THE UNIVERSITY OF TEXAS AT AUSTIN
Department of Statistics and Data Sciences
College of Natural Sciences

SDS 385: Stat Models for Big Data

Lecture 9: Sampling methods

Purnamrita Sarkar
Department of Statistics and Data Science
The University of Texas at Austin

https://psarkar.github.io/teaching

Sampling for matrix multiplication

- Goal: multiply two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$
- Used as an inner routine in many algorithms.
- Lets first try the vanilla algorithm for warmup.

The naive algorithm

Algorithm 1 Vanilla three-look matrix multiplication algorithm.

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

Output: The product AB

```
for i = 1 to m do
    for j = 1 to p do
        (AB)_ij = 0
        for k = 1 to n do
            (AB)_ik += A_ij B_jk
        end for
    end for
end for
Return AB
```

The naive algorithm - complexity

- Step 1 has m loops
- Steo 2 has n loops
- Steo 3 has p loops
- Total mnp computation.
- When $m = n = p$ it takes $O(n^3)$ time. Too much!

Alternatives

- Strassen algorithm (1969) takes $O(n^{2.81})$ time
- Coppersmith-Winograd algorithm (2010) takes $O(n^{2.375})$ time
  - Often used as a building block in other algorithms to prove theoretical time bounds.
  - However, unlike the Strassen algorithm, it is not used in practice because it only provides an advantage for matrices so large that they cannot be processed by modern hardware
- Stother's algorithm (2011) $O(n^{2.374})$
- Vassilevska William's algorithm (2011) $O(n^{2.3728642})$
  - Le Galls improvement (2014) $O(n^{2.3728639})$

Another much simpler approach–Sample

- Notation:
  - Let $A^i \in \mathbb{R}^m$ is the i^{th} column of A
  - $B_i \in R^p$ is the i^{th} row of B.
- Note that
$$(AB)_{ik} = \sum_j A_{ij} B_{jk} = A_i^T B^j$$
- Instead, we will consider a matrix multiplication as the sum of outer products, i.e.
$$AB = \sum_{i=1}^{n} A^i B_i^T$$
- Why?

Take one

- Forget about matrices, say you have n real numbers and you want to calculate the sum.
- If you sample k of these uniformly at random, you can approximate the sum by:

$$\sum_i x_i \approx \sum_{j=1}^{k} (n/k) x_{j*}$$

- $P(j^* = i) = 1/n$ for $i \in \{1, \ldots, n\}$

Take one

- This works because:

$$E[\sum_{j=1}^{k}(n/k)x_{j*}] = \sum_{j=1}^{k}\frac{n}{k}\sum_{i=1}^{n}\frac{1}{n}x_i = \sum_{i=1}^{n}x_i$$

- Variance is:

$$\mathrm{var}(\sum_{j=1}^{k}(n/k)x_{j*}) = \frac{n^2}{k}\underbrace{\left(\sum_i x_i^2/n - \bar{x}^2\right)}_{\text{variance of the numbers}}$$

- Note that the variance of this approximation increases if the numbers are very different from each other, i.e. the variance of the numbers you are summing is large
- Solution – weighted sampling.

Take two

- Sample an element j with probability p_j
- p_j sums to one, but not necessarily uniform.
- So lets try:

$$\sum_i x_i \approx \frac{1}{k}\sum_{j=1}^{k} x_{j*}/p_{j*}$$

- The expectation is:

$$E[\frac{1}{k}\sum_{j=1}^{k}\frac{x_{j*}}{p_{j*}}] = \frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{n} p_i\frac{x_i}{p_i} = \sum_{i=1}^{n} x_i$$

Take two

- The variance is:
$$\frac{1}{k}\left(\sum_i x_i^2/p_i - n^2\bar{x}^2\right)$$
- Now if you choose $p_i = x_i^2/\|x\|_2^2$
- $$\frac{1}{k}\left(\sum_i x_i^2/p_i - n^2\bar{x}^2\right) = \frac{1}{k}\left(\sum_i x_i^2/p_i - n^2\bar{x}^2\right)$$
- How do you minimize w.r.t p_i such that $\sum_i p_i = 1$?

