EE270
Large scale matrix computation, optimization and learning

Instructor : Mert Pilanci

Stanford University

Tuesday, Jan 2 2020

Lecture 2

Randomized Linear Algebra

Approximate Matrix Multiplication

Randomized Algorithms

- algorithms that employ a degree of randomness to guide its behavior
- we hope to achieve good performance in the *average case*
- the algorithm's performance is a random variable

Randomized Algorithms

Are approximations satisfactory?

- depends on the application
- often acceptable for minimizing training error up to statistical precision
- implicit regularization effect
- when not satisfactory, they can be used as initializers for exact and costly methods
- moreover, exact methods might not work at all for very large scale problems

Probability background and notation

- X : discrete random variable taking values $x_1, \ldots, x_n$
- Expectation $\mathbb{E}[X]$

$$\mathbb{E}[X] = \sum_i x_i \mathbb{P}[X = x_i]$$

- Properties:

  linearity

- $\mathbb{E}[cX] = c\mathbb{E}[X]$ where c is a constant
- $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ where X and Y are two random variables

Probability background and notation

- Variance

$$\mathbf{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

- $$\begin{aligned}\mathbf{Var}[X] &= \mathbb{E}[X^2] - 2\mathbb{E}[X\mathbb{E}X] + \mathbb{E}[\mathbb{E}[X]^2] \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2\end{aligned}$$

Probability background and notation

$$\begin{aligned}\mathbf{Var}[X] &= \mathbb{E}[(X-\mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2]-\mathbb{E}[X]^2\end{aligned}$$

- Variance properties
- $\mathbf{Var}[cX] = c^2\mathbf{Var}[X]$ where c is a constant

Probability background and notation

$$\begin{aligned}\mathbf{Var}[X] &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2\end{aligned}$$

- Variance properties
- $\mathbf{Var}[cX] = c^2\mathbf{Var}[X]$ where c is a constant
- $\mathbf{Var}[X + Y] = \mathbb{E}(X + Y)^2 - (\mathbb{E}[X] + \mathbb{E}[Y])^2 =$ $\mathbb{E}[X^2] - \mathbb{E}[X]^2 + \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 + 2(\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y])$
- $\mathbf{Var}[X + Y] = \mathbf{Var}[X] + \mathbf{Var}[Y]$ for X, Y uncorrelated ($\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$)
- independence implies uncorrelatedness

Probability background and notation

- Averaging independent realizations reduce variance

  Let X_1 and X_2 be independent and identically distributed
- $$\mathbf{Var}[\tfrac{X_1+X_2}{2}] = \tfrac{1}{4}\mathbf{Var}[X_1 + X_2]$$
  $$= \tfrac{1}{4}\left(\mathbf{Var}[X_1] + \mathbf{Var}[X_2]\right) = \tfrac{1}{2}\mathbf{Var}[X_1]$$

Example: randomized counting

- **Deterministic counting**

  Set $\texttt{counter} = 0$

  Increment $\texttt{counter} \leftarrow \texttt{counter} + 1$ for every item
- space complexity is $\log_2(n)$ bits for n items

Example: randomized counting

- **Deterministic counting**

  Set $\texttt{counter} = 0$

  Increment $\texttt{counter} \leftarrow \texttt{counter} + 1$ for every item
- space complexity is $\log_2(n)$ bits for n items
- **Approximate randomized counting**

  keep only the exponent to reduce space.
- For example, in base 2, the counter can estimate the count to be 1, 2, 4, 8, 16, 32, and all of the powers of two.

Example: randomized counting

- **Deterministic counting**

  Set `counter` $= 0$

  Increment `counter` $\leftarrow$ `counter` $+ 1$ for every item
- space complexity is $\log_2(n)$ bits for n items
- **Approximate randomized counting**

  keep only the exponent to reduce space.
- For example, in base 2, the counter can estimate the count to be 1, 2, 4, 8, 16, 32, and all of the powers of two.
- *flip a coin* the number of times of the counter's current value. If it comes up Heads each time, then increment the counter. Otherwise, do not increment it.
- space complexity is $\log_2 \log_2(n)$ bits for n items

Example: randomized counting

- **Approximate randomized counting**

  Set $X = 0$

  Increment $X \leftarrow X + 1$ with probability 2^{-X} for every item.

