MinHash Sketch: Techniques and Applications

Qiyu Liu

Hong Kong University of Science and Technology

qliuau@cse.ust.hk

June 4, 2018

Overview

Background

- Sketch of a database is a randomized data structure maintaining a summary of some interesting information (e.g., cardinality, similarity, aggregation, etc.)
- Sketch technique is powerful in **massive** data analytics.
- Operations and queries that are specified with respect to the explicit and often very large subsets, can be processed instead in **sketch space**.

Background – Cont.

A key properties that a sketch should have is *mergeable/composable* which means adding an element or taking the union of multiple subsets can be performed in sketch space.

For a dataset X, its sketch is denoted as $S(X)$. Here are two basic operations in sketch space:

1. **Inserting an element**: Given a set X and an element $y \in U$, $S(X \cup \{y\})$ can be calculated according to $S(X)$ and y.
2. **Merging two sets**: Given two sets X and Y, $S(X \cup Y)$ can be calculated according to $S(X)$ and $S(Y)$.

Supporting for insertion and merging make the sketch technique suitable for **stream** processing and **parallel or distributed** computing respectively.

Order Statistics

- **Definition**: Given a collection of samples $\{X_1, \cdots, X_n\}$, denote the sorted sequence (by ascending order) as $\{X_{(1)}, \cdots, X_{(n)}\}$. $X_{(k)}$ is called *kth order statistics*. Specifically, $X_{(1)}$ and $X_{(n)}$ are minima and maxima respectively.
- **Distribution of Order Statistics**: Suppose $\{X_1, \cdots, X_n\}$ are drawn from distribution $F(x)$, the CDF of *kth* order statistics $X_{(k)}$ is:

$$F_k(x) = \Pr(X_{(k)} \leq x) = \sum_{j=k}^{n} \binom{j}{n} F(x)^j (1 - F(x))^{n-j}, x \in \mathbb{R} \quad (1)$$

The corresponding density function is:

$$f_k(x) = \frac{dF}{dx} = \frac{n!}{(k-1)!(n-k)!} F(x)^{k-1} (1 - F(x))^{n-k} f(x), x \in \mathbb{R} \quad (2)$$

Order Statistics – Cont.

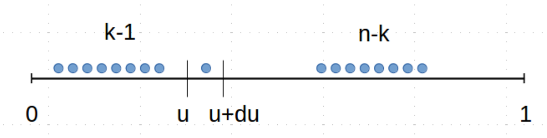
We study a special case where samples $X_1, \cdots, X_n$ are drawn from uniform distribution $U(0,1)$. The probability that $X_{(k)}$ falls into $[u, u+du]$ is:

$$p(u \leq X_{(k)} \leq u + du) =$$

$$\binom{k-1}{n} u^{k-1}(1-u)^{n-k+1} \binom{1}{n-k+1} \frac{du}{1-u} \left(\frac{1-u-du}{1-u}\right)^{n-k}$$

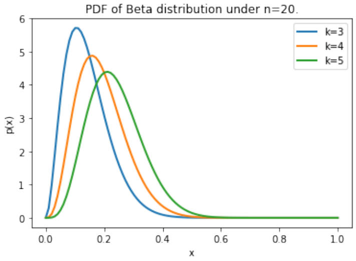
$$= \frac{n!}{(k-1)!(n-k)!} u^{k-1}(1-u)^{n-k} du + o(du)$$

which refers to a Beta distribution $Beta(k, n-k+1)$.



Order Statistics – Cont.

Illustration of $Beta(k, n-k+1)$:



Expectation of distribution $Beta(k, n-k+1)$ is $\frac{k}{n+1}$, which implies that if we observe $X_{(k)}$, an unbiased estimator of n is $\hat{n} = \frac{k}{X_{(k)}} - 1$ and this will be the foundation of MinHash technique.

MinHash Sketch Technique

We will focus on 3 MinHash sketch techniques:

- *k*-mins sketch [1, 2]
- *k*-partition sketch [1, 3, 4]
- bottom-*k* sketch [2, 5]

k-mins Sketch

k-mins Sketch includes the smallest rank in each of k independent rank assignments. There are k different rank functions $r_i(\cdot)$ and the sketch $S(X) = (\tau_1, \cdots, \tau_k)$ has $\tau_i = \min_{y \in X} r_i(y)$. When viewed as a sample, it corresponds to sampling k times with replacement.

k-mins Sketch – Cont.

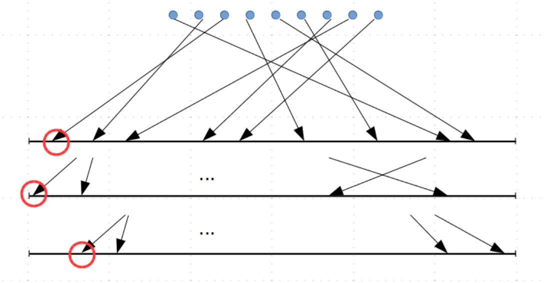


Figure: Illustration of *k*-mins Sketch.

k-mins Sketch – Cont.

Instead of directly using integral ranks, we always adopt *continuous ranking* where rank values uniformly fall into interval $[0, 1]$. This is because hash function can be used to map any type of data points to $[0, 1]$. Suppose that we use M-bits hash function $H : X \rightarrow 2^M - 1$, then $\frac{H(x)}{2^M-1}$ falls into $[0, 1]$ uniformly based on the assumption that $H(\cdot)$ is an ideal hash function.

