2 Column Sampling Algorithms

Column (Row) sampling techniques select a subset of *"important"* columns (or rows) of the original matrix, often randomly (sometimes deterministically) with respect to a well-defined probability distribution; and show that sampled matrix is a good approximation to original one.

Column sampling methods vary in the way they define notion of *importance*. Often importance of an item is the weight associated to it, e.g. file records have associated weights as size of the file, and IP addresses have weights as number of times the IP address makes a request. In order to give an intuition to why it is necessary to sample important items, consider a set of weighted items $S = \{(a_1, w_1), (a_2, w_2), \cdots, (a_n, w_n)\}$ that we want to summarize with a small yet representative sample. Let's define a representative sample as the one estimates total weight of S (i.e. $W_s = \sum_{i=1}^{n} w_i$), in expectation. Clearly, this is achievable with a sample set of size one also: we sample any item $(a_j, w_j) \in S$ with an arbitrary fixed probability p, and rescales it to have weight W_s/p. This way sample set has total weight W_s in expectation, but has a large variance too. To lower down the variance, it is necessary to allow heavy items (i.e. important items) to get sampled with higher probability. This generic description is called "Importance Sampling", and is described in algorithm 2.0.1. Importance Sampling is a meta algorithm that once instantiated with different settings of $\{p_i\}_{i=1}^{n}$ and c adapts to many column sampling algorithms.

Algorithm 2.0.1 Importance Sampling

```
1: Input: $A \in \mathbb{R}^{d \times n}, 1 \le c \le n$
2: Output: $B \in \mathbb{R}^{d \times c}$
3: $B \leftarrow$ all zeros matrix $\in \mathbb{R}^{d \times c}$
4: for $i \in [n]$ do
5:     Compute probability $p_i$ for row $A_{:,i}$
6:     for $j \in [c]$ do
7:         Insert (and rescale) $A_{:,i}$ into $B_{:,j}$ with probability $p_i$
8: return $B$
```

Note most column sampling algorithms sample columns with replacement, i.e. every column of A can get sampled more than one time. Below we describe two most common column sampling algorithms which sample with different notion of importance, one achieves an additive error bound and the other achieves the better relative error bound.

2.1 LinearTime SVD

Drineas *et al.* [2] proposed "LinearTime SVD" (LTSVD) which samples columns proportional to their squared norm, i.e. $p_i = \|A_{:,i}\|^2/|A\|_F^2$; LTSVD is described in algorithm 2.1.1. The strategy behind this algorithm is to pick c columns of A with replacement, rescale each by factor $1/\sqrt{c\, p_i}$ to form matrix $B \in \mathbb{R}^{d \times c}$, then compute left singular vectors and corresponding singular values of B which will be an approximations to left singular vectors of A.

Algorithm 2.1.1 Linear Time SVD

```
1: Input: $A \in \mathbb{R}^{d\times n}, 1 \le c \le n, 1 \le k \le \min(n,d)$
2: $B \leftarrow$ all zeros matrix $\in \mathbb{R}^{d\times c}$
3: for $i \in [n]$ do
4:     Compute probability $p_i = |A_{:,i}|^2/\|A\|_F^2$ for column $A_{:,i}$
5:     for $j \in [c]$ do
6:         Insert $A_{:,i}$ into $B_{:,j}$ with probability $p_i$
7:         Rescale $B_{:,j}$ by $1/\sqrt{c\,p_i}$
8: Compute $B^TB$ and its SVD, $B^TB = Y\Sigma^2Y^T$
9: Compute $H = BY^T\Sigma^{-1}$
10: return $H_k$ (first $k$ columns of $H$)
```

At first glance, it might seem LTSVD needs two passes over data: one for computing sampling probabilities (or in other words $\|A\|_F$) and one for sampling actual columns, however in[1], authors showed sampling proportional to $|A_{(:,i)}|^2/\sum_{j=1}^{i}|A_{(:,j)}|^2$ is equivalent to sampling from probability distribution $p_i = |A_{:,i}|^2/\|A\|_F^2$; therefore LTSVD needs only one pass over data to sample columns. For more details, look at "SELECT" algorithm in [1].

Authors proved if $c \ge 4k\eta^2/\beta\varepsilon^2$ where $\eta = 1 + \sqrt{(8/\beta)\log(1/\delta)}$ and $\beta \le 1$ is a positive constant and $\delta \in (0,1)$, then LTSVD achieves Frobenius error bound

$$\|A - H_kH_k^TA\|_F^2 \le \|A - A_k\|_F^2 + \varepsilon\|A\|_F^2$$

and if $c \ge 4\eta^2/\beta\varepsilon^2$ it achieves spectral error bound

$$\|A - H_kH_k^TA\|_2^2 \le \|A - A_k\|_2^2 + \varepsilon\|A\|_F^2$$

with probability $1-\delta$. Note $H_kH_k^TA$ is equivalent to $\pi_{B_k}(A)$ which is the projection of A onto best rank k of B.