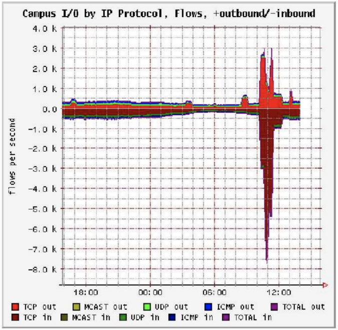
  Output $\tilde{n} = 2^X - 1$
- space complexity is $\log_2 \log_2(n)$ bits for n items

Example: randomized counting

- **Approximate randomized counting**
  Set $X = 0$
  Increment $X \leftarrow X + 1$ with probability 2^{-X} for every item.
  Output $\tilde{n} = 2^X - 1$
- space complexity is $\log_2 \log_2(n)$ bits for n items
  Lemma 1 $\mathbb{E}\tilde{n} = \mathbb{E}2^X - 1 = n$ (Unbiased)
  $\mathbf{Var}[\tilde{n}] \leq \frac{1}{2}n^2$
- Variance can be reduced by averaging multiple trials
- $\tilde{n}_1, ..., \tilde{n}_r$ i.i.d. trials, $\mathbf{Var}(\frac{1}{r}\sum_{i=1}^{r} n_i) = \frac{1}{r}\mathbf{Var}(\tilde{n}_1)$
  Morris's Algorithm (1977)

A randomized counting application

From Estan-Varghese-Fisk: traces of attacks
Need number of active connections in time slices.



Incoming/Outgoing flows at 40Gbits/second.
Code Red Worm: 0.5GBytes of compressed data per hour (2001).
CISCO: in 11 minutes, a worm infected 500,000,000 machines.

slide credit: Flajolet

Classical Matrix Multiplication Algorithm

Let $A \in R^{n \times d}$ and $B \in R^{d \times p}$

$$(AB)_{ij} = \sum_{k=1}^{d} A_{ik} B_{kj}$$

Classical Matrix Multiplication Algorithm

Let $A \in R^{n \times d}$ and $B \in R^{d \times p}$

$$(AB)_{ij} = \sum_{k=1}^{d} A_{ik} B_{kj}$$

Algorithm 2 Vanilla three-look matrix multiplication algorithm

Input: An $n \times d$ matrix A and an $d \times p$ matrix B

Output: The product AB

```
for i = 1 to n do
    for j = 1 to p do
        (AB)_ij = 0
        for k = 1 to d do
            (AB)_ij += A_ik B_kj
        end for
    end for
end for
```

Classical Matrix Multiplication Algorithm

Let $A \in R^{n \times d}$ and $B \in R^{d \times p}$

$$(AB)_{ij} = \sum_{k=1}^{d} A_{ik} B_{kj}$$

Algorithm 3 Vanilla three-look matrix multiplication algorithm

Input: An $n \times d$ matrix A and an $d \times p$ matrix B

Output: The product AB

```
for i = 1 to n do
    for j = 1 to p do
        (AB)_ij = 0
        for k = 1 to d do
            (AB)_ij += A_ik B_kj
        end for
    end for
end for
```

- Complexity: $O(ndp)$

Faster Matrix Multiplication

Square matrix multiplication $n = d = p$

- Classical $O(n^3)$
- Strassen (1969) $O(n^{2.8074})$
- Coppersmith-Winograd (1990) $O(n^{2.376})$
- Vassilevska Williams (2013) $O(n^{2.3728642})$
- Le Gall (2014) $O(n^{2.3728639})$
- J. Alman and V. Williams (December 2020) $O(n^{2.3728596})$

Faster Matrix Multiplication

Square matrix multiplication $n = d = p$

- Classical $O(n^3)$
- Strassen (1969) $O(n^{2.8074})$
- Coppersmith-Winograd (1990) $O(n^{2.376})$
- Vassilevska Williams (2013) $O(n^{2.3728642})$
- Le Gall (2014) $O(n^{2.3728639})$
- J. Alman and V. Williams (December 2020) $O(n^{2.3728596})$

The greatest lower bound for the exponent of matrix multiplication algorithm is generally called ω.

