Stat260/CS294: Randomized Algorithms for Matrices and Data

Lecture 16 - 10/28/2013

Lecture 16: Relative-error Low-rank Matrix Approximation with Sampling and Projections

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.

16 Relative-error Low-rank Matrix Approximation

Today, we will start to discuss how to improve the rather coarse additive-error low-rank matrix approximation algorithms from the last two classes to obtain much better results for low-rank matrix approximation. Importantly, "better" means very different things to different research communities, and thus we will discuss several different notions of better. We will start by describing how to improve the additive-error bounds we have been discussing to relative-error low-rank matrix approximation. Here is the reading for today.

- Drineas, Mahoney, and Muthukrishnan, "Relative-Error CUR Matrix Decompositions"

Today, in particular, we will cover the following topics.

- Various types of column/row-based low-rank approximations.
- Different classes of approaches to choosing good columns/rows.
- A generalization of the LS regression algorithm to fat matrices that are low rank.
- Basic results for CX/CUR low-rank decompositions that achieve relative-error reconstruction guarantees on the top part of the spectrum with respect to the Frobenius norm error.

16.1 Low-rank Approximations via Columns/Rows

We are now going to consider much better (in the sense of TCS) low-rank matrix approximations—these approximations will come with so-called relative-error, or $1 \pm \epsilon$, approximation guarantees. This will be an improvement over the additive-error algorithm presented in the last class since the scale of the additional error for these improved algorithms will be the base residual, i.e., the norm of the part of the matrix that is not captured by the best rank k approximation. As with our discussion of ℓ_2 regression and additive-error low-rank matrix approximation algorithms, our discussion here will start with algorithms that randomly sample actual columns and/or rows, and then it will consider the extension of these ideas to algorithms that perform random projections.

CX matrix decompositions. We will start with the following definition.

Definition 1 *Given a matrix $A \in \mathbb{R}^{m \times n}$, for a matrix $C \in \mathbb{R}^{m \times c}$, consisting of c actual columns of A, the matrix $A' = CX$ is a* column-based matrix approximation *or a* CX matrix decomposition*, for any matrix $X \in \mathbb{R}^{c \times n}$.*

Things to note about this definition.

- First, although we have defined it in general, we are most interested in the case that $c \ll n$.
- Second, one may think of this as a decomposition that decomposes A into a small number of "dictionary elements" or "basis columns," each of which is an actual column of A. Although this is of less interest from an abstract perspective, it can be very important in certain applications.
- Third, if we choose $X = C^+A$, where C^+ is the Moore-Penrose generalized inverse of C, then this is the "best" CX approximation, for that C, with respect to reconstruction error for any unitarily invariant matrix norm.
- Finally, many algorithms, e.g., traditional QR decompositions, can be thought of a providing CX decompositions—just keep the columns that are used in the Gram-Schmidt procedure of the algorithm. The question of interest will be what quality-of-approximation results can be proven for the procedure.

CUR matrix decompositions. We will next consider the following definition.

Definition 2 *Given a matrix $A \in \mathbb{R}^{m \times n}$, for a matrix $C \in \mathbb{R}^{m \times c}$, consisting of c actual columns of A and a matrix $R \in \mathbb{R}^{r \times n}$, consisting of r actual rows of A, the matrix $A' = CUR$ is a* column-row-based matrix approximation *or a* CUR matrix decomposition*, for any matrix $U \in \mathbb{R}^{c \times r}$.*

Things to note about this definition.

- First, it is sometimes called a pseudoskeletal decomposition, and it has also been called a generalized Nyström approximation.
- Second, it is also a CX decomposition, on the original matrix as well as on its transpose, where the matrix X has a particular structure of $X = UR$.
- Third, in terms of it's singular value structure, since both C and R contain singular value information from the original matrix A, the matrix U should contain "inverse of A" information, in order to provide a good low-rank approximation. For example, it could consist of the generalized inverse of the intersection of C and R, or a related quantity that is quicker to compute.
- Finally, while we have defined CX and CUR in terms of actual columns, the analysis will try to quantify how well C and R represent the column space and row space of A. If C and R were permitted to be matrices consisting of linear combinations of columns, then one choice (and the best choice with respect to unitarily invariant matrix norms) would be to set them to be the left and right singular vectors, in which case the middle matrix would be the matrix of singular values.

