Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 18 - 11/04/2013

Lecture 18: Randomized Low-rank 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.

18 Randomized Low-rank Approximation in Practice, Cont.

Today, we will continue with the discussion from last time. Again, from last time, here is the reading for today.

- Boutsidis, Mahoney, and Drineas "An Improved Approximation Algorithm for the Column Subset Selection Problem"
- Halko, Martinsson, and Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions"

Recall that last time we described a more refined structural result for low-rank matrix approximation. Today and next class we will describe how to use it for two seemingly very different types of low-rank matrix approximation.

In particular, today we will cover the following topics.

- The Column Subset Selection Problem.

18.1 Column Subset Selection Problem

Last time, we talked about a basic structural result for low-rank matrix approximation via RandNLA algorithms, and we said how it was crucial for obtaining improved results in theory and in practice when one wants to sample $o(k\log(k))$, e.g., exactly k or $k+p$ for a small positive integer p, number of columns. The first example of this—historically as well as in terms of what we will do this semester—has to do with the so-called Column Subset Selection Problem (CSSP).

The CSSP is the problem of choosing the best set of *exactly* k columns from an input matrix A. Importantly, this problem is intractable (in the sense of TCS complexity theory) for nearly every interesting formalization of best, e.g., to get columns that capture the most mass, to get columns that are maximally uncorrelated, to get columns that span a parallelepiped of maximum volume, etc. The reason is that asking for actual columns is in many ways very different than asking for the set of linear combinations of columns that are best by any of these notions. The leading eigenvectors or singular vectors are best in terms of maximizing variance, being maximally uncorrelated, etc, and this is fundamental to linear algebra and Euclidean spaces, but asking for the best set of actual columns is a much more combinatorial problem. Here, we will consider the following version of the CSSP.

Definition 1 *Given $A \in \mathbb{R}^{m \times n}$ and $k \in \mathbb{Z}^{+}$, choose the k columns of A to form the matrix $C \in \mathbb{R}^{m \times k}$ such that the residual $\|A - P_C A\|_{\xi}$, where (say) $\xi \in \{2, F, *\}$, is minimized over all $\binom{n}{k}$ choices for C.*

This is a nice version of the problem for us for the following reasons. Basically, it is asking to capture mass on the top part of the spectrum, so it is doing what TCS-based versions of the low-rank approximation problem are doing, but it is also asking for exactly k columns, which is doing something very different and more like the usual NLA approach. Relatedly, we can use it to highlight several of the differences between NLA and TCS perspectives. Let's do that now.

In NLA:

- The algorithms have historically been almost exclusively deterministic, typically greedy (depending on things like pivot rule decisions, etc.) algorithms.
- There are strong connections between the CSSP and QR, RRQR, etc. algorithms, and one typically chooses exactly k columns with these procedures.
- There is a strong emphasis on conditioning, backward error analysis, and constant factors in the running time.
- Most of the effort focuses on getting good spectral norm bounds.

In TCS:

- There has been a long tradition of randomization used inside the algorithm, in general, with RandNLA projection algorithms, and also for problems like CX/CUR that have some similarities with the CSSP.
- One typically keeps more than k column, e.g., $O\left(k \log(k)/\epsilon^2\right)$ columns, where the big-O hides sometimes unknown factors.
- Most of the effort focuses on getting good Frobenius norm bounds.

Importantly, it isn't obvious how to combine these two very different approaches. For example, when running typical NLA algorithms, if one looks at the details of the pivot rule decisions, etc., then it isn't clear from the analysis that the bounds will improve if one keeps a few extra columns. Alternatively, when running typical TCS algorithms, one often can't get worst case bounds by keeping fewer columns, typically for rather basic reasons such as reduction to the coupon collector problem.

Here, we will describe an algorithm for the CSSP that combines these two perspectives in a non-trivial manner. It is a two-stage algorithm, where in the first stage one chooses a random sample with a TCS-style approach, and where in the second stage one cuts back to exactly k columns by running a QR on the chosen columns. (Actually, there is an important subtlety, where the QR is done on the sampled version of the matrix of right singular subspace, and where the corresponding columns from the original matrix are kept.)