- $2 \leq \omega$ because one has to read all the n^2 entries and hence $2 \leq \omega < 2.373$
- it is unknown whether $2 < \omega$

Faster Matrix Multiplication

Square matrix multiplication $n = d = p$

- Classical $O(n^3)$
- Strassen (1969) $O(n^{2.8074})$
- Coppersmith-Winograd (1990) $O(n^{2.376})$
- Vassilevska Williams (2013) $O(n^{2.3728642})$
- Le Gall (2014) $O(n^{2.3728639})$
- J. Alman and V. Williams (December 2020) $O(n^{2.3728596})$

The greatest lower bound for the exponent of matrix multiplication algorithm is generally called ω.

- $2 \leq \omega$ because one has to read all the n^2 entries and hence $2 \leq \omega < 2.373$
- it is unknown whether $2 < \omega$
- some are *galactic algorithms* (Lipton and Regan)

only of theoretical interest and impractical due to large constants

Strassen showed[1] how to use 7 scalar multiplies for 2×2 matrix multiplication

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

classical algorithm

$$\begin{aligned}
M_1 &= A_{11}B_{11} \\
M_2 &= A_{12}B_{21} \\
M_3 &= A_{11}B_{12} \\
M_4 &= A_{12}B_{22} \\
M_5 &= A_{21}B_{11} \\
M_6 &= A_{22}B_{21} \\
M_7 &= A_{21}B_{12} \\
M_8 &= A_{22}B_{22} \\
C_{11} &= M_1 + M_2 \\
C_{12} &= M_3 + M_4 \\
C_{21} &= M_5 + M_6 \\
C_{22} &= M_7 + M_8
\end{aligned}$$

Strassen's algorithm

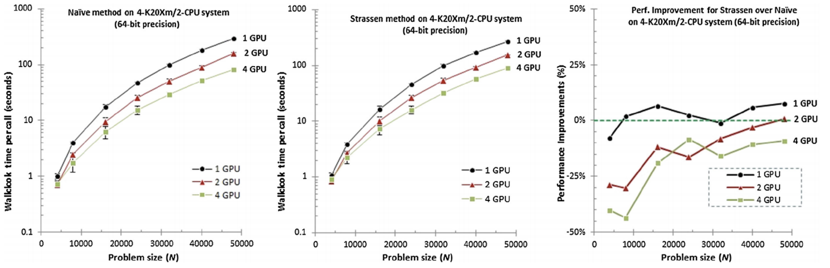
$$\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22})B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

[1] V. Strassen, Gaussian Elimination is not Optimal, 1969

Classical Matrix Multiplication vs Strassen's Method and others

- The constants in fast matrix multiplication methods are high and for a typical application the classical method works better.
- The submatrices in recursion take extra space.
- Because of the limited precision of computer arithmetic on noninteger values, larger errors accumulate

Time comparison: Classical vs Strassen Matrix Multiplication



(a) Wallclock time of Naïve (b) Wallclock time of Strassen (c) Perf. Improvement of Stassen vs. Naïve

Matrix Multiplication on High-Density Multi-GPU Architectures: Theoretical and Experimental Investigations. Zhang and Gao, 2015

Notation

- For a matrix $A \in \mathbb{R}^{n \times d}$
- $A^{(j)} \in \mathbb{R}^{n \times 1}$ denotes the j-th column of A as a column vector
- $A_{(i)} \in \mathbb{R}^{1 \times d}$ denotes i-th row of A is a row vector