2.1.1 Error Analysis

Here, we demonstrate the proof for Frobenius error bound only. Let svd decomposition of A and B be $A = USV^T$ and $B = H\Sigma Y^T$, respectively. Note that column space of A and B are captured by U and H, respectively. We decompose left-hand side of the Frobenius error bound as following

$$\begin{aligned}
\|A - H_kH_k^TA\|_F^2 &= \mathsf{Tr}\left((A - H_kH_k^TA)^T(A - H_kH_k^TA)\right) && \text{due to } \|X\|_F^2 = \mathsf{Tr}(X^TX)\\
&= \mathsf{Tr}\left(A^TA - A^TH_kH_k^TA - (H_kH_k^TA)^TA + (H_kH_k^TA)^TH_kH_k^TA\right)\\
&= \mathsf{Tr}\left(A^TA - 2A^TH_kH_k^TA + A^TH_kH_k^TH_kH_k^TA\right)\\
&= \mathsf{Tr}\left(A^TA - 2A^TH_kH_k^TA + A^TH_kH_k^TA\right) && \text{due to } H_k^TH_k = I_k\\
&= \mathsf{Tr}\left(A^TA - A^TH_kH_k^TA\right)\\
&= \mathsf{Tr}\left(A^TA\right) - \mathsf{Tr}\left(A^TH_kH_k^TA\right) && \text{due to linearity of trace}\\
&= \|A\|_F^2 - \|A^TH_k\|_F^2 && \text{due to } \|X\|_F^2 = \mathsf{Tr}(X^TX)
\end{aligned}$$

In order to bound $\|A^T H_k\|_F^2$, we can write:

$$\begin{aligned}\left|\|A^T H_k\|_F^2 - \|B^T H_k\|_F^2\right| &\le \sqrt{k}\left(\sum_{t=1}^{k}\left(\|A^T H_{:,t}\|^2 - \|B^T H_{:,t}\|^2\right)^2\right)^{1/2} \quad \text{using Cauchy-Schwartz inequality}\\ &= \sqrt{k}\left(\sum_{t=1}^{k}\left(H_{:,t}^T(AA^T - BB^T)H_{:,t}\right)^2\right)^{1/2}\\ &\le \sqrt{k}\|AA^T - BB^T\|_F\end{aligned}$$

Note Cauchy-Schwartz inequality states that $\left(\sum_{i=1}^{k}(x_i - y_i)\right)^2 \le k\sum_{i=1}^{k}(x_i - y_i)^2$.

Now in order to bound $\|B^T H_k\|_F^2$ we write

$$\begin{aligned}\left|\|B^T H_k\|_F^2 - \|A^T U_k\|_F^2\right| &\le \sqrt{k}\left(\sum_{t=1}^{k}\left(\Sigma_{t,t}^2 - S_{t,t}^2\right)^2\right)^{1/2} \qquad \text{due to } \|X\|_F^2 = \mathsf{Tr}(X^T X)\\ &\le \sqrt{k}\|BB^T - AA^T\|_F\end{aligned}$$

Where again first transition follows by Cauchy Schwartz inequality. Using triangle inequality on above two bounds, we achieve:

$$\begin{aligned}\left|\|A^T H_k\|_F^2 - \|A^T U_k\|_F^2\right| &\le \left|\|A^T H_k\|_F^2 - \|B^T H_k\|_F^2\right| + \left|\|B^T H_k\|_F^2 - \|A^T U_k\|_F^2\right|\\ &\le 2\sqrt{k}\|AA^T - BB^T\|_F\end{aligned}$$

Therefore we can simplify final error bound as

$$\|A - H_k H_k^T A\|_F^2 = \|A\|_F^2 - \|H_k H_k^T A\|_F^2 \le \|A - A_k\|_F^2 + 2\sqrt{k}\|AA^T - BB^T\|_F$$

The only thing to left is to bound $\|AA^T - BB^T\|_F$, for that we use the following theorem from [1].

Theorem 2.1.1. *Let $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{d\times r}$ and $c \in \mathbb{Z}^+$ such that $1 \le c \le n$ and $\{p_i\}_{i=1}^n$ be probability distribution over columns of A and rows of B such that $p_i \ge \frac{\beta\|A_{:,i}\|\|B_{i,:}\|}{\sum_{j=1}^n\|A_{:,j}\|\|B_{j,:}\|}$ for some positive constant $\beta \le 1$. If matrix $C \in \mathbb{R}^{d\times c}$ is constructed by sampling columns of A according to $\{p_i\}_{i=1}^n$ and matrix $D \in \mathbb{R}^{c\times r}$ is constructed by picking same rows of B, then with probability atleast $1-\delta$*

$$\|AB - CD\|_F^2 \le \frac{\mu^2}{\beta c}\|A\|_F^2\|B\|_F^2$$

where $\delta \in (0,1)$, $\mu = 1 + \sqrt{(8/\beta)\log(1/\delta)}$.

Using theorem 2.1.1, we bound $\|AA^T - BB^T\|_F^2 \le \frac{\mu^2}{\beta c}\|A\|_F^2\|A^T\|_F^2 = \frac{\mu^2}{\beta c}\|A\|_F^4$, putting all these together we get the final Fronebius error bound as

$$\|A - H_k H_k^T A\|_F^2 \le \|A - A_k\|_F^2 + \sqrt{\frac{4k\mu^2}{\beta c}}\|A\|_F^2$$

Setting $c = O(k/\varepsilon^2)$ gives an ε-related bound as claimed by the algorithm.

