Lecture 11: Streaming, Sketching, and Sampling (Chapter 6 of Textbook B)

Jinwoo Shin

AI503: Mathematics for AI

Think about this practical scenario;

Input data is too large to be stored in random access memory.

In this lecture, we focus on how to solving such massive data problems via (1) *Streaming*, (2) *Sketching*, and (3) *Sampling*.

Roadmap

KAIST AI
Graduate School of AI

(1) Streaming

(2) Sampling

(3) Sketching

Roadmap

KAIST AI
Graduate School of AI

(1) Streaming

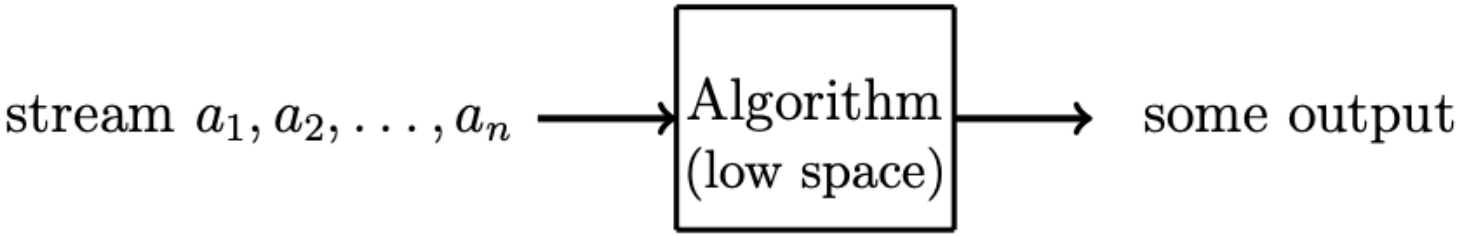
(2) Sampling

(3) Sketching

Streaming model

One model for dealing with such massive data problems is the *streaming* model.

- $a_1, a_2, \ldots, a_n$ arrive one at a time.
- The goal is for our algorithm to compute some statistics, property or summary of these data without using too much memory (i.e., less than n).
- To be specific, assume that each a_i might be an integer in $\{1, \ldots, m\}$ where $m = 2^b$, our goal is to produce some desired output using space polynomial in b and $\log n$.



Representative Problems in Streaming Model

KAIST AI
Graduate School of AI

In the subsequent slides, we dig into the following important and representative problems in a streaming model.

- Computing number of distinct elements in a data stream
- Computing number of occurrences of a given element in a data stream
- Detecting the majority and frequent elements in a data stream

Representative Problems in Streaming Model

KAIST AI
Graduate School of AI

In the subsequent slides, we dig into the following important and representative problems in a streaming model.

- Computing number of distinct elements in a data stream
- Computing number of occurrences of a given element in a data stream
- Detecting the majority and frequent elements in a data stream

Number of Distinct Elements in a Data Stream

KAIST AI
Graduate School of AI

Problem setup

- Consider a sequence $a_1, a_2, \ldots, a_n$ of n elements, each a_i an integer in the range 1 to m where n and m are very large.
- Suppose we wish to determine the number of distinct a_i in the sequence.

Naive solution

- Just store a bit-vector that records which elements have been so far and which have not.
- However, this method requires $O(n \log m)$ space by storing a list of all distinct elements that have been.

Number of Distinct Elements in a Data Stream

However, we desire an algorithm that requires less than $O(n)$ space (e.g., $O(\log n)$)

At first, we are going to show that exact deterministic algorithm has a lower bound on memory.

Then, we will show that randomization and approximation can get around this problem.

Algorithm for the Number of Distinct Elements

Any exact deterministic algorithm must use at least m bits of memory on some sequence of length $m + 1$.

- Assume that we have seen $a_1, a_2, \ldots, a_m$. Suppose for sake of contradiction that our algorithm uses less than m bits of memory.
- We have $2^m - 1$ possible cases, yet the memory is capable of only $\leq 2^{m-1}$ states.
- Hence, there exists two different subsets S_1, S_2 of $\{1, 2, \ldots, m\}$, leading to the same memory.
- If S_1, S_2 are of different sizes, the algorithm is incorrect for one of them. Otherwise, if the next element is in $S_1 \setminus S_2$, the algorithm will give the same answer for both cases and be incorrect for one of them.

To beat the lower bound, consider approximating the number of distinct elements.