Notation

- For a matrix $A \in \mathbb{R}^{n \times d}$
- $A^{(j)} \in \mathbb{R}^{n \times 1}$ denotes the j-th column of A as a column vector
- $A_{(i)} \in \mathbb{R}^{1 \times d}$ denotes i-th row of A is a row vector
- $A = \left[\begin{array}{ccc} A^{(1)} & \ldots & A^{(d)} \end{array} \right]$
- $A = \left[\begin{array}{c} A_{(1)} \\ \vdots \\ A_{(n)} \end{array} \right]$

Notation

- for a vector $x \in \mathbb{R}^n$
- $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$ denotes its Euclidean length (ℓ_2-norm)

Notation

- for a vector $x \in \mathbb{R}^n$
- $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$ denotes its Euclidean length (ℓ_2-norm)
- for a matrix $A \in \mathbb{R}^{n \times d}$
- $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^d |A_{ij}|^2}$ is the Frobenius norm
- $\|A\|_F = \|\mathbf{vec}(A)\|_2)$

  where **vec** reshapes A into an $nd \times 1$ vector

Approximate Matrix Multiplication by random sampling

- matrix multiplication formula

$$(AB)_{ij} = \sum_{k=1}^{d} A_{ik} B_{kj} = A_{(i)} B^{(j)}$$

- $A_{(k)} B^{(k)}$ are **inner products**

Approximate Matrix Multiplication by random sampling

- matrix multiplication formula

$$(AB)_{ij} = \sum_{k=1}^{d} A_{ik} B_{kj} = A_{(i)} B^{(j)}$$

- $A_{(k)} B^{(k)}$ are **inner products**
- same formula as a sum of **outer products**

$$AB = \sum_{k=1}^{d} A^{(k)} B_{(k)}$$

- $A^k B_k$ are rank-1 matrices

Approximate Matrix Multiplication by random sampling

- matrix multiplication as sum of outer products

$$AB = \sum_{k=1}^{d} A^{(k)} B_{(k)}$$

- **basic idea**: sample m indices $i_1, \dots, i_m \in \{1, \dots, d\}$

$$AB \approx^{?} \sum_{t=1}^{m} A^{(i_t)} B_{(i_t)}$$

Required probability background

- Probability, events, random variables
- Expectation, variance, standard deviation
- Conditional probability, independence

A probability refresher will be posted on the course webpage

Approximate Matrix Multiplication by weighted sampling

- matrix multiplication as sum of outer products

$$AB = \sum_{k=1}^{d} A^{(k)} B_{(k)}$$

- **weighted sampling:** sample m indices $i_1, \ldots, i_m \in \{1, \ldots, d\}$ independently with replacement such that
- $\mathbb{P}[i_t = k] = p_k$ for all t

  $p_1, \ldots, p_d$ is a discrete probability distribution

$$AB \approx \frac{1}{m} \sum_{t=1}^{m} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t)}$$

Approximate Matrix Multiplication by weighted sampling

- **weighted sampling:** sample m indices $i_1, \ldots, i_m \in \{1, \ldots, d\}$ independently with replacement such that
- $\mathbb{P}[i_t = k] = p_k$ for all t

$$AB \approx \frac{1}{m} \sum_{t=1}^{m} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t)}$$

- $\mathbb{E}\left[\frac{1}{m} \sum_{t=1}^{m} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t)}\right] =$

Approximate Matrix Multiplication by weighted sampling

- yields a smaller matrix multiplication problem

$$AB \approx \frac{1}{m}\sum_{t=1}^{m}\frac{1}{p_{i_t}}A^{(i_t)}B_{(i_t)} \triangleq CR$$

- $C = \begin{bmatrix} \frac{1}{\sqrt{mp_{i_1}}}A^{(i_1)} & \dots & \frac{1}{\sqrt{mp_{i_m}}}A^{(i_m)} \end{bmatrix}$
- $R = \begin{bmatrix} \frac{1}{\sqrt{mp_{i_1}}}A_{(i_1)} \\ \dots \\ \frac{1}{\sqrt{mp_{i_m}}}A_{(i_m)} \end{bmatrix}$

