Algorithms for Big Data (X)

Chihao Zhang

Shanghai Jiao Tong University

Nov. 22, 2019

Matrix Multiplication

Matrix Multiplication

Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, we computes $C = AB$.

Matrix Multiplication

Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, we computes $C = AB$.

For $m = n = p$, the naive algorithm costs $O(n^3)$ multiplication operations.

Matrix Multiplication

Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$, we computes $C = AB$.

For $m = n = p$, the naive algorithm costs $O(n^3)$ multiplication operations.

The Strassen's algorithm reduces the cost to $O(n^{2.81})$.

Matrix Multiplication

Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, we computes $C = AB$.

For $m = n = p$, the naive algorithm costs $O(n^3)$ multiplication operations.

The Strassen's algorithm reduces the cost to $O(n^{2.81})$.

The best algorithm so far costs $O(n^\omega)$ where $\omega < 2.3728639$.

Matrix Multiplication

Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, we computes $C = AB$.

For $m = n = p$, the naive algorithm costs $O(n^3)$ multiplication operations.

The Strassen's algorithm reduces the cost to $O(n^{2.81})$.

The best algorithm so far costs $O(n^{\omega})$ where $\omega < 2.3728639$.

Today we will introduce a Monte-Carlo algorithm to approximate AB.

Review of Linear Algebra

Review of Linear Algebra

Assume $A = \begin{bmatrix}\mathbf{a}_1, \dots, \mathbf{a}_n\end{bmatrix}$ and $B = \begin{bmatrix}\mathbf{b}_1^\top \\ \vdots \\ \mathbf{b}_n^\top\end{bmatrix}$.

Review of Linear Algebra

Assume $A = [\mathbf{a}_1, \dots, \mathbf{a}_n]$ and $B = \begin{bmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_n^T \end{bmatrix}$.

Then $AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T$, where each $\mathbf{a}_i \mathbf{b}_i^T$ is of rank 1.

Review of Linear Algebra

Assume $A = [\mathbf{a}_1, \ldots, \mathbf{a}_n]$ and $B = \begin{bmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_n^T \end{bmatrix}$.

Then $AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T$, where each $\mathbf{a}_i \mathbf{b}_i^T$ is of rank 1.

The Frobenius norm of a matrix $A = (a_{ij})_{1 \le i \le m, 1 \le j \le n}$ is

$$\|A\|_F \triangleq \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}.$$

The Algorithm

The Algorithm

Note that $AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^{\mathsf{T}}$.

The Algorithm

Note that $AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T$.

The algorithm randomly pick indices $i \in [n]$ independently c times (with replacement).

The Algorithm

Note that $AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T$.

The algorithm randomly pick indices $i \in [n]$ independently c times (with replacement).

Let $J : [c] \to [n]$ denote the indices.

The Algorithm

Note that $AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i^T$.

The algorithm randomly pick indices $i \in [n]$ independently c times (with replacement).

Let $J : [c] \to [n]$ denote the indices.

Output $\sum_{i=1}^{c} w(J(i)) \cdot \mathbf{a}_{J(i)} \mathbf{b}_{J(i)}^T$, where $w(J(i))$ is some weight to be determined.

We fix a distribution on $[n]$ (p_i for $i \in [n]$ satisfying $\sum_{i\in[n]} p_i = 1$).

We fix a distribution on $[n]$ (p_i for $i \in [n]$ satisfying $\sum_{i\in[n]} p_i = 1$).

Therefore, the index j is picked $c \cdot p_j$ times in expectation, so we can set $w(j) = (cp_j)^{-1}$.

We fix a distribution on $[n]$ (p_i for $i \in [n]$ satisfying $\sum_{i\in[n]} p_i = 1$).

Therefore, the index j is picked $c \cdot p_j$ times in expectation, so we can set $w(j) = (cp_j)^{-1}$.

It is convenient to formulate the algorithm using matrices. Define a random sampling matrix $\Pi = (\pi_{ij}) \in \mathbb{R}^{c\times c}$ such that

$$\pi_{ij} = \begin{cases} (cp_i)^{-\frac{1}{2}} & \text{if } i = J(j) \\ 0 & \text{otherwise} \end{cases}$$

.

We fix a distribution on $[n]$ (p_i for $i \in [n]$ satisfying $\sum_{i\in[n]} p_i = 1$).

Therefore, the index j is picked $c \cdot p_j$ times in expectation, so we can set $w(j) = (cp_j)^{-1}$.

It is convenient to formulate the algorithm using matrices. Define a random sampling matrix $\Pi = (\pi_{ij}) \in \mathbb{R}^{c\times c}$ such that

$$\pi_{ij} = \begin{cases} (cp_i)^{-\frac{1}{2}} & \text{if } i = J(j) \\ 0 & \text{otherwise} \end{cases}$$

.

Then our algorithm outputs $A'B'$ where

$$A' = A\Pi \text{ and } B' = \Pi^T B.$$

Analysis

Analysis

We are going to choose some $(p_i)_{i \in [n]}$ so that $A'B' \approx AB$.