- Our algorithm will produce a number that is within a constant factor of the correct answer using randomization and thus a small probability of failure.

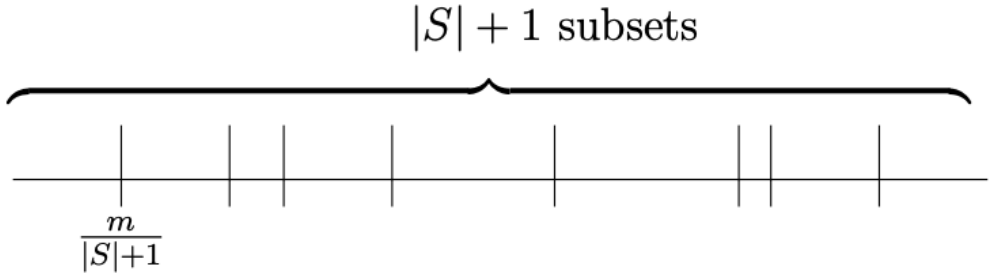
Algorithm for the Number of Distinct Elements

The intuition is as follows:

- Suppose the set S of distinct elements was itself chosen uniformly at random from $\{1, \ldots, m\}$.
- Let *min* denotes the minimum element in S.
- Then, the expected value of *min* would be
  - $\frac{m}{2}$ if there was one distinct element.
  - $\frac{m}{3}$ if there was two distinct elements.
  - $\frac{m}{K+1}$ if there was K distinct elements.

Algorithm for the Number of Distinct Elements

More generally, for a random set S, the expected value of the minimum is approximately $\frac{m}{|S|+1}$.



Solving $min = \frac{m}{|S|+1}$ yields $|S| = \frac{m}{min} - 1$.

This suggests keeping track of the minimum element in $O(\log m)$ space in order to give an estimate of $|S|$.

Algorithm for the Number of Distinct Elements

However, in general, the set S might not have been chosen uniformly at random.

- In particular, if the elements of S were obtained by selecting the $|S|$ smallest elements of $\{1, 2, \ldots, m\}$, the aforementioned approximation would give a very bad answer.

Fortunately, we can convert our intuition into an algorithm that works well with high probability on every sequence via hashing.

Algorithm for the Number of Distinct Elements

Specifically, we will use a hash function h where

$$h : \{1, 2, \ldots, m\} \to \{0, 1, 2, M - 1\},$$

and then instead of keeping track of the minimum element $a_i \in S$, we will keep track of the minimum *hash value*.

The question now is: what properties of a hash function do we need?

Since we need to store h, we cannot use a totally random mapping since that would take too many bits.

Luckily, a pairwise independent hash function, which can be stored compactly is sufficient.

Algorithm for the Number of Distinct Elements

2-Universal (Pairwise Independent) Hash Functions

- A set of hash functions

$$H = \{h | h : \{1, 2, \ldots, m\} \to \{0, 1, 2, \ldots, M-1\}\}$$

is *2-universal* or *pairwise independent* if for all x and y in $\{1, 2, \ldots, m\}$ with $x \neq y$, $h(x)$ and $h(y)$ are each equally likely to be any element of $\{0, 1, 2, \ldots, M-1\}$ and are statistically independent. Namely,

$$\text{Prob}_{h \sim H}(h(x) = w \text{ and } h(y) = z) = \frac{1}{M^2}$$

Algorithm for the Number of Distinct Elements

2-Universal (Pairwise Independent) Hash Functions

- An example of a 2-universal family of hash functions.
  - Let M be a prime greater than m. For each pair of integers a and b in the range $[0, M-1]$, define a hash function.

$$h_{ab}(x) = ax + b \pmod{M}$$

  - To store the hash function h_{ab}, store the two integers a and b. This requires only $O(\log M)$ space.

Algorithm for the Number of Distinct Elements

KAIST AI
Graduate School of AI

2-Universal (Pairwise Independent) Hash Functions

- An example of a 2-universal family of hash functions.
  - To see that h_{ab} constructs the family is 2-universal, note that $h(x) = w$ and $h(y) = z$ if and only if

$$\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} w \\ z \end{pmatrix} \pmod{M}$$

  if $x \neq y$, the matrix $\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix}$ is invertible modulo M. Thus

$$\begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix}^{-1} \begin{pmatrix} w \\ z \end{pmatrix} \pmod{M}$$

  and for each $\begin{pmatrix} w \\ z \end{pmatrix}$ there is a unique $\begin{pmatrix} a \\ b \end{pmatrix}$. Hence

$$\text{Prob}\big(h(x) = w \text{ and } h(y) = z\big) = \frac{1}{M^2}$$

  and H is 2-universal.