Take two

- Lagrange multipliers!

$$L(\pi, \lambda) = \frac{1}{k}\left(\sum_i x_i^2/p_i - n^2\bar{x}^2\right) + \lambda(\sum_i p_i - 1)$$

- Differentiating w.r.t p_i and setting to zero gives:

$$\frac{x_i^2}{p_i^2} - \lambda = 0$$

$$p_i = \frac{|x_i|}{\sum_j |x_j|}$$

- The second line uses the fact that $\sum_i p_i = 1$ to solve for the λ

Take two

- So if all $x_i > 0$ then this choice of p_i gives variance 0!
- Can you explain this?
- But lets not stray from matrix multiplication

Matrix multiplication

Algorithm 2 The `BasicMatrixMultiplication` algorithm.

Input: An $m \times n$ matrix A, an $n \times p$ matrix B, a positive integer c, and probabilities $\{p_i\}_{i=1}^n$.

Output: Matrices C and R such that $CR \approx AB$

for $t = 1$ to c **do**

  Pick $i_t \in \{1, \ldots, n\}$ with probability $\mathbf{Pr}[i_t = k] = p_k$, in i.i.d. trials, with replacement

  Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$ and $R_{(t)} = B_{(i_t)}/\sqrt{cp_{i_t}}$.

end for

Return C and R.

Analogy to the sum

- For the sum we wanted to compute $\sum_i x_i$
- Here we want to compute $\sum_i A^i B_i^T$
- So the analog of x_i is the rank one outer product matrix $A^i B_i^T$
- Note that $\|A^i B_i^T\|_2^2 = \|A^i\|_2^2 \|B_i^T\|_2^2$
- Use $p_i := \frac{\|A^i\|_2 \|B_i\|_2}{\sum_i \|A^i\|_2 \|B_i\|_2}$

Initial results

- $E[(CR)_{ij}] = (AB)_{ij}$
- $\mathrm{var}((CR)_{ij}) = \frac{1}{c}\left(\sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{p_k} - (AB)_{ij}^2\right)$
- Now the x_k's are the same as $A_{ik}B_{kj}$. Remember our formula for mean?
  - The mean of $\frac{1}{c}\sum_{j=1}^{c} \frac{x_{j^*}}{p_{j^*}}$ was $\sum_{k=1}^{n} x_k$
  - Plug in $x_k = A_{ik}B_{kj}$ to get the mean as $\sum_k A_{ik}B_{kj}$
- How about variance?
  - Variance was: $\frac{1}{c}\left(\sum_k \frac{x_k^2}{p_k} - (\sum_k x_k)^2\right)$
  - Variance of $(CR)_{ij}$ is $\frac{1}{c}\left(\sum_k \frac{x_k^2}{p_k} - (\sum_k x_k)^2\right)$
  - Plug in $x_k = A_{ik}B_{kj}$ to get the right expression.

Error bounds

- $E[\|AB - CR\|_F^2] = \frac{1}{c}\left(\sum_{k=1}^{n} \frac{\|A^k\|^2 \|B_k\|^2}{p_k} - \|AB\|_F^2\right)$
- Why?

$$
\begin{aligned}
E\left[\|AB - CR\|_F^2\right] &= \sum_{ij} E[(CR)_{ij} - (AB)_{ij}]^2 \\
&= \sum_{ij} \mathrm{var}((CR)_{ij}) \\
&= \frac{1}{c} \sum_{ij} \left(\sum_{k=1}^{n} \frac{A_{ik}^2 B_{kj}^2}{p_k} - (AB)_{ij}^2\right)
\end{aligned}
$$

- Exchange the two sums and get the answer.