2.1.2 Space and Run Time Analysis

For constructing the sample matrix B, we need to read the data which takes $O(nnz(A))$ and sample columns with replacement which takes $O(cn)$. An additional $O(c^2d+c^3)$ time is needed to construct B^TB and takes its svd. Thus, the total running time of LinearTimeSVD is $O(cn+c^2d+c^3+nnz(A))$. The space usage of the algorithm is $O(cd+c)=O(cd)$, $O(cd)$ for the sketch B and $O(c)$ for maintaining the sampling probabilities.

2.2 Leverage Score Sampling

In another line of work, people have tried to sample columns according to their leverage (or importance) scores. Intuitively, leverage scores are statistics about matrix A that determine which columns (or rows) are most representative with respect to a rank-k subspace of A. The formal definition of it is as follows:

Leverage Scores: Let $V_k \in \mathbb{R}^{d\times k}$ contain the top k right singular vectors of matrix $A \in \mathbb{R}^{n\times d}$. Then the rank-$k$ leverage scores of the i-th column of A are defined as $\ell_i^{(k)} = \|[V_k]_{i,:}\|^2$.

The history of leverage score sampling dates back to a work by Joliffe[4], in which he proposed a deterministic approach for picking columns of A having largest leverage scores. In [3], Drineas *et al.* extended Joliffe's algorithm as to a randomized method, which allowed them to sample columns of A with a probability proportional to leverage scores; this method is described in algorithm 2.2.1 and it's often called *Subspace Sampling* (SSS) as it samples columns with the probability proportional to their contribution to the best rank k subspace of the matrix.

Algorithm 2.2.1 Randomized Leverage Score Sampling (SSS)

```
1: Input: $A \in \mathbb{R}^{d\times n}, 1 \le c \le n, 1 \le k \le \min(n,d)$
2: $B \leftarrow$ all zeros matrix $\in \mathbb{R}^{d\times c}$
3: Compute svd$(A)$ as $A = USV^T$
4: for $i \in [n]$ do
5:     Compute probability $p_i = |V_{k_{(i,:)}}|^2/k$, for each column $A_{:,i}$
6:     for $j \in [c]$ do
7:         Insert $A_{:,i}$ into $B_{:,j}$ by sampling with replacement
8:         Each column is rescaled by $1/\sqrt{cp_i}$
9: return $B$
```

This method samples $c = O(k\log k/\varepsilon^2)$ columns and achieve a $(1+\varepsilon)$-relative error bound as following

$$\|A - BB^\dagger A\|_\zeta^2 \le (1+\varepsilon)\|A - A_k\|_\zeta^2$$

Where $\zeta = \{2, F\}$ and $\varepsilon \in (0, 1/2)$. Note $BB^\dagger A$ is equivalent to $\pi_B(A)$, i.e. projection of A onto the space spanned by B. Unlike LTSVD which projects onto a rank-k subspace, $BB^\dagger A$ projects onto B which might be of rank upto $k\log k/\varepsilon^2$.

Note that this algorithm needs to take the svd of A first to compute sampling probabilities, and that would need a pass over data, $O(nd^2)$ time and $O(nd)$ space to store the whole matrix. After that another pass is needed to do the actual column sampling phase. The total running time of the algorithm is $O(nd^2+cn)$ since sampling is done with replacement, and the total space usage of the algorithm is $O(nd+cd) = O(nd)$.

Table 2.2 summarizes run time, space usage and error bounds of above mentioned Column Sampling methods.

	# PASSES	RUN TIME	SPACE USAGE	ERROR BOUND
LTSVD[2]	1	$O(k/\varepsilon^2(n + kd/\varepsilon^2 + k^2/\varepsilon^4) + nnz(A))$	$O(kd/\varepsilon^2)$	$\|A - \pi_{B_k} A\|_F^2 \leq OPT_F + \varepsilon\|A\|_F^2$
		$O(1/\varepsilon^2(n + d/\varepsilon^2 + /\varepsilon^4) + nnz(A))$	$O(d/\varepsilon^2)$	$\|A - \pi_{B_k} A\|_2^2 \leq OPT_2 + \varepsilon\|A\|_F^2$
SSS[3]	2	$O(nd^2 + (nk \log k)/\varepsilon^2)$	$O(nd)$	$\|A - \pi_B A\|_{F,2}^2 \leq (1+\varepsilon)OPT_{F,2}$

Table 2.1: Comparing different Column Sampling algorithms. We define $OPT_2 = \|A - A_k\|_2$ and $OPT_F = \|A - A_k\|_F$. $B \in \mathbb{R}^{d \times c}$ is the matrix algorithms sample from A.

Bibliography

[1] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[2] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[3] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008.

[4] Ian T Jolliffe. Discarding variables in a principal component analysis. i: Artificial data. *Applied statistics*, pages 160–173, 1972.