Algorithm for the Number of Distinct Elements

Let $b_1, b_2, \ldots, b_d$ be the distinct values that appear in the input.

Select h from the 2-universal family of hash functions H.

Then the set $S = \{h(b_1), h(b_2), \ldots, h(b_d)\}$ is a set of d random and pairwise independent values from the set $\{0, 1, 2, \ldots, M-1\}$.

Lemma 1.

With probability at least $\frac{2}{3} - \frac{d}{M}$, we have $\frac{d}{6} \leq \frac{M}{min} \leq 6d$, where min is the smallest element of S.

Remark. $\frac{M}{min}$ is a good estimate for d, the number of distinct elements in the input, where $min = min(S)$.

Representative Problems in Streaming Model

KAIST AI
Graduate School of AI

In the subsequent slides, we dig into the following important and representative problems in a streaming model.

- Computing number of distinct elements in a data stream
- Computing number of occurrences of a given element in a data stream
- Detecting the majority and frequent elements in a data stream

Algorithm for the Number of Occurrences of a Given Element

Even a naive algorithm only requires at most $\log n$ space where n is the length of the stream.

- Just increase a counter when encountering the given element.

This algorithm is widely used since it needs only $\log n$, but one can further reduce the required space to $\log \log n$ with approximation and randomness.

The algorithm will be explained in the subsequent slide as the technique may give insight into how to solve some other problems.

Algorithm for the Number of Occurrences of a Given Element

Consider a string if 0's and 1's of length n in which we wish to count the number of occurrences of 1's.

- Let m be the number of 1's that occur in the sequence.
- Keep a value k such that 2^k is approximately the number m of occurrences.
- Storing k requires only $\log \log n$ bits of memory.

Algorithm 1. Number of occurrences of a given element

- Start with $k = 0$.
- For each occurrence of a 1, add one to k with probability $1/2^k$.
- At the end of the sequence, the quantity $2^k - 1$ is the estimate of m.

Representative Problems in Streaming Model

KAIST AI
Graduate School of AI

In the subsequent slides, we dig into the following important and representative problems in a streaming model.

- Computing number of distinct elements in a data stream
- Computing number of occurrences of a given element in a data stream
- Detecting the majority and frequent elements in a data stream

Majority Algorithm

The goal is to determine whether there exist some number which occurs more than $n/2$ times where n is the length of a sequence.

- To be specific, we are given a stream of integers $a_1, a_2, \ldots, a_n$, each a_i belonging to $\{1, 2, \ldots, m\}$
- We want to determine whether there is some $s \in \{1, 2, \ldots, m\}$ which occurs more than $n/2$ times and if so which s.

With a deterministic algorithm, it requires $\Omega(\min(n, m))$ space to solve the problem exactly on read-once streaming data.

Majority Algorithm

Surprisingly, we can bypass the aforementioned lower bound by slightly weakening our goal. We present a majority algorithm which reduces required space.

Algorithm 2. Majority algorithm

- Store a_1 and initialize a counter to one.
- For each subsequent a_i, if a_i is the same as the currently stored item, increment the counter by one. If it differs, decrease the counter by one provided the counter is nonzero.
- If the counter is zero, then store a_i, and set the counter to one.

This algorithm guarantees to output majority element if it exists. However, it may outputs false positive, (but, not "false negatives")

It requires $O(\log m + \log n)$ space.

Algorithm Frequent

Next, by modifying the majority algorithm, one can devise an algorithm that detects items with frequency above some threshold.

Algorithm 3. Algorithm frequent

- Maintain a list of items being counted. Initially the list is empty.
- For each item, if it is the same as some item on the list, increment its counter by one.
- If it differs from all the items on the list, then if there are less than k items on the list, add the item to the list with its counter set to one. If there are already k items on the list, decrease each of the current counters by one.
- Delete an element from the list if its count becomes zero.

This algorithm requires $O(k \log n + k \log m)$ space by keeping k counters instead of just one counter as did in the majority algorithm.