Nyström approximations. We will next consider the following definition, which generalizes the notion of CUR decompositions to SPSD matrices. (SPSD matrices are of interest in many applications. In machine learning, in particular, they are known as kernels.)

Definition 3 *Given a matrix $A \in \mathbb{R}^{n \times n}$ that is SPSD, for a matrix $C \in \mathbb{R}^{m \times c}$, consisting of c actual columns of A, let $U = W^+$, where the $c \times c$ matrix W is the intersection of C with itself, then the matrix $A' = CUC^T = CW^+C^T$ is a low-rank* Nyström approximation *to A.*

Things to note about this definition.

- First, this is just a CUR approximation for a SPSD matrix, where one chooses the same columns and rows and a symmetric middle matrix, in order to preserve the SPSD property.
- Second, the name Nyström comes because decompositions of this form have been used historically in scientific computing, but there has been interest in decompositions of this form in recent years in ML.
- Third, it is not immediate, given the usual constructions of CUR, that one can choose some columns and rows and preserve the SPSD property; alternatively, if you choose the same columns and rows, then you preserve the SPSD property, but the analysis that leads to good quality-of-approximation bounds typically fails.

The reason for these points is that the analysis boils down to a LS approximation, and thus one gets an asymmetry in the choice of C and then R. Relatedly, it is difficult to certify that a matrix is SPSD, unless one has the square root, the computation of which we want to avoid, or unless the matrix has some strong property like being diagonally dominant. One can do this with uniform sampling and preserve the SPSD property, and this empirically works well in certain cases, but it can perform very poorly in general. Alternatively, one can get around some of these problems by working within the SPSD sketching model and appealing to the "implicit square roor" result due to Gittens.

More general sketching models. Although the three previous definitions were in terms of actual columns, if we view those columns as $C = AS$, where A is the original matrix and S is a sampling matrix, it is fair to ask how sensitive the results are to the particular form of S. For example, could it be a projection matrix, or could it be some other deterministic matrix? In general, the answer is Yes. We discussed this last week, when we described how the analysis for the additive-error low-rank sampling algorithms extends more-or-less immediately to additive-error bounds for random projection algorithms, and it holds more generally. (You also got a chance to work through it in detail on the homework.) We will describe this more generally below, but for now we just state this result for SPSD matrices.

Definition 4 *Let $A \in \mathbb{R}^{n \times n}$ be SPSD. Let $S \in \mathbb{R}^{n \times \ell}$, with $\ell \ll n$, be a sketching matrix (e.g., a sampling matrix or a projection matrix or anything else). Let $C = AS$, and let $W = S^TAS$. Then, a matrix A' of the form $A' = CW^+C^T$ is a low-rank approximation with rank$(A') \leq k$ that we call the* SPSD sketching model.

Things to note about this definition.

- First, as stated, this is not guaranteed to be numerically stable, so we can truncate the small singular values and/or regularize/smooth them out in different ways. This has been done in both ML and scientific computing, although the details of how the do it and the exact results that are established differ somewhat.
- Second, Nyström is an example of this, if S is a sampling matrix, but this can include random projections also, and other things also, if S is more general.
- Third, Gittens has shown that if you work with a low-rank approximation satisfying this, then in a certain precise sense, you are implicitly working on the square root of A; this means that you can get much stronger bounds than naive methods like uniform sampling. We will get back to this later.