Optimal weights

- The optimal $p_k = \frac{\|A^k\|\|B_k\|}{\sum_k \|A^k\|\|B_k\|}$
- Same Lagrange multiplier trick to see $p_k = |x_k| / \sum_i |x_i|$
- Plug in $x_k = \|A^k\|\|B_k\|$
- Plug in to get

$$E[\|CR - AB\|_F^2] = \frac{1}{c}\left(\left(\sum_{k=1}^{n} \|A^k\|\|B_k\|\right)^2 - \|AB\|_F^2\right)$$

Computation time

- Calculate p_k in $O(n(m+p))$ time.
- Computational saving from $mnp \to \max(mn, np)$
- What if I did a sub-optimal weighted sampling?
- Take $p_i = \|A_i\|_F^2 / \|A\|_F^2$
- Now the error becomes $\frac{1}{c}(\|A\|_F^2 \|B\|_F^2 - \|AB\|_F^2)$

Nearly linear time SVD

- Input: $A \in \mathbb{R}^{m \times n}$ and $k > 0$
- Goal: Find an orthogonal matrix H_k such that $\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + \text{small}$
- A_k is the rank k approximation of A.

Nearly linear time SVD

LinearTimeSVD Algorithm

Input: $A \in \mathbb{R}^{m\times n}$, $c, k \in \mathbb{Z}^+$ s.t. $1 \le k \le c \le n$, $\{p_i\}_{i=1}^n$ s.t. $p_i \ge 0$ and $\sum_{i=1}^n p_i = 1$.

Output: $H_k \in \mathbb{R}^{m\times k}$ and $\sigma_t(C), t = 1, \ldots, k$.

- For $t = 1$ to c,
  - Pick $i_t \in 1, \ldots, n$ with $\mathbf{Pr}[i_t = \alpha] = p_\alpha$, $\alpha = 1, \ldots, n$.
  - Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$.
- Compute C^TC and its singular value decomposition; say $C^TC = \sum_{t=1}^c \sigma_t^2(C) y^t {y^t}^T$.
- Compute $h^t = Cy^t/\sigma_t(C)$ for $t = 1, \ldots, k$.
- Return H_k, where $H_k^{(t)} = h^t$, and $\sigma_t(C), t = 1, \ldots, k$.

Constant time SVD

ConstantTimeSVD Algorithm

Input: $A \in \mathbb{R}^{m\times n}$, $c, w, k \in \mathbb{Z}^+$ s.t. $1 \le w \le m$, $1 \le c \le n$, and $1 \le k \le \min(w,c)$, and $\{p_i\}_{i=1}^n$ s.t. $p_i \ge 0$ and $\sum_{i=1}^n p_i = 1$.

Output: $\sigma_t(W), t = 1,\ldots,\ell$ and a "description" of $\tilde{H}_\ell \in \mathbb{R}^{m\times\ell}$.

- For $t = 1$ to c,
  - Pick $i_t \in 1,\ldots,n$ with $\mathbf{Pr}[i_t = \alpha] = p_\alpha$, $\alpha = 1,\ldots,n$ and save $\{(i_t, p_{j_t}) : t = 1,\ldots,c\}$.
  - Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$. (Note that C is not explicitly constructed in RAM.)
- Choose $\{q_j\}_{j=1}^m$ s.t. $q_j = \left|C_{(j)}\right|^2 / \|C\|_F^2$.
- For $t = 1$ to w,
  - Pick $j_t \in 1,\ldots,m$ with $\mathbf{Pr}[j_t = \alpha] = q_\alpha$, $\alpha = 1,\ldots,m$.
  - Set $W_{(t)} = C_{(j_t)}/\sqrt{wq_{j_t}}$.
- Compute W^TW and its singular value decomposition. Say $W^TW = \sum_{t=1}^c \sigma_t^2(W) z^t z^{tT}$.
- If a $\|\cdot\|_F$ bound is desired, set $\gamma = \epsilon/100k$,

  Else if a $\|\cdot\|_2$ bound is desired, set $\gamma = \epsilon/100$.
- Let $\ell = \min\{k, \max\{t : \sigma_t^2(W) \ge \gamma \|W\|_F^2\}\}$.
- Return singular values $\{\sigma_t(W)\}_{t=1}^\ell$ and their corresponding singular vectors $\left\{z^t\right\}_{t=1}^\ell$.

Acknowledgment

Drineas, Kannan and Mahoney's paper on matrix multiplication.

Acknowledgment

Ullman's lecture notes from "Mining of Massive Datasets"