Roadmap

KAIST AI
Graduate School of AI

(1) Streaming

(2) Sampling

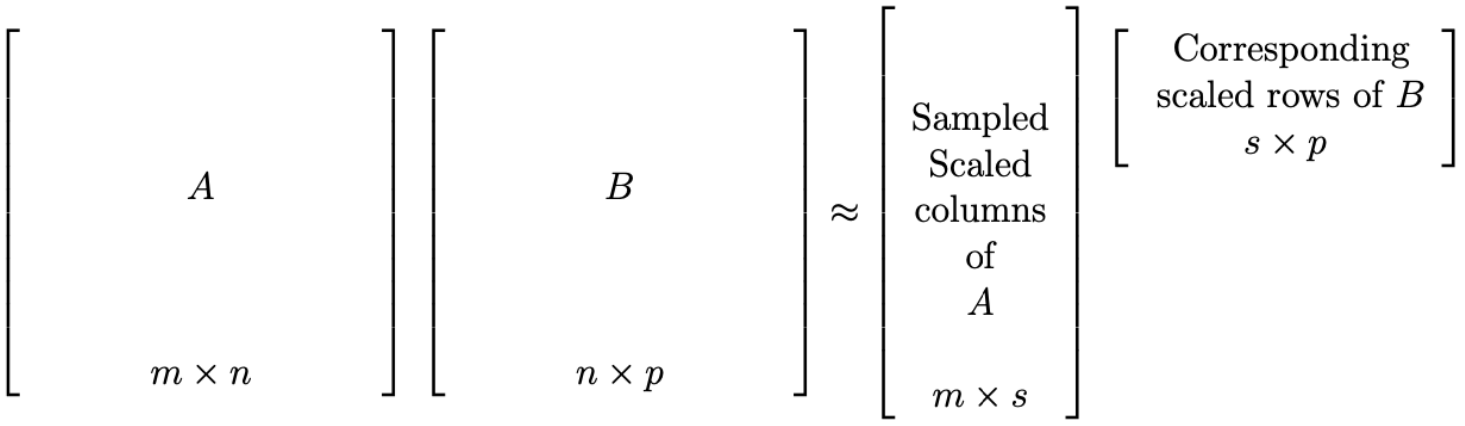
(3) Sketching

Matrix Multiplication using Sampling

How to reduce computation for multiplication AB of two huge matrix A and B?

We can achieve the reduction by using sampling and approximation.

The goal is to find two matrix of which multiplication approximates AB.



Matrix Multiplication using Sampling

KAIST AI
Graduate School of AI

Theorem 2.

Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. The product AB can be estimated by CR, where $C \in \mathbb{R}^{m \times s}$ and $R \in \mathbb{R}^{s \times n}$ consist of columns of A and rows of B, respectively. The corresponding rows and columns are picked according to *length-squared distribution* and scaled. The error is bounded by:

$$E(\|AB - CR\|_F^2) \leq \frac{1}{s}\|A\|_F^2 \|B\|_F^2.$$

Proof. Observe that

$$AB = \sum_{k=1}^{n} A(:,k)B(k,:)$$

$A(:,k)$ denote the k^{th} column of A; $A(:,k)$ is a $m \times 1$ matrix. Likewise, $B(k,:)$ is the k^{th} row of B; $B(k,:)$ is a $1 \times n$ matrix. Note that for each value of k, $A(:,k)B(k,:)$ is an $m \times p$ matrix each element of which is a single product of elements A and B.

Matrix Multiplication using Sampling

Proof cont'd. We can form a random matrix

$$X = \begin{cases} \frac{1}{p_1} A(:,1)B(1,:) & \text{with prob. } p_1 \\ \frac{1}{p_2} A(:,2)B(2,:) & \text{with prob. } p_2 \\ & \vdots \\ \frac{1}{p_n} A(:,n)B(n,:) & \text{with prob. } p_n \end{cases}$$

Then, $E(X) = \sum_{k=1}^{n} \text{Prob}[Z = k] \frac{1}{p_k} A(:,k)B(k,:) = AB$.