Analysis

We are going to choose some $(p_i)_{i\in[n]}$ so that $A'B' \approx AB$.

Fix i, j for any $k \in [c]$, we let $X_k = \left(\frac{\mathbf{a}_{J(k)}\mathbf{b}^T_{J(k)}}{cp_{J(k)}}\right)_{ij}$.

Analysis

We are going to choose some $(p_i)_{i\in[n]}$ so that $A'B' \approx AB$.

Fix i, j for any $k \in [c]$, we let $X_k = \left(\frac{\mathbf{a}_{J(k)}\mathbf{b}^T_{J(k)}}{cp_{J(k)}}\right)_{ij}$.

$$\mathbf{E}[X_k] = \sum_{\ell=1}^{n} p_\ell \left(\frac{\mathbf{a}_\ell \mathbf{b}_\ell^T}{cp_\ell}\right)_{ij} = \frac{1}{c}(AB)_{ij}$$

$$\mathbf{E}\left[X_k^2\right] = \sum_{\ell=1}^{n} p_\ell \left(\frac{\mathbf{a}_\ell \mathbf{b}_\ell^T}{cp_\ell}\right)_{ij}^2 = \sum_{\ell=1}^{n} \frac{a_{\ell i}^2 b_{\ell j}^2}{c^2 p_\ell}$$

$$\mathbf{Var}[X_k] = \sum_{\ell=1}^{n} \frac{a_{\ell i}^2 b_{\ell j}^2}{c^2 p_\ell} - \frac{1}{c^2}(AB)_{ij}^2 .$$

Therefore,

$$\mathbf{E}\left[(A'B')_{ij}\right] = \sum_{k=1}^{c} \mathbf{E}\left[X_k\right] = (AB)_{ij}.$$

Therefore,

$$\mathbf{E}\left[(A'B')_{ij}\right] = \sum_{k=1}^{c} \mathbf{E}\left[X_k\right] = (AB)_{ij}.$$

We are going to study the concentration of this algorithm.

Therefore,

$$\mathbf{E}\left[(A'B')_{ij}\right] = \sum_{k=1}^{c} \mathbf{E}\left[X_k\right] = (AB)_{ij}.$$

We are going to study the concentration of this algorithm.

We compute that

$$\begin{aligned}
\mathbf{E}\left[\|AB - A'B'\|_F^2\right] &= \sum_{i=1}^{n} \sum_{j=1}^{p} \mathbf{E}\left[(AB - A'B')_{ij}^2\right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{p} \mathbf{Var}\left[(A'B')_{ij}\right] \\
&= \frac{1}{c}\left(\sum_{\ell=1}^{n} \frac{1}{p_\ell}\|\mathbf{a}_\ell\|^2\|\mathbf{b}_\ell\|^2 - \|AB\|_F^2\right)
\end{aligned}$$

If we choose $p_\ell \sim \|a_\ell\|\|b_\ell\|$, then

$$\mathbf{E}\left[\|AB - A'B'\|_F^2\right] = \frac{1}{c}\left(\left(\sum_{\ell=1}^{n} \|\mathbf{a}_\ell\|\|\mathbf{b}_\ell\|\right)^2 - \|AB\|_F^2\right)$$
$$\leq \frac{1}{c}\left(\sum_{\ell=1}^{n} \|\mathbf{a}_\ell\|\|\mathbf{b}_\ell\|\right)^2$$
$$\leq \frac{1}{c}\|A\|_F^2\|B\|_F^2.$$

Therefore, by Chebyshev's inequality,

$$\mathbf{Pr}\left[\|AB - A'B'\|_F > \varepsilon\|A\|_F\|B\|_F\right] = \mathbf{Pr}\left[\|AB - A'B'\|_F^2 > \varepsilon^2\|A\|_F^2\|B\|_F^2\right] \le \frac{1}{c\varepsilon^2}.$$

Therefore, by Chebyshev's inequality,

$$\mathbf{Pr}\left[\|AB - A'B'\|_F > \varepsilon\|A\|_F\|B\|_F\right] = \mathbf{Pr}\left[\|AB - A'B'\|_F^2 > \varepsilon^2\|A\|_F^2\|B\|_F^2\right] \leq \frac{1}{c\varepsilon^2}.$$

We can use a variant of median trick to boost the algorithm.

Therefore, by Chebyshev's inequality,

$$\mathbf{Pr}\left[\|AB - A'B'\|_F > \varepsilon \|A\|_F \|B\|_F\right] = \mathbf{Pr}\left[\|AB - A'B'\|_F^2 > \varepsilon^2 \|A\|_F^2 \|B\|_F^2\right] \leq \frac{1}{c\varepsilon^2}.$$

We can use a variant of median trick to boost the algorithm.

We can choose $c = O(\frac{1}{\varepsilon^2} \log\left(\frac{1}{\delta}\right))$ to achieve $1 - \delta$ probability of correctness.

Graph Spectrum