15:853: Algorithms in the Real World
Fall 2019

Lecture 21

November 14 2019

Lecturer: Rashmi Vinayak *Scribe: Yifan Pu*

21.1 Recap From Last Class

Recall from last lecture that we were interested in finding the ϵ-heavy -hitters in a streaming model. We formulated an easier question, the count-min query, which asks for an estimate of each element with error at most $\epsilon||x^t||_1$, and created the basic count-min sketch method which can be found on the previous scribe note (using $h : U \mapsto [M]$, h from a universal hash function family).
Setting the estimate to be $y_i = A[h(i)]$, we can deduce (see previous scribe note) that $E[\text{error} = y_i - x_i^t] \leq \frac{||x^t||_1}{M}$. By setting $M = 1/\epsilon$, we have an expected error no more than $\epsilon||x^t||_1$.
However, this is an expectation, and we want to amplify the success probability that the error is within $\epsilon||x^t||_1$.

21.2 Amplification of the Success Probability

To amplify the success probability, we can carry out independent repetition of the same estimates, and combine results appropriately.
In this example, we choose l independent hash functions $h_i : U \mapsto [M]$ from universal hash function family H, then maintain l arrays A_i and then alter the previous count-min sketch.

Algorithm 1 Count-Min Sketch

```
while update a_t appears do
  for k = 1 to l do
    if a_t = add i then
      A_k[h_k(i)]++
    else {a_t = del i}
      A_k[h_k(i)]--
    end if
  end for
end while
```

Then, take our estimate $y_i = \min_{k=1,\ldots,l} A_k[h_k(i)]$.
By Markov inequality, and that $E[\text{error}] \leq \frac{||x^t||_1}{M}$, define $E_t = A_t[h_t(i)] - x_i^t$, where E_t is the

error of the tth estimator, then $P[E_t > \frac{2||x^t||_1}{M}] \leq \frac{1}{2}$.
Since the hash functions are chosen independently:

$$P[E_1 > \frac{2||x^t||_1}{M} \cap ... \cap E_l > \frac{2||x^t||_1}{M}] = P[E_1 > \frac{2||x^t||_1}{M}]...P[E_l > \frac{2||x^t||_1}{M}] > \frac{2||x^t||_1}{M}] \leq \frac{1}{2^l}$$

As such, recall that $y_i = \min_{k=1,...,l} A_k[h_k(i)]$, so if all the arrays give an estimator that exceeds $\frac{2||x^t||_1}{M}$ in error, then y_i must have such errors too (noting that all errors are positive errors).

$$P[\text{error of } y_i \geq \frac{2||x^t||_1}{M}] \leq \frac{1}{2^l}$$

Setting $M = 2/\epsilon$, we see our success probability that the error is at most $\epsilon||x^t||_1$ has been amplified to the probability of $\geq 1 - (1/2^l)$.

21.3 Construction and Space Complexity

Suppose we want to satisfy the count-min query (error at most $\epsilon||x^t||_1$) with probability of failure δ. We take the previous construction and set $M = 2/\epsilon$, and l such that $\delta = \frac{1}{2^l}$, so $l = \log(1/\delta)$. Then, the size of the array we need to maintain is

$$lM = \log(1/\delta)(2/\epsilon) \in O(1/\epsilon \log(1/\delta))$$

Additionally, we need to store the hash functions. Recall that for universal hash functions $h : U \mapsto [M]$ where $|U| = 2^u, |M| = 2^m$, we can represent the function with a $m \times u$ matrix. The size of the hash functions we need to store is:

$$l \log(M) \log(U) = \log(1/\delta) \log(2/\epsilon) \log(|U|)$$

Combining the two, the total space is therefore $\frac{1}{\epsilon}\text{polylog}(1/\delta, 1/\epsilon, |U|)$.

21.4 Minhash

The idea of Minhash function is useful in determining the similarity of documents. The documents are processed into shingles, which turn into sets. Then we measure the similarity of documents by the sets it created, using Jaccard Similarity. We express the Jaccard Similarity of A, B as $\text{SIM}(A, B)$:

$$\text{SIM}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Comparing the similarity through sets alone takes too much space, so we need to represent each set by a signature and use it to estimate the similarity.
See details of shingles, characteristic matrix of sets and minhash on lecture slides.

Theorem 21.1.

$$P[minhash(S) = minhash(T)] = SIM(S, T)$$

over a randomly chosen permutation for minhash.

Proof Sketch:
Set X to be the number of rows in the characteristic matrix of S, T with entry 1 for both S, T, Y to be the number of rows with entry 1 for either S, T but not both, and Z to be the number of rows with entry 0 for both S, T .
Then, observe that $S \cap T$ contains all elements that are in S, T, which means the row of the element has 1 in both S, T (type X). Also, observe that rows of type X cannot be rows of type Y, and also rows of type X or Y are simply element rows with membership in either S or T, which is just $S \cup T$. Since type X and Y are exclusive, then $X + Y = |S \cup T|$:

$$\text{SIM}(S, T) = \frac{X}{X + Y}$$

Now, for the minhash of S, T be equal, we need the permutation such that the row of type X appear before row of type Y. The probability of that is $\frac{X}{X+Y}$. The intuition is that if $Z = 0$, then the first row must be where we decide if this is row of type X or Y, and probability of success is $\frac{X}{X+Y}$. Intuitively, the value of Z should also not affect the probability of success, since as long as no row of type X or Y appears, the appearance of row of type Z does not affect the odds of row X or Y appearing first.
■

We can generate the minhash signature. Let $h_1, ..., h_n$ be different minhash functions generated by independently chosen permutations. We set the signature of set S, $\text{SIG}(S) = [h_1(S), ..., h_n(S)]$. We can also represent multiple signatures through a matrix where each row is a hash function h_i, each column is a set S_i, and each entry at the ith row jth column as $h_i(S_j)$.
From **Theorem 21.1** we see a possible empirical estimate of $\text{SIM}(S, T)$ using the signature of S, T. The fraction of coordinates where $\text{SIG}(S), \text{SIG}(T)$ are the same is a good estimate for $P[\text{minhash}(S) = \text{minhash}(T)]$, which means we can use this to estimate $\text{SIM}(S, T)$.