Pictorial illustration. CX and CUR decompositions take a particularly nice form, which is illustrated as follows.

$$\underbrace{\begin{pmatrix} \\ \quad A \quad \\ \\ \end{pmatrix}}_{m\times n} \approx \underbrace{\begin{pmatrix} \\ C \\ \\ \end{pmatrix}}_{m\times c} \underbrace{\begin{pmatrix} \quad X \quad \end{pmatrix}}_{c\times n} \tag{1}$$

$$\approx \underbrace{\begin{pmatrix} \\ C \\ \\ \end{pmatrix}}_{m\times c} \underbrace{\begin{pmatrix} U \end{pmatrix}}_{c\times r} \underbrace{\begin{pmatrix} \quad R \quad \end{pmatrix}}_{r\times n}. \tag{2}$$

For Nyström approximations, the decomposition is of course more symmetric than for general matrices.

16.2 Approaches to Choosing Good Columns and Rows

Let's consider how to construct a decomposition of the form $A \approx CX$ or $A \approx CUR$. In particular, there are several questions. First, how do you choose "good" columns C? (For now, we will focus on good in the sense that we want to reproduce well the top part of the spectrum of the original matrix, but we will return later when we discuss the Column Subset Selection Problem to other notions of a good set of columns.) Among the answers are:

- Perform a QR decomposition, and keep the associated columns. Often this actually perform fairly well empirically. Most of the bounds are for the spectral norm of the reconstruction error (although there are a bunch of variants, e.g., vanilla QR, rank-revealing QR, etc. that can guarantee other properties). But note that this method can get "stuck" in "corners," which operationally basically means that they need to make smarter pivot rule decisions in order to get better results.

- Perform a greedy iterative algorithm. This also often does reasonably well in practice, either being very greedy, like with methods like GreedyPursuit, etc., or greedy in a softer randomized way, e.g., don't keep the column at each step that is best according to some metric, but instead keep columns that are biased toward those that are best by that metric. These methods can be more difficult to analyze, leading to weaker bounds.
- Perform random sampling, e.g., by choosing a small number of columns with probabilities proportional to $\|A^{(i)}\|_2^2$. This is pass efficient, and you get additive error bounds, which as we have said is good but certainly not great.
- Perform random iterative sampling with probabilities that depend on the norm of the residual. The iterative algorithm we discussed is a version of this. It is is weakly pass efficient, in that it takes a number of rounds that is "small," in that it depends on k, and you get additive error bounds with a scale that improves exponentially with the number of rounds.
- Construct some sort of (potentially more expensive) score over the columns of A and use those an an importance sampling distribution. In particular, given the $1 \pm \epsilon$ relative-error bounds for ℓ_2 regression, perhaps that can be used to get $1 \pm \epsilon$ relative-error bounds for low-rank approximation problems.

Second, given C, how do you choose "good" columns R? Among the answers are:

- Ignore C and choose rows R using one of the methods described above by looking at A^T. Often, in this case, you can still construct U such that if you have additive-error bounds for C then you get additive-error bounds for CUR. Alternatively, often you can construct U such that if you have $1 \pm \epsilon$ bounds originally, then you get $2 \pm \epsilon$ bounds by combining them in this relatively naive way.
- Perhaps we can use information in C and take advantage of ℓ_2 regression result to get rows that are good with respect to C (which are good with respect to A) and combine them in order to get $1 \pm \epsilon$ bounds for CUR.

In fact, we will be able to do this last suggestion. The basic idea is that we will choose columns C that are relative-error good approximations to the best rank k approximation to A; and then choose rows R that are relative-error good approximations for those columns C. (BTW, this strategy that is asymmetric with respect to rows and columns explains why it is difficult—which as we have said is overcomeable—to do very good Nyström approximation of general SPSD matrices.) In particular, the choice for each of these two steps is a special case of a generalization of the over-determined ℓ_2 regression problem. Thus, let's consider that problem.

16.3 Generalized ℓ_2 Regression

The approximation symbol in Eqn. (1) and Eqn. (2) is meant to be more than suggestive. In particular, in addition to providing an approximation to A in some vague sense, the analysis for relative-error CX and CUR decompositions (which we will get to now) proceeds by showing that (1) one can find a small number of columns C that are relative-error-good with respect to the columns that define the space that is the best rank k approximation to A, and (2) one can then find a small number of rows R that are relative-error-good with respect to those columns C. Thus, in both cases, the approximation is a generalization of the relative-error random sampling algorithm for the least-squares problem.