Approximate Matrix Multiplication

Algorithm 4 Approximate Matrix Multiplication via Sampling

Input: An $n \times d$ matrix A and an $d \times p$ matrix B, an integer m and probabilities $\{p_k\}_{k=1}^{d}$

Output: Matrices CR such that $CR \approx AB$

1: **for** $t = 1$ to m **do**

2: Pick $i_t \in \{1, \ldots, d\}$ with probability $\mathbb{P}[i_t = k] = p_k$ in i.i.d. with replacement

3: Set $C^{(t)} = \frac{1}{\sqrt{mp_{i_t}}} A^{(i_t)}$ and $R_{(t)} = \frac{1}{\sqrt{mp_{i_t}}} B_{(i_t)}$

4: **end for**

Approximate Matrix Multiplication

Algorithm 5 Approximate Matrix Multiplication via Sampling

Input: An $n \times d$ matrix A and an $d \times p$ matrix B, an integer m and probabilities $\{p_k\}_{k=1}^{d}$

Output: Matrices CR such that $CR \approx AB$

1: **for** $t = 1$ to m **do**

2: Pick $i_t \in \{1, ..., d\}$ with probability $\mathbb{P}[i_t = k] = p_k$ in i.i.d. with replacement

3: Set $C^{(t)} = \frac{1}{\sqrt{mp_{i_t}}} A^{(i_t)}$ and $R_{(t)} = \frac{1}{\sqrt{mp_{i_t}}} B_{(i_t)}$

4: **end for**

- We can multiply CR using the classical algorithm
- Complexity $O(nmp)$

Sampling probabilities

- Uniform sampling $p_k = \frac{1}{d}$ for all $k = 1, ..., m$.

$$AB \approx \frac{1}{m} \sum_{t=1}^{m} \frac{1}{d^{-1}} A^{(i_t)} B_{(i_t)} \triangleq CR$$

- $C = \left[\begin{array}{ccc} \frac{\sqrt{d}}{\sqrt{m}} A^{(i_1)} & \dots & \frac{\sqrt{d}}{\sqrt{m}} A^{(i_m)} \end{array} \right]$
- $R = \left[\begin{array}{c} \frac{\sqrt{d}}{\sqrt{m}} A_{(i_1)} \\ \dots \\ \frac{\sqrt{d}}{\sqrt{m}} A_{(i_m)} \end{array} \right]$

AMM mean and variance

$$AB \approx CR = \frac{1}{m} \sum_{t=1}^{m} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t)}$$

- Mean and variance of the matrix multiplication estimator

  Lemma 2
- $\mathbb{E}\left[(CR)_{ij}\right] = (AB)_{ij}$
- $\mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{m}\sum_{k=1}^{d} \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{m}(AB)_{ij}^2$

AMM mean and variance

$$AB \approx CR = \frac{1}{m} \sum_{t=1}^{m} \frac{1}{p_{i_t}} A^{(i_t)} B_{(i_t)}$$

- Mean and variance of the matrix multiplication estimator

  Lemma 2

- $\mathbb{E}\left[(CR)_{ij}\right] = (AB)_{ij}$
- $\mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{m} \sum_{k=1}^{d} \frac{A_{ik}^2 B_{kj}^2}{p_k} - \frac{1}{m}(AB)_{ij}^2$
- $\mathbb{E}\|AB - CR\|_F^2 = \sum_{ij} \mathbb{E}(AB - CR)_{ij}^2 = \sum_{ij} \mathbf{Var}[(CR)_{ij}]$

$$= \frac{1}{m} \sum_{k=1}^{d} \frac{\sum_i A_{ik}^2 \sum_j B_{kj}^2}{p_k} - \frac{1}{m}\|AB\|_F^2$$

$$= \frac{1}{m} \sum_{k=1}^{d} \frac{1}{p_k} \|A^{(k)}\|_2^2 \|B_{(k)}\|_2^2 - \frac{1}{m}\|AB\|_F^2$$