By combining hash function, the k-mins sketch of X can be rewritten as $S(X) = (h_1, \cdots, h_k)$ where $h_i = \min_{y \in X} H_i(x)$ and $H_1(\cdot), \cdots, H_k(\cdot)$ are k different hash functions.

k-partition Sketch

k-partition Sketch uses a single rank assignment (instead of k) together with a uniform at random mapping of items to k buckets. We use $b : U \rightarrow [k]$ for the bucket mapping and r for the rank assignment. The sketch $S(X) = (\tau_1, \cdots, \tau_k)$ then includes the item with minimum rank in each bucket. That is $\tau_i = \min_{y \in X | b(y)=i} r(y)$. If the set is empty, the entry is the typically defined as the supremum of the domain of r.

k-partion Sketch – Cont.

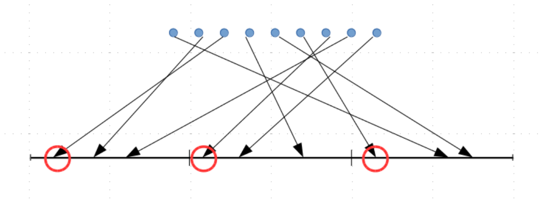


Figure: Illustration of *k*-partition Sketch.

Bottom-k Sketch

Bottom-k Sketch also only use one rank assignment. Instead of storing the minimum rank value, bottom-k sketch uses k smallest ranks, that is, $S(X) = (\tau_1, \cdots, \tau_k)$ where τ_i is i-th smallest rank value among X.

Bottom-k Sketch – Cont.

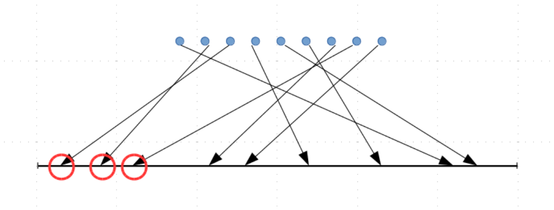


Figure: Illustration of bottom-*k* Sketch.

Update of Sketch: Insertion

The insertion operation is to get $S(X \cup \{y\})$ based on $S(X)$ and y.

- k-min Sketch: calculate $r_i(y)$ using hash function and then compare, coordinate-wise, each rank with the respective one in the sketch $S(X)$, taking the minimum of the two values. This process requires $O(k)$ time complexity.
- k-partition Sketch: first use $b(y)$ to determine which bucket y belonging to and then calculate $r(y)$. If the bucket $b(y)$ is empty, $\tau_i \leftarrow r(y)$; otherwise, $\tau_i \leftarrow \min\{\tau_i, r(y)\}$. This process only requires $O(1)$ time.
- Bottom-k Sketch: compute $r(y)$ and compare it with $\tau_k \in S$, if $r(y) \geq \tau_k$, $S(X \cup \{y\}) = S(X)$; otherwise, using binary-search to decide the position of $r(y)$. This process requires $O(1)$ time if $S(X)$ is not modified; otherwise, $O(\log k)$.

Update of Sketch: Insertion – Cont.

Imagine a data stream scenario where data point comes one by one and we maintain a k-min sketch S for the data points that we have seen until now, denoted by X. Suppose $N = |X|$. Then we have the following conclusion.

Proporsition: Update of S is bounded by $O(k \log N)$.

Proof. The probability that $S(X)$ needs to be updated when data point x_i arrives is:

$$Prob_i = \begin{cases} 1, & i \leq k \\ \frac{k}{i}, & i \geq k+1. \end{cases}$$

Thus, the expectation number of update for this k-mins sketch is:

$$E[\#update] = k + \sum_{i=k+1}^{N} \frac{k}{i} = O(k \log N).$$

Update of Sketch: Merge

The merge operation is to get $S(X \cup Y)$ based on $S(X)$ and $S(Y)$.

- k-min Sketch: $S(X \cup Y) = (\min\{\tau_1^{(X)}, \tau_1^{(Y)}\}, \cdots, \min\{\tau_k^{(X)}, \tau_k^{(Y)}\})$.
- k-partition Sketch: The same as k-min Sketch.
- Bottom-k Sketch: $S(X \cup Y) = topk(S(X) \cup S(Y))$.

All these merge methods of sketch take $O(k)$ time.

Size Estimation of Transitive Closure [FOCS '94]

Definition (Single Source Transitive Closure)

Consider directed graph $G = (V, E)$ where $n = |V|$ and $m = |E|$. The single source transitive closure of $u \in V$ is the set of nodes $v \in V$, such that u can reach v via a directed path between them.

Definition (All-pair Transitive Closure)

Consider directed graph $G = (V, E)$ where $n = |V|$ and $m = |E|$. The all-pair transitive closure is the set of all pairs of nodes $(u, v) \in V \times V$, such that u can reach v via a directed path between them.

According to the definition, the single source transitive closure can be calculated by any graph search algorithm (e.g., BFS) in linear time. As for the all pair transitive closure problem, the best exact result is $O(mn)$ (performing multiple BFS).

Size Estimation of Transitive Closure – Cont.

Problem Formulation: Given two sets X and Y and let $S : Y \to 2^X$ be a mapping. The objective is to estimate $\sum_{y \in Y} |S(y)|$.

References

1. Flajolet, Philippe, and G. Nigel Martin. "Probabilistic counting algorithms for database applications." Journal of computer and system sciences 31.2 (1985): 182-209.
2. Cohen, Edith. "Size-estimation framework with applications to transitive closure and reachability." Journal of Computer and System Sciences 55.3 (1997): 441-453.
3. Flajolet, Philippe, et al. "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm." AofA: Analysis of Algorithms. Discrete Mathematics and Theoretical Computer Science, 2007.
4. Li, Ping, Art Owen, and Cun-Hui Zhang. "One permutation hashing." Advances in Neural Information Processing Systems. 2012.
5. Broder, Andrei Z. "On the resemblance and containment of documents." Compression and Complexity of Sequences 1997. Proceedings. IEEE, 1997.

The End