Thus, since our bounds for CX and CUR will in fact boil down to a generalization of the over-determined ℓ_2 regression problem we discussed earlier, we will start with that. Recall that before we considered solving

$$\min_x \|Ax - b\|_2^2 ,$$

where A is a "tall" matrix and b is a vector. Here, we will consider the generalization where A is an arbitrarily-sized matrix with rank no greater than k, and we will also consider the generalization of the case where the right and side is a general matrix B rather than a vector. Here is our algorithm.

Algorithm 1 The GENERALIZED ℓ_2 REGRESSION Algorithm.

```
Input: An $m \times n$ matrix $A$, with rank no greater than $k$, an $m \times p$ matrix $B$, an integers $c$ s.t. $1 \le k \le c \le n$, and a probability distribution $\{p_i\}_{i=1}^n$.
Output: An $m \times p$ matrix $\tilde{X}_{opt}$ and a number $\tilde{\mathcal{Z}} \in \mathbb{R}$.
1: Use the probabilities $\{p_i\}_{i=1}^n$ to form a random sampling matrix $S \in \mathbb{R}^{m \times c}$ and a diagonal rescaling matrix $D$.
2: Construct $DS^TA$ and $DS^TB$.
3: Solve the subsampled problem with a black box to get
```

$$\begin{aligned} \tilde{X}_{opt} &= (DS^TA)^+ DS^TB & (3)\\ \tilde{\mathcal{Z}} &= \min_{X \in \mathbb{R}^{m \times p}} \left\| DS^TB - DS^TA\tilde{X}_{opt} \right\|_F . & (4) \end{aligned}$$

Here is the main theorem that we can prove about this algorithm. Note that is is the obvious generalization of the over-determined ℓ_2 regression result.

Theorem 1 *Suppose $A \in \mathbb{R}^{m \times n}$ has rank no greater than k, $B \in \mathbb{R}^{m \times p}$, $\epsilon \in (0, 1]$, and let $\mathcal{Z} = \min_{X \in \mathbb{R}^{n \times p}} \|B - AX\|_F = \|B - AX_{opt}\|_F$, where $X_{opt} = A^+B = A_k^+B$. Run Algorithm 1 with any sampling probabilities of the form*

$$p_i \ge \beta \frac{\left\| (U_{A,k})_{(i)} \right\|_2^2}{\sum_{j=1}^n \left\| (U_{A,k})_{(j)} \right\|_2^2} = \frac{\beta}{k} \left\| (U_{A,k})_{(i)} \right\|_2^2, \qquad \forall i \in [n], \tag{5}$$

for some $\beta \in (0, 1]$, and assume that the output of the algorithm is a number $\tilde{\mathcal{Z}}$ and an $n \times p$ matrix $\tilde{X}_{opt}$. If exactly $r = 3200k^2/\beta\epsilon^2$ rows are chosen with the EXACTLY(c) *algorithm, then with probability at least* 0.7*:*

$$\begin{aligned} \left\| B - A\tilde{X}_{opt} \right\|_F &\le (1 + \epsilon)\, \mathcal{Z}, & (6)\\ \left\| X_{opt} - \tilde{X}_{opt} \right\|_F &\le \frac{\epsilon}{\sigma_{\min}(A_k)} \mathcal{Z}. & (7) \end{aligned}$$

If, in addition, we assume that $\left\| U_{A,k}U_{A,k}^TB \right\|_F \ge \gamma \|B\|_F$, for some fixed $\gamma \in (0, 1]$, then with probability at least 0.7*:*

$$\left\| X_{opt} - \tilde{X}_{opt} \right\|_F \le \epsilon \left(\kappa(A_k) \sqrt{\gamma^{-2} - 1} \right) \|X_{opt}\|_F . \tag{8}$$

Similarly, under the same assumptions, if $r = O(k \log k/\beta\epsilon^2)$ rows are chosen in expectation with the Expected(c) *algorithm, then with probability at least* 0.7*, (6), (7), and (8) hold.*

Things to note about this result.