Uniform sampling guarantees

- $p_k = \frac{1}{d}$ for $k = 1, \dots, d$

$$AB \approx CR = \frac{d}{m} \sum_{t=1}^{m} A^{(i_t)} B_{(i_t)}$$

- We can choose sampling set before looking at data (*oblivious*)
- AMM algorithm can be performed in one pass over data

$$\mathbb{E}\|AB - CR\|_F^2 = \frac{d}{m} \sum_{k=1}^{d} \|A^{(k)}\|_2^2 \|B_{(k)}\|_2^2 - \frac{1}{m}\|AB\|_F^2$$

Optimal sampling probabilities

- Optimal sampling probabilities to minimize $\mathbb{E}\|AB - CR\|_F$
  i.e., sum of variances

$$\min_{\substack{p_1,\dots p_d \geq 0 \\ \sum p_k = 1}} \mathbb{E}\|AB - CR\|_F$$

$$= \min_{\substack{p_1,\dots p_d \geq 0 \\ \sum p_k = 1}} \frac{1}{m} \sum_{k=1}^{d} \frac{1}{p_k} \|A^{(k)}\|_2^2 \|B_{(k)}\|_2^2 - \frac{1}{m}\|AB\|_F^2$$

Optimal sampling probabilities

Let $q_1, ..., q_d \in \mathbb{R}$ given

$$\min_{\substack{p_1,\dots p_d \geq 0 \\ \sum p_k = 1}} \sum_{k=1}^{d} \frac{q_k^2}{p_k}$$

- introduce a Lagrange multiplier for the constraint $\sum p_k = 1$

Optimal sampling probabilities

- Nonuniform sampling

$$p_k = \frac{\|A^{(k)}\|_2 \|B^{(k)}\|_2}{\sum_i \|A^{(k)}\|_2 \|B^{(k)}\|_2}$$

- minimizes $\mathbb{E}\|AB - CR\|_F$
- $\mathbb{E}\|AB - CR\|_F^2 = \frac{1}{m} \sum_{k=1}^{d} \frac{1}{p_k} \|A^{(k)}\|_2^2 \|B_{(k)}\|_2^2 - \frac{1}{m}\|AB\|_F^2$

$$= \frac{1}{m} \left(\sum_{k=1}^{d} \|A^{(k)}\|_2 \|B_{(k)}\|_2 \right)^2 - \frac{1}{m}\|AB\|_F^2$$

is the optimal error

Special case: computing $A^T A$

- Nonuniform sampling

$$p_k = \frac{\|A_{(k)}\|_2^2}{\sum_i \|A_{(k)}\|_2}$$

- minimizes $\mathbb{E}\|A^T A - CR\|_F$
  note that $C = R^T$

Probability Bounds

- So far we have results on the expectation of the error
- Markov's Inequality
- If Z is a non-negative random variable and $t > 0$, then

$$\mathbb{P}\left[Z > t\right] \leq \frac{\mathbb{E}Z}{t}$$

Probability Bounds for AMM

- Upper-bounding the error

$$
\begin{aligned}
\mathbb{E}\|AB - CR\|_F^2 &= \frac{1}{m}\left(\sum_{k=1}^{d}\|A^{(k)}\|_2\|B_{(k)}\|_2\right)^2 - \frac{1}{m}\|AB\|_F^2 \\
&\leq \frac{1}{m}\left(\sum_{k=1}^{d}\|A^{(k)}\|_2\|B_{(k)}\|_2\right)^2 \\
&\leq \frac{1}{m}\left(\sqrt{\sum_{k=1}^{d}\|A^{(k)}\|_2^2}\sqrt{\sum_{k=1}^{d}\|B_{(k)}\|_2^2}\right)^2 \\
&= \frac{1}{m}\|A\|_F^2\|B\|_F^2 \,.
\end{aligned}
$$

