}V_{SU}^T \Sigma V^T \\ &= W\Sigma_{SU}V_{SU}^T\Sigma V^T. \end{aligned}$$

From this (and since Σ_{SU} is invertible, by the approximate matrix multiplication bound, since we have sampled sufficiently many columns), it follows that

$$\begin{aligned} A\tilde{R}^{-1} &= U\Sigma V^T V\Sigma^{-1}V_{SU}\Sigma_{SU}^{-1}W \\ &= UV_{SU}\Sigma_{SU}^{-1}W. \end{aligned}$$

From this, it follows that

$$\begin{aligned} \left\|\left(A\tilde{R}^{-1}\right)^T A\tilde{R}^{-1}\right\|_2 &= \left\|W\Sigma_{SU}^{-1}V_{SU}^T U^T U V_{SU}\Sigma_{SU}^{-1}W^T\right\|_2 \\ &= \left\|W\Sigma_{SU}^{-2}W^T\right\|_2 \\ &= \left\|\Sigma_{SU}^{-2}\right\|_2 \\ &= \left\|\Sigma_{SU}^{-1}\right\|_2^2, \end{aligned}$$

where the penultimate equality follows since W is orthogonal and the last equality follows since Σ_{SU} is diagonal. Similarly,

$$\left\| \left(\left(A\tilde{R}^{-1} \right)^T \left(A\tilde{R}^{-1} \right) \right)^{-1} \right\|_2 = \|\Sigma_{SU}\|_2^2$$

This, using that $\kappa(\alpha) = \left(\|\alpha^T\alpha\|_2 \left\| (\alpha^T\alpha)^{-1} \right\|_2 \right)^{1/2}$ for a matrix α, if follows that

$$\begin{aligned} \kappa\left(A\tilde{R}^{-1}\right) &= \left(\left\| \left(A\tilde{R}^{-1}\right)^T A\tilde{R}^{-1} \right\|_2 \left\| \left(\left(A\tilde{R}^{-1}\right)^T A\tilde{R}^{-1} \right)^{-1} \right\|_2 \right)^{1/2} \\ &= \left(\|\Sigma_{SU}^{-1}\|_2^2 \|\Sigma_{SU}\|_2^2 \right)^{1/2} \\ &= \|\Sigma_{SU}^{-1}\|_2 \|\Sigma_{SU}\|_2 \\ &= \kappa(SU). \end{aligned}$$

This establishes the claim.

So, given that claim, back to the main proof. Recall that

$$\mathbf{E}\left[\|I - U^T S^T SU\|_2 \right] \le \tau.$$

By Markov's Inequality, it follows that

$$\mathbf{Pr}\left[\|I - U^T S^T SU\|_2 \ge \delta^{-1}\tau \right] \le \delta.$$

This, with probability $\ge 1-\delta$, we have that

$$\|I - U^T S^T SU\|_2 < \delta^{-1}\tau < 1,$$

and thus SU is full rank.

$\diamond$

Next, recall that every eigenvalue λ of $U^T S^T SU$ is the Rayleigh quotient of some vector $x \neq 0$, i.e.,

$$\begin{aligned} \lambda &= \frac{x^T U^T S^T SUx}{x^T x} \\ &= \frac{x^T x - x^T \left(U^T S^T SU - I\right) x}{x^T x} \\ &= 1 + \eta, \end{aligned}$$

where η is the Rayleigh quotient of $I - U^T S^T SU$.

Since this is a symmetric matrix, it's singular values are the absolute eigenvalues. Thus, $|\eta| < \delta^{-1}\tau$. Thus, all the eigenvalues of $U^T S^T SU$ are in the interval $\left(1-\delta^{-1}\tau, 1+\delta^{-1}\tau\right)$. Thus,

$$\kappa(SU) \le \sqrt{\frac{1+\delta^{-1}\tau}{1-\delta^{-1}\tau}}.$$

12.2 Randomness in error guarantees versus in running time

So far, we have been describing algorithms in the "deterministic running time and probabilistic error guarantees" framework. That is, we parameterize/formulate the problem such that we guarantee that we take not more than

$$O\left(f\left(n,\epsilon,\delta\right)\right)$$

time, where $f\left(n,\epsilon,\delta\right))$ is some function of the size n of the problem, the error parameter ϵ, and a parameter δ specifying the probability with which the algorithm may completely fail.

That is most common in TCS, where simpler analysis for worst-case input is of interest, but in certain areas, e.g., NLA and other areas concerned with providing implementations, it is more convenient to parameterize/formulate the problem in a "probabilistic running time and deterministic error" manner. This involves making a statement of the form

$$\mathbf{Pr}\left[\text{Number of FLOPS for } \leq \epsilon \text{ relative error } \geq f\left(n,\epsilon,\delta\right)\right] \leq \delta.$$

That is, in this case, we can show that we are guaranteed to get the correct answer, but the running time is a random quantity.

A few things to note.

- These algorithms are sometimes known in TCS as Las Vegas algorithms, to distinguish them from Monte Carlo algorithm that have a deterministic running time but a probabilistic error guarantee. Fortunately, for many RandNLA algorithms, it is relatively straightforward to convert a Monte Carlo type algorithm to a Las Vegas type algorithm.
- This parameterization/formulation is a particularly convenient when we downsample more aggressively than worst-cast theory permits, since we can still get a good preconditioner (since a low-rank perturbation of a good preconditioner is still a good preconditioner) and we can often still get good iterative properties due to the way the eigenvalues cluster. In particular, we might need to iterate more, but we won't fail completely.

12.3 Putting it all together into the Blendenpik algorithm

With all this in place, here is the basic Blendenpik algorithm.

We will go into more detail on how/why this algorithm works next time (in terms of condition number bounds, potentially loosing rank, since $r = o\left(d\log(d)\right)$, etc.), but here are a few final notes for today.

- Depending on how aggressively we downsample, the condition number $\kappa\left(AR^{-1}\right)$ might be higher than $1+\epsilon$. If it is too much larger, then LSQR converges slowly, but the algorithm does not fail completely, and it eventually converges. In this sense, we get Las Vegas guarantees.
- Since the downsampling is very aggressive, the preconditioner can actually be rank-deficient or ill-conditioned. The solution that Blendenpik uses is to estimate the condition number with a condition number estimator from LAPACK, and if it is $\geq \epsilon_{mach}^{-1}/5$ then randomly project and sample again.

Algorithm 1 The `Blendenpik` algorithm.

```
Input: $A \in \mathbb{R}^{n \times d}$, and $b \in \mathbb{R}^n$.
Output: $\tilde{x}_{opt} \in \mathbb{R}^d$
1: while Not returned do
2:     Compute $HDA$ and $HDb$, where $HD$ is one of several Randomized Hadamard Transforms.
3:     Randomly sample $\gamma d/n$ rows of $A$ and corresponding elements of $b$, where $\gamma \approx 2d$, and let $S$
       be the associated sampling-and-rescaling matrix.
4:     Compute $SHDA = QR$.
5:     $\tilde{\kappa} = \kappa_{estimate}(R)$, with LAPACK's DTRCON routine.
6:     if $\tilde{\kappa}^{-1} > 5\epsilon_{mach}$ then
7:         $\tilde{x}_{opt} = \text{LSQR}\left(A, b, R, 10^{-14}\right)$ and return
8:     else if Number of iterations $> 3$ then
9:         Call LAPACK and return
10:    end if
11: end while
```

- In Blendenpik, they use LSQR, but one could use other iterative procedures, and one gets similar results.

We will go into more detail on these topics next time.