- Clearly, the factors of 3200 and so on are artifacts of the analysis and the particular (now out of date) bounds that were used to establish this result. We include them here in the statement of this result for ease of comparison with the DMM paper.
- Also, the proof of this theorem is a pretty immediate generalization of our previous analysis of very over-determined $\ell_2$regression, and in particular it boils down to two approximate matrix multiplication bounds that generalize the previous structural results. (The role of the low dimension in the previous result is replaced here with the exactly low-rank space that captures all of the matrix A.) Thus, we won't provide it here.
- We will apply this result to do CX and CUR on general matrices A with general rank (and by extension relative-error low-rank random projection algorithms), and there will be no assumption of being exactly low-rank on those matrices. We will prove that those CX/CUR results work by appealing to this generalization of ℓ_2 regression that works for matrices of exactly rank k. (Actually, the rank can be less, and there are some numerical issues there, but we won't go into them.) So, think of this rank restriction as being inside the analysis of sampling and projection algorithms on arbitrary matrices (for the analysis of CX/CUR/projections/etc.), and it is *not* an assumption about the input.
- As with the rectangular regression problem, if the other sketching operators satisfy those two conditions, then the same results go through. In particular, we can use random projections of appropriate sized, as well as CX, CUR, and Nyström approximation results below. One can view this as a modification of Algorithm 1 to hold for general sketching matrices or as preprocessing the input with a random projection based preconditioning.

16.4 CX and CUR Decompositions of General Matrices

Next, we will use the generalized ℓ_2 regression result to get very fine relative-error bounds on CX and CUR decompositions. (By extension, this will also give relative-error bounds random projection algorithms for low-rank matrix approximation, when the dimensions of the projection are chosen appropriately. We won't go into those here, but see the homework for details.) We'll first describe a few related algorithms and establish quality-of-approximation bounds, and then we will discuss running time considerations.

Here is a randomized algorithm for constructing CX matrix decompositions. The algorithm takes as input a matrix $A \in \mathbb{R}^{m \times n}$, a rank parameter k, and an error parameter ϵ, and it returns as output a matrix $C \in \mathbb{R}^{m \times c}$. It does the following steps.

1. Compute (exactly or approximately) the distribution $\{p_i\}_{i=1}^n$, where $p_i = \frac{1}{k} \left\| (U_{A,k})_{(i)} \right\|_2^2$.
2. Using $\{p_i\}_{i=1}^n$ as an importance sampling distribution, construct a random sampling matrix $S_C \in \mathbb{R}^{n \times c}$ and a diagonal rescaling matrix $D_C \in \mathbb{R}^{c \times c}$.

3. Construct $C = AS_C D_C \in \mathbb{R}^{c \times c}$, a matrix consisting of a small number of columns of A.

Here is what we can prove regarding this CX algorithm.

Theorem 2 *Let $A \in \mathbb{R}^{m \times n}$, and let $k \in \mathbb{Z}^+$. If we call the above algorithm with $c = O\left(k \log(k)/\epsilon^2\right)$, then with constant probability we have that*

$$\left\| A - CC^\dagger A \right\|_F \leq (1+\epsilon) \left\| A - A_k \right\|_F .$$

Proof: Since $C = AS_C D_C$, we have that $X_{opt} = C^\dagger A$ is the matrix that minimized $\|A - CX\|_F$. Then, we have the following chain of equalities and inequalities.

