11TH TUTORIAL ON RANDOMIZED ALGORITHMS

Streaming and sketching

1. *The k Minimum Values (KMV) sketch.* We look at a different approach for counting distinct elements (quite popular in practice): Let $h : [N] \to [0, 1]$ be a hash function[1]. *For a parameter k, store the k smallest hash values of the distinct stream elements, i.e., we store k pairs (item j, h(j));* note that the sketch is not updated if any stream item arrives again. When queried for cardinality, return $(k-1)/v_k$, where v_k is the k-th smallest hash value (the largest one stored).

Analyze the algorithm and prove that for $k \geq c/\varepsilon^2$ (where c is a large enough constant) the algorithm gives an ε-approximation of F_0 = the number of distinct elements.

a) Show that the probability of $(k-1)/v_k > (1+\varepsilon) \cdot F_0$ is an arbitrarily small constant (depending on c). For now, assume a fully random hash function (i.e., all items hashed into $[0, 1]$ uniformly and independently). A similar analysis applies to the other inequality, that is, bounding the probability of $(k-1)/v_k < (1-\varepsilon) \cdot F_0$.

b) What is wrong with h being fully random? What kind of hash functions would be sufficient for the analysis?

c) How to get a $(1+\varepsilon)$-approximation with probability $1-\delta$ for any $\delta > 0$? How does the space complexity depend on δ?

d) *New:* Let's explore other useful properties of KMV: How to estimate the size of the union or intersection of two streams, each processed separately using one instance of KMV but both using the same hash function? What is the error?

2. *Count-Min sketch for frequency estimation.* Similarly to CountSketch, our aim is to estimate frequencies under both insertions and deletions but with a different guarantee and assuming all frequencies are non-negative at the end. We use the following sketch for estimating frequencies f_i":

Algorithm 1 Count–Min Sketch

Initialize:

1: $C[1 \dots t][1 \dots k] \leftarrow \vec{0}$, where $k \leftarrow \frac{2}{\varepsilon}$ and $t \leftarrow \lceil \log(1/\delta) \rceil$
2: Choose t independent hash functions $h_1, \dots, h_t : [n] \to [k]$, each from a universal family

Process ***(token*** (j, c)***)*****:**

1: **for** $i \leftarrow 1$ **to** t **do**
2: $\quad C[i][h_i(j)] \leftarrow C[i][h_i(j)] + c$
3: **end for**

Output ***(query*** a***)*****:**

1: report $\hat{f}_a = \min_{1 \leq i \leq t} C[i][h_i(a)]$

a) Using the assumption that all frequencies are non-negative at the end, derive lower and upper bounds on the estimator of a single row. That is, for any $a \in [n]$ and row $i \in [t]$ show that

$$0 \leq C[i][h_i(a)] - f_a \leq \varepsilon \cdot \|\mathbf{f}\|_1 \,. \tag{1}$$

with a constant probability.

[1] We are hashing into a real interval $[0, 1]$ for simplicity of the analysis. Instead, when implemented, we would hash into integers $\{0, \dots, R\}$ for large enough $R = \text{poly}(N)$; you may try to figure out how this changes the analysis.

b) Show a high probability bound for the final estimator $\hat{a}_j$ for frequency f_a.

c) Compare CountSketch (from the lecture) and Count-Min sketch, both in terms of their description and their properties.

d) Bonus: Can you derive a more refined bound on the error of Count-Min? That is, replace $\|\mathbf{f}\|_1$ by a smaller quantity in (1). (Hint: show that, say, $k/8$ most frequent items do not collide with a in row i with constant probability.)

3. *Bonus: linear sketching.* You have seen several sketches: the Tug-of-War sketch for estimating the second frequency moment F_2 and CountSketch and Count-Min sketch for estimating frequencies. In fact, these are *linear sketches*, that is, they can be viewed as linear maps of the frequency vector $\mathbf{f}$ to a much smaller dimension.

a) Here's a practically very useful property: Suppose we have a massive dataset distributed over many machines. How to efficiently obtain a linear sketch (say, a CountSketch) of the whole dataset, using as little communication among the machines as possible?

b) Take any of these sketches and describe the matrix of the corresponding linear map (dimensions and entries). How do we store this matrix?

c) How to get an estimate of the Euclidean distance between high-dimensional vectors u and v, where u and v are described by point-wise updates to their coordinates? What error guarantee can we get? (That is, the input is a sequence of triples $(u/v, j, c)$, where the first entry determines whether we update u or v, the second one specifies the coordinate, and the last one is the increase of the value of coordinate j, which may be negative.)

d) How to estimate the inner product between two vectors u and v (specified as above)? What error guarantee can we get?