Here is the algorithm. The following algorithm takes as input a matrix $A \in \mathbb{R}^{m \times n}$ and a number $k \in \mathbb{Z}^+$, and it returns as output a matrix $C \in \mathbb{R}^{m \times k}$ consisting of exactly k columns of A.

1. Initial stage
    (a) Compute (exactly or approximately) the top k right singular vectors of A, call them V_k.
    (b) From that orthogonal basis, compute (exactly or approximately) the importance sampling probabilities $\{p_i\}_{i=1}^n$, where
    $$p_i = \frac{1}{k} \left\| (V_k)_{(i)} \right\|_2^2 .$$
    (c) Let $c = \Theta\left(k \log(k)\right)$ (where there is no dependence on ϵ, e.g., set $\epsilon = 1/2$ in the previous sampling algorithm).
2. Randomized Stage
    (a) For $t \in [c]$, choose an integer from $[n]$ with probability p_i, and if i is chosen then keep the scaling factor $1/\sqrt{cp_i}$. Form the sampling matrix S_1 and the diagonal rescaling matrix D_1.
3. Deterministic Stage
    (a) Run a deterministic QR algorithm, e.g., an algorithm of Pan or the Gu-Eisenstat algorithm, on $V_k^T S_1 D_1$ to get exactly k columns from $V_k^T S_1 D_1$, thereby forming the sampling matrix S_2.
    (b) Return the corresponding k columns of A, i.e., return $C = AS_1S_2$.

Here are several comments regarding this algorithm.

- For simplicity, we described this algorithm in terms of the exact right singular vectors. The proof of the main quality-of-approximation theorem for this algorithm uses the main structural result for low-rank approximation, and that result is robust to using an approximate basis for the right/left singular subspace. Thus, the running time of this algorithm could be improved by approximating that subspace, say, with a random projection algorithm.
- The running time bottleneck for this algorithm is the time to compute V_k^T, either exactly or approximately.
- The sampling probabilities given in the algorithm work for the Frobenius and Trace norms. For the spectral norm, one should use exact or approximate probabilities of the form

$$\begin{aligned} p_i &= \frac{\frac{1}{2}\left\|(V_k)_{(i)}\right\|_2^2}{\sum_{j=1}^n \left\|(V_k)_{(j)}\right\|_2^2} + \frac{\frac{1}{2}\left\|(A-A_k)^{(i)}\right\|_2^2}{\sum_{j=1}^n \left\|(A-A_k)^{(j)}\right\|_2^2} \\ &= \frac{\left\|(V_k)_{(i)}\right\|_2^2}{2k} + \frac{\|A^{(i)}\|_2^2 - \left(AV_kV_k^T\right)^{(i)}}{2\left(\|A\|_F^2 - \|AV_kV_k^T\|_F^2\right)}. \end{aligned}$$

While this looks complicated, these probabilities can computed from only the top part of the spectrum of A, i.e., a knowledge of V_k (exactly or approximately) suffices to compute them. It is an open question whether these more complicated probabilities are necessary or simply a weakness of the analysis.

- Note that since the algorithm returns exactly k columns, there is no need to worry about rescaling and thus the algorithm doesn't do that.

Here is the main theorem that we can prove about this algorithm.

Theorem 1 *Let $A \in \mathbb{R}^{m \times n}$, and let $k \in \mathbb{Z}^+$. If we run the above CSSP algorithm, then with constant probability, the following hold.*

$$\begin{aligned}
\|A - P_C A\|_2 &\leq \Theta\left(k \log^{1/2}(k)\right) \|A - A_k\|_2 + \Theta\left(k^{3/4} \log^{1/4(k)}\right) \|A - A_k\|_F \\
\|A - P_C A\|_F &\leq \Theta\left(k \log^{1/2}(k)\right) \|A - A_k\|_F \\
\|A - P_C A\|_* &\leq \|A - A_k\|_* + \Theta\left(k^{3/2} \log^{1/2}(k)\right) \|A - A_k\|_F
\end{aligned}$$

Proof: We will prove the theorem by establishing a sequence of lemmas.

Lemma 1 *Given S_1 and D_1 from the algorithm, with constant (say, at least* 0.9*) probability,*

$$\sigma_k\left(V_k^T S_1 D_1\right) \geq \frac{1}{2}.$$

In particular, $V_k^T S_1 D_1$ has full rank.