$$\begin{aligned}
\left\| A - CC^\dagger A \right\|_F &= \left\| A - (AS_C D_C)(AS_C D_C)^\dagger A \right\|_F \\
&\leq \left\| A - (AS_C D_C)(P_{A_k} AS_C D_C)^\dagger P_{A_k} A \right\|_F \quad (\text{where } P_{A_k} = U_{A,k} U_{A,k}^T) \\
&= \left\| A - (AS_C D_C)(A_k S_C D_C)^\dagger A_k \right\|_F \\
&\leq (1+\epsilon) \left\| A - AA_k^\dagger A_k \right\|_F \quad (\text{by the generalized LS result}) \\
&= (1+\epsilon) \left\| A - A_k \right\|_F .
\end{aligned}$$

$\diamond$

Remark. This holds with constant probability, but that probability can be boosted to $1-\delta$ using standard methods. Also, for simplicity, this is stated such that $A' = CC^\dagger A$ might have rank $> k$, but actually the following even stronger result holds. If we consider $A'' = C\left(P_{A,k} C\right)^\dagger P_{A,k} A$, then the analysis of this theorem can also be used to show that A'' is a CX approximation, such that it has rank no greater than k and also that it is also a $1 \pm \epsilon$ relative-error approximation.

Next, let's consider how to extend this CX result to CUR decompositions. We'll show a trivial way that gives a weaker $2+\epsilon$ constant-factor approximation, and then we'll show a non-trivial way that gives stronger $1+\epsilon$ relative-error approximation.

Here is the weaker randomized algorithm for construction CUR matrix decompositions. It basically amounts to calling the previous algorithm on A and A^T separately. The algorithm takes as input a matrix $A \in \mathbb{R}^m \times n$, a rank parameter k, and an error parameter ϵ, and it returns as output matrices C, U, and R. It does the following steps.

1. With $c = O\left(k \log(k)/\epsilon^2\right)$, call the previous algorithm on A to get a matrix $C \in \mathbb{R}^{m \times c}$
2. With $r = O\left(k \log(k)/\epsilon^2\right)$, call the previous algorithm on A^T to get a matrix $R \in \mathbb{R}^{r \times n}$.
3. Let $U = C^\dagger A R^\dagger$.

Note that U clearly has the singular value structure of the pseudo-inverse of A; this is true more generally, but this weaker construction makes it very obvious.

Here is what we can prove regarding this weaker CUR algorithm.

Theorem 3 *Let $A \in \mathbb{R}^{m \times n}$, and let $k \in \mathbb{Z}^{+}$. If we call the above algorithm with $c = O\left(k \log(k)/\epsilon^2\right)$ and $r = O\left(k \log(k)/\epsilon^2\right)$, then with constant probability we have that*

$$\|A - CUR\|_F \leq (2+\epsilon)\ \|A - A_k\|_F .$$

Proof:

$$\begin{aligned}
\|A - CUR\|_F &= \left\|A - CC^{\dagger}AR^{\dagger}R\right\|_F \\
&\leq \left\|A - CC^{\dagger}A\right\|_F + \left\|CC^{\dagger}A - CC^{\dagger}AR^{\dagger}R\right\|_F \quad \text{(by submultiplicitivity)} \\
&\leq \left\|A - CC^{\dagger}A\right\|_F + \left\|A - AR^{\dagger}R\right\|_F \quad \text{(since } CC^{\dagger} \text{ only decreases the norm)} \\
&= \|A - P_C A\|_F + \|A - AP_R\|_F \\
&\leq (2+\epsilon)\ \|A - A_k\|_F
\end{aligned}$$

◇

That factor of $(2+\epsilon)$ might not matter if these algorithms were applied to matrices that were really *very* well approximated by a low-rank matrix, but they are often applied to matrices that are only *moderately* low-rank, in which case that factor is much larger and can matter a lot. Also, the increase is "real" in that the choice of columns and rows is uncoupled, which in many practical applications introduces a lot of additional error. To remedy this, we are interested in coupling the choice of C and R, as this will permit us to obtain $1+\epsilon$ relative-error approximation for columns and rows together.