Proof cont'd. So, the expected Frobenius error is

$$
\begin{aligned}
E(\|AB - X\|_F^2) &= \mathrm{Var}(X) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{p} \mathrm{Var}(x_{ij}) \\
&= \sum_{ij} E(x_{ij}^2) - \sum_{ij} E(x_{ij})^2 \\
&= \sum_{ij}\sum_{k=1}^{n} p_k \left(\frac{1}{p_k^2} a_{ik}^2 b_{kj}^2\right) - \|AB\|_F^2 \\
&= \sum_{k=1}^{n} \frac{1}{p_k} \|A(:,k)\|_2^2 \|B(k,:)\|_2^2 - \|AB\|_F^2.
\end{aligned}
$$

Matrix Multiplication using Sampling

KAIST AI
Graduate School of AI

Proof cont'd.

$$E(\|AB - X\|_F^2) = \sum_{k=1}^{n} \frac{1}{p_k} \|A(:,k)\|_2^2 \|B(k,:)\|_2^2 - \|AB\|_F^2$$

Now choose p_k (length squared sampling) as

$$p_k = \frac{\|A(:,k)\|_2^2}{\|A\|_F^2}.$$

Then,

$$E(\|AB - X\|_F^2) \leq \|A\|_F^2 \|B\|_F^2$$

Matrix Multiplication using Sampling

KAIST AI
Graduate School of AI

Proof cont'd.

$$E(\|AB - X\|_F^2) \leq \|A\|_F^2 \|B\|_F^2$$

However, this bound is not good enough. To reduce the variance term, we can generate s samples $X_i, i = 1, \ldots, s$. We take $\frac{1}{s}\sum_{i=1}^{s} X_i$ as an estimator of matrix AB. Then,

$$\text{Var}(\frac{1}{s}\sum_{i=1}^{s} X_i) = \frac{1}{s}\text{Var}(X) \leq \frac{1}{s}\|A\|_F^2 \|B\|_F^2$$

Finally, we can interpret our estimator as the matrix multiplication. Let $k_1, \ldots, k_s$ be the k's chosen in each trial. Expanding this, gives:

$$\frac{1}{s}\sum_{i=1}^{s} X_i = \frac{1}{s}(\frac{A(:,k_1)B(k_1,:)}{p_{k_1}} + \frac{A(:,k_2)B(k_2,:)}{p_{k_2}} + \cdots + \frac{A(:,k_s B(k_s,:))}{p_{k_s}}).$$

Matrix Multiplication using Sampling

KAIST AI
Graduate School of AI

Proof cont'd.

$$\frac{1}{s}\sum_{i=1}^{s} X_i = \frac{1}{s}\left(\frac{A(:,k_1)B(k_1,:)}{p_{k_1}} + \frac{A(:,k_2)B(k_2,:)}{p_{k_2}} + \cdots + \frac{A(:,k_s B(k_s,:))}{p_{k_s}}\right).$$

We will find it convenient to write this as the product of an $m \times s$ matrix with a $s \times p$ matrix as the following slide shows. Let C be the $m \times s$ matrix consisting of the following columns which are scaled versions of the chosen columns of A:

$$\frac{A(:,k_1)}{\sqrt{sp_{k_1}}}, \frac{A(:,k_2)}{\sqrt{sp_{k_2}}}, \ldots, \frac{A(:,k_s)}{\sqrt{sp_{k_s}}}$$

Likewise, define R to be the $s \times p$ matrix with the corresponding rows of B similarly scaled, namely, R has rows

$$\frac{B(k_1,:)}{\sqrt{sp_{k_1}}}, \frac{B(k_2,:)}{\sqrt{sp_{k_2}}}, \ldots, \frac{B(k_s,:)}{\sqrt{sp_{k_s}}}.$$

