Data mining: homework 1

Edo liberty

Assignment

1. Describe an algorithm which samples roughly n elements from a stream, each uniformly and independently at random. More precisely, in every point in the stream, after processing N elements, each of the elements $a_1