Here is the stronger randomized algorithm for construction CUR matrix decompositions. It basically amounts to calling the CX algorithm to choose columns from A and then calling the same CX algorithm on C^T to get columns of C^T. These columns of C^T are rows of C, and the algorithm keeps the corresponding rows of A. The following algorithm takes as input a matrix $A \in \mathbb{R}^m \times n$, a matrix $C \in \mathbb{R}^{m \times c}$ consisting of c columns of A, a rank parameter k, and an error parameter ϵ, and it returns as output matrices $R \in \mathbb{R}^{r \times n}$ consisting of r rows of A, a matrix $W \in \mathbb{R}^{c \times r}$ consisting of the corresponding r rows of C, and a matrix $U \in \mathbb{R}^{r \times c}$. It does the following steps.

1. Compute probabilities $p_i = \frac{1}{c}\left\|\left(U_C^T\right)^{(i)}\right\|_2^2$, for all $i \in [m]$.
2. Construct a sampling matrix S_R and a diagonal rescaling matrix D_R.
3. Construct $R = D_R S_R^T A$, consisting of a few rescaled rows of A, and return it.
4. Construct $W = D_R S_R^T C$, consisting of a few rescaled rows of C, and return it.
5. Let $U = W^{\dagger}$, and return it.

Here is what we can prove regarding this algorithm

Theorem 4 *Given matrices A and C. If we choose $r = O\left(c \log(c)/\epsilon^2\right)$, then*

$$\|A - CUR\|_F \leq (1+\epsilon)\ \left\|A - CC^{\dagger}A\right\|_F .$$

Proof: Consider the problem of approximating the solution to

$$\min_{X \in \mathbb{R}^{c \times n}} \|CX - A\|_F$$

by randomly sampling rows from C and A. Then, we have that

$$\left\| A - \underbrace{C}_{C} \underbrace{\left(D_R S_R^T C\right)^{\dagger}}_{U} \underbrace{D_R S_R^T A}_{R} \right\|_F \leq (1+\epsilon) \left\| A - CC^{\dagger} A \right\|_F$$

which establishes the result.

$\diamond$

Remark. Clearly, we can combine the two previous results, which gives us the stronger CUR matrix decomposition, as follows:

$$\begin{aligned} \|A - CUR\|_F &\leq (1+\epsilon) \left\| A - CC^{\dagger} A \right\|_F \\ &\leq (1+\epsilon)^2 \|A - A_k\|_F \\ &\leq \left(1+\epsilon'\right) \|A - A_k\|_F . \end{aligned}$$

Running time. Let's say a few words about the running time of these $(1+\epsilon)$ relative-error CX and CUR matrix decompositions. The bottleneck to both of these algorithms is the computation of the importance sampling probabilities, which depend on the leverage scores relative to the best rank-k approximation to A. Thus, naively, one could spend $\Theta\left(n^3\right)$ time, computing the full SVD and use that to compute the leverage scores. One might hope to compute an approximation to the best rank k approximation to A and use the leverage scores from that. In this case, the running time of both of these algorithms boils down to the time to compute a low-rank approximation to A. As we have seen, this is roughly $O(mnk)$ for deterministic iterative methods and roughly $(mn\log(k))$ for randomized methods. (Actually, it could be even faster, depending on the values of parameters, if one uses the input-sparsity-time projection algorithms that we are not going to be able to cover). That basically works, meaning essentially that CX/CUR decompositions can be computed in "random projection time." This is true in theory as well as in practical implementations. There are, however, some subtleties (that appear even for traditional deterministic algorithms for approximating subspaces) that we should point out.

The basic issue is that the problem of computing the leverage scores relative to the best rank k approximation to a matrix is *not* a well-posed problem. If there is a strong eigenvalue gap assumption or if the matrix is rectangular, then it is, but it is not in general. (This is also true for approximating subspaces, e.g., with traditional deterministic iterative methods.) To see this, recall that a possible matrix of left singular vectors is an identity matrix, and one could have the top k singular values be 1 and the bottom $n-k$ singular values be slightly less than 1. In this case, if any of the singular values that is less than 1 "swaps" with one of those that is equal to 1, which could happen with a very small perturbation, then the corresponding leverage scores relative to the best rank k approximation to A would change completely. To deal with this issue, we instead ask for leverage scores that are good relative to some subspace that is close to the best rank k approximation to A. (BTW, this is similar to the solution employed by traditional deterministic algorithms for approximating subspaces.)