Proof:[of lemma] To bound $\sigma_k\left(V_k^T S_1 D_1\right)$, we bound

$$\left\|V_k^T S_1 D_1 D_1 S_t^T V_k^T - I_k\right\|_2,$$

showing that it is $\leq 1/2$. Note that our sampling probabilities that are used by the algorithm are approximately optimal for this, i.e., $p_i \geq \frac{1}{2k}\left\|\left(V_k^T\right)_{(i)}\right\|_2^2$, and so we have that $\beta = 1/2$. We can set $\epsilon = 1/2$, and the lemma follows by the approximate matrix multiplication theorem.

$\diamond$

(Note that the proof of that lemma is essentially the same argument that we used for over-determined regression problems, except that there we were sampling rows, using the leverage scores defined by the column space, while here we are sampling columns, so we use leverage scores defined by the best rank k approximation to the row space.)

Lemma 2

$$\|A - P_C A\|_\xi \leq \|A - A_k\|_\xi + \sigma_k^{-1}\left(V_k^T S_1 D_1\right) \|(A - A_k) S_1 D_1\|_\xi$$

Proof:[of lemma] First, observe that since we are projecting onto exactly k columns, rescaling doesn't matter, and so we can rescale or not as convenient. Next we have that

$$\begin{aligned} A - P_C A &= A - AS_1S_2\left(AS_1S_2\right)^\dagger A \\ &= A - AS_1D_1S_2\left(AS_1D_1S_2\right)^\dagger A \\ &= A - AS\left(AS\right)^\dagger A, \end{aligned}$$

where $S = S_1D_1S_2 \in \mathbb{R}^{n\times k}$ is a sketching matrix representing the two steps of the algorithm. By the main structural lemma for low-rank approximation, we have that

$$\begin{aligned} \left\|A - P_C A\right\|_\xi &\leq \left\|A - A_k\right\|_\xi + \left\|\Sigma_{k,\perp}V_{k,\perp}S\left(V_k^TS\right)^\dagger\right\|_\xi \\ &= \left\|A - A_k\right\|_\xi + \left\|U_{k,\perp}\Sigma_{k,\perp}V_{k,\perp}S\left(V_k^TS\right)^\dagger\right\|_\xi \\ &= \left\|A - A_k\right\|_\xi + \left\|\left(A - A_k\right)S\left(V_k^TS\right)^\dagger\right\|_\xi \\ &\leq \left\|A - A_k\right\|_\xi + \left\|\left(A - A_k\right)S\right\|_\xi \left\|\left(V_k^TS\right)^\dagger\right\|_2 \quad \text{(by strong submultiplicitivity)} \\ &\leq \left\|A - A_k\right\|_\xi + \sigma_k^{-1}\left(V_k^TS_1D_1\right)\left\|\left(A - A_k\right)S_1D_1\right\|_\xi . \end{aligned}$$

The last line follows since $S = S_1D_1S_2$, since S_2 is orthogonal, and since $V_k^TS_1D_1$ has full rank.

◇

Lemma 3 *With constant probability (say, ≥ 0.9), we have that $\sigma_k^{-1}\left(V_kS_1D_1S_2\right) \leq 2\sqrt{2k\left(c-k\right)+1}$.*

Proof:[of lemma] This follows since we call the analysis of the Gu-Eisenstat QR routine.

◇

(We would get a different result here if we used a different QR algorithm, other than that of Gu-Eisenstat, at that step of the algorithm.)

Lemma 4 *With constant probability (say, ≥ 0.9), we have that*

$$\begin{aligned} \left\|\left(A - A_k\right)S_1D_1\right\|_2 &\leq \left\|A - A_k\right\|_2 + \frac{12}{c^{1/4}}\left\|A - A_k\right\|_F \\ \left\|\left(A - A_k\right)S_1D_1\right\|_F &\leq \sqrt{10}\left\|A - A_k\right\|_F \\ \left\|\left(A - A_k\right)S_1D_1\right\|_* &\leq \sqrt{10c}\left\|A - A_k\right\|_F . \end{aligned}$$

Proof:[of lemma] This is straightforward, so we will omit.

◇

The theorem follows by combining the results from these lemmas.

◇