Then, $CR = \frac{1}{s}\sum_{i=1}^{s} X_i$.■

Roadmap

KAIST AI
Graduate School of AI

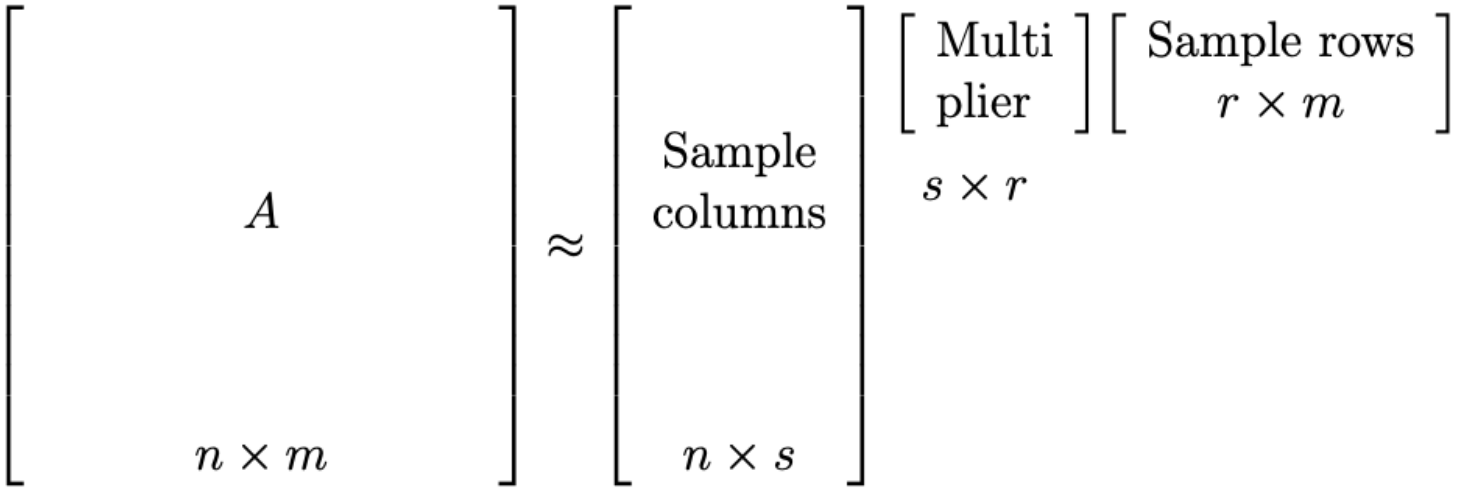
(1) Streaming

(2) Sampling

(3) Sketching

Sketching

Next, we want to show that for any data matrix, a sample of columns and rows, each picked according to the length squared distribution provides a good *sketch* of matrix, meaning that one can find an good approximation $A \approx CUR$ with an error bound estimation. Here $A \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{n \times s}$, $U \in \mathbb{R}^{s \times r}$, and $R \in \mathbb{R}^{r \times m}$.



Sketch of a Large Matrix

We first present a simple idea that does not work, but that motivates an idea that does; application to the Theorem 2 by writing A as AI.

- Idea: write A as AI, where I is the $n \times n$ identity matrix. Approximate the product AI using Theorem 2, i.e., by sampling s columns of A according to the length-squared distribution. Then as in the Theorem 2, write $AI \approx CW$, where W consists of a scaled version of the s rows of I corresponding to the s columns of A that were picked. Theorem 2 bounds the error $\|A - CW\|_F^2$ by $\|A\|_F^2 \|I\|_F^2 / s = \frac{n}{s} \|A\|_F^2$.
- Limitation: we would like the error to be a small fraction of $\|A\|_F^2$ which would require $s \geq n$, which clearly of no use since this would pick as many or more columns than the whole of A.

Sketch of a Large Matrix

The main concept is to use the identity-like matrix P instead of I in the just prior discussion.

(Although P does not appear in the following Theorem, P takes a crucial role in lemmas for the Theorem.)

Theorem 3.

Let $A \in \mathbb{R}^{m \times n}$. We can compute $C \in \mathbb{R}^{m \times s}$ by sampling s columns of A (via length squared sampling and scaling), $R \in \mathbb{R}^{r \times n}$ by sampling r rows of A (via length squared sampling and scaling), and $U \in \mathbb{R}^{s \times r}$ such that

$$E(\|A - CUR\|_2^2) \leq (\frac{2}{\sqrt{r}} + \frac{2r}{s})\|A\|_F^2.$$

We postpone the proof, before that, we introduce lemmas for the proof.

Sketch of a Large Matrix

KAIST AI
Graduate School of AI

Lemma 1.