Here is the definition of close subspaces.

Definition 5 *Given a matrix $A \in \mathbb{R}^{m\times n}$ and a rank parameter $k \ll \min\{m,n\}$, let A_k be the best rank k approximation to A. Let S_ϵ be the set of rank k matrices that are a good approximation to A, in the sense that*

$$S_\epsilon = \left\{ X \in \mathbb{R}^{m\times n} : \mathit{rank}(X) = k \text{ and } \|A - X\|_\xi \leq (1+\epsilon) \|A - A_k\|_\xi \right\},$$

where $\|\cdot\|_\xi$ is a matrix norm.

Remark. Note that the notion of closeness here depends on the norm used to measure closeness. One obtains somewhat different results depending on whether one uses the spectral versus the Frobenius norm.

Given this definition, here is a notion of approximate leverage scores, that is approximate not only in that individual elements can be up to a factor of β off, but also that they can be only approximate with respect to some subspace that is close to the best rank k approximation to A.

Definition 6 *We will call the numbers $\hat{p}_i$ (for all $i \in [m]$) the β-approximate normalized leverage scores of A relative the best rank k approximation to A if there exists a matrix $X \in S_\epsilon$ such that*

$$\hat{p}_i \geq \frac{\beta}{k} \left\| (U_X)_{(i)} \right\|_2^2 \quad \text{and} \quad \sum_{i=1}^m \hat{p}_i = 1,$$

where here $U_X \in \mathbb{R}^{n\times k}$ is a matrix of left singular vectors of X.

Here is an algorithm to approximate these scores. Basically, it does a random projection to construct a tall matrix, and then it calls the previous fast algorithm to approximate the leverage scores of tall matrices. This algorithm takes as input a matrix $A \in \mathbb{R}^{m\times n}$, with rank$(A) = \rho$, and a rank parameter $k \ll \rho$, and it returns as output a vector of numbers $\{p_i\}_{i=1}^m$ that is a probability distribution. The algorithm does the following steps.

1. Construct a random projection matrix $\Pi \in \mathbb{R}^{n\times 2k}$ with i.i.d. Gaussian entries.
2. Compute the matrix $B = \left(AA^T\right)^q A\Pi \in \mathbb{R}^{m\times 2k}$.
3. Compute approximations to the leverage scores of the "tall" matrix B with the previous algorithm. Let $\hat{\ell}_i$ be the returned approximations.
4. Return $p_i = \frac{\hat{\ell}_i}{\sum_{j=1}^m \hat{\ell}_j}$.

Here is what we can say about this algorithm.

- This algorithm computes normalized scores that are $1 \pm \epsilon$ approximations to the leverage scores of the best rank k approximation to A, with constant probability.
- If $q = 0$, then this provides bounds with respect to the Frobenius norm notion of closeness in Definition 5; while if $q > 0$, then this can be used to provide bounds with respect to the spectral norm notion of closeness in Definition 5.

- The precise theoretical statement of the running time of this algorithm is rather complex, and it depends on whether one is interested in Frobenius or spectral norm approximations of nearby subspaces from Definition 5.
- Empirically, with reasonably-good implementations, if $q = 0$, then the algorithm takes roughly "random projection time," since that is the computational bottleneck, and while one can get reasonable reconstruction error, the actual leverage scores relative to the best rank k space are poorly approximated. If q is a small integer, then the algorithm takes somewhat longer, but one gets better spectral norm bounds and one approximates the actual leverage scores relative to the best rank k space quite well. Clearly, additional iterations beyond that can be slower even than traditional deterministic methods. See one of the sections in the long version of the Gittens-Mahoney Nyström paper for details on these empirical claims.