Applying Markov's inequality

- $\mathbb{P}\left[\|AB - CR\|_F^2 > \epsilon^2\|A\|_F^2\|B\|_F^2\right] \leq \frac{\mathbb{E}\|AB-CR\|_F^2}{\epsilon\|A\|_F^2\|B\|_F^2} \leq \frac{1}{m\,\epsilon^2}$

Final Probability Bound

- For any $\delta > 0$, set $m = \frac{1}{\delta \epsilon^2}$ to obtain

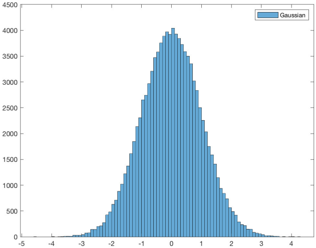
$$\mathbb{P}\left[\|AB - CR\|_F > \epsilon \|A\|_F \|B\|_F\right] \leq \delta \tag{1}$$

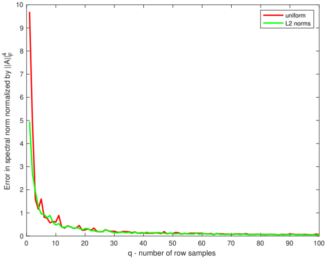
- i.e., $\|AB - CR\|_F < \epsilon \|A\|_F \|B\|_F$ with probability $1 - \delta$.

Numerical simulations for AMM

- Approximating $A^T A$

  rows of A are i.i.d. Gaussian

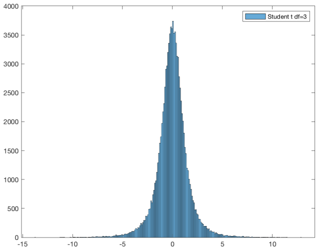


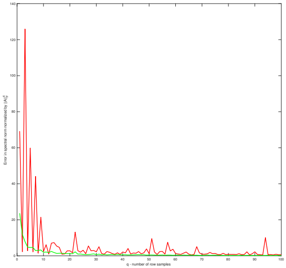


Numerical simulations for AMM

- Approximating $A^T A$

  rows of A are i.i.d. Student's t-distribution (3 degrees of freedom)

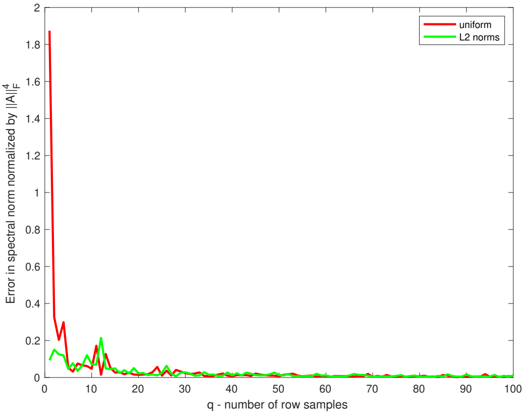




Numerical simulations for AMM

- Approximating $A^T A$

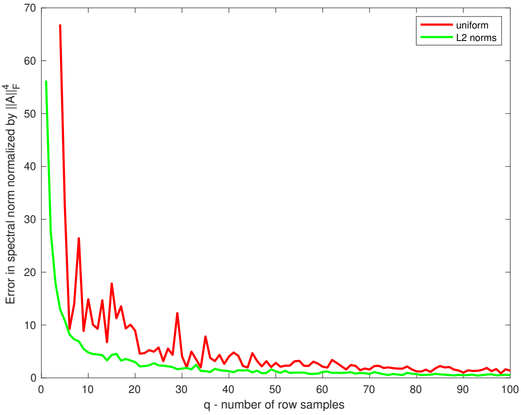
  a subset of the CIFAR dataset



Numerical simulations for AMM

- Approximating $A^T A$

  sparse matrix from a computational fluid dynamics model



Questions?