Assume RR^T is invertible. Then, $P = R^T(RR^T)^{-1}R$ has the following property:
(i) P acts as the identity on the row space of R. i.e., $P\boldsymbol{x} = \boldsymbol{x}$ for every $\boldsymbol{x}$ of the form $\boldsymbol{x} = R^T\boldsymbol{y}$.
(ii) if $\boldsymbol{x}$ is orthogonal to the row space of R, then $P\boldsymbol{x} = 0$

Proof. For any $\boldsymbol{x} = R^T\boldsymbol{y}$ in the row space of R, we have

$$P\boldsymbol{x} = R^T(RR^T)^{-1}R\boldsymbol{x} = R^T(RR^T)^{-1}RR^T\boldsymbol{y} = R^T\boldsymbol{y} = \boldsymbol{x}.$$

Also, if $\boldsymbol{x}$ is orthogonal to every row of R, then $R\boldsymbol{x} = 0$, so $P\boldsymbol{x} = 0$.

Lemma 2.

Let R be constructed by length squared sampling and scaling r rows of A. Let $P = R^T(RR^T)^{-1}R$. Then, $A \approx AP$ and the error $E(\|A - AP\|_2^2)$ is at most $\frac{1}{\sqrt{r}}\|A\|_F^2$.

Remark. This lemma shows that P is a good approximation of identity matrix.

Proof. First, recall that

$$\|A - AP\|_2^2 = \max_{\{\boldsymbol{x}:\|\boldsymbol{x}\|=1\}} \|(A - AP)\boldsymbol{x}\|_2^2.$$

Now suppose $\boldsymbol{x}$ is in the row space V of R.

From Lemma 1, $P\boldsymbol{x} = \boldsymbol{x}$, so for $\boldsymbol{x} \in V, (A - AP)\boldsymbol{x} = 0$.

Proof cont'd. Since every vector can be written as a sum of a vector in V plus a vector orthogonal to V, this implies that the maximum must therefore occur at some $\boldsymbol{x} \in V^{\perp}$.

For such $\boldsymbol{x}$, by Lemma 1, $(A - AP)\boldsymbol{x} = A\boldsymbol{x}$. Thus,

$$\begin{aligned}
\|(A - AP)\boldsymbol{x}\|_2^2 &= \|A\boldsymbol{x}\|_2^2 \\
&= \boldsymbol{x}^T A^T A \boldsymbol{x} \\
&= \boldsymbol{x}^T (A^T A - R^T R)\boldsymbol{x} \\
&\leq \|A^T A - R^T R\|_2 \cdot \|\boldsymbol{x}\|_2^2 \\
&= \|A^T A - R^T R\|_2 \\
&= \|A^T A - R^T R\|_F.
\end{aligned}$$

This implies that $\|(A - AP)\boldsymbol{x}\|_2^2 \leq \|A^T A - R^T R\|_F$. So, it suffices to prove that $E(\|A^T A - R^T R\|_F^2) \leq \|A\|_F^4 / r$. From Theorem 2, since we can think of $R^T R$ as a way of estimating $A^T A$ by picking according to length-squared distribution columns of A^T, i.e., rows of A. This proves Theorem 3. ■

Now, we are ready to prove Theorem 3.

Theorem 3.

Let $A \in \mathbb{R}^{m \times n}$. We can compute $C \in \mathbb{R}^{m \times s}$ by sampling s columns of A (via length squared sampling and scaling), $R \in \mathbb{R}^{r \times n}$ by sampling r rows of A (via length squared sampling and scaling), and $U \in \mathbb{R}^{s \times r}$ such that

$$E(\|A - CUR\|_2^2) \leq (\frac{2}{\sqrt{r}} + \frac{2r}{s})\|A\|_F^2.$$

Proof. Construct R by length squared sampling and scaling r rows of A. Construct C by length squared sampling s columns of A, and let U be the corresponding scaled rows of $R^T(RR^T)^{-1}$.

From Theorem 2,

$$E(\|AP - CUR\|_2^2) \leq E(\|AP - CUR\|_F^2) \leq \frac{\|A\|_F^2\|P\|_F^2}{s} \leq \frac{r}{s}\|A\|_F^2.$$

Proof cont'd. From Lemma 2, $E(\|A - AP\|_2^2) \leq \frac{1}{\sqrt{r}}\|A\|_F^2$.

By triangle inequality, $\|A - CUR\|_2 \leq \|A - AP\|_2 + \|AP - CUR\|_2$, which in turn implies that $\|A - CUR\|_2^2 \leq 2\|A - AP\|_2^2 + 2\|AP - CUR\|_2^2$.

Then, $E(\|A - CUR\|_2^2) \leq (\frac{2}{\sqrt{r}} + \frac{2r}{s})\|A\|_F^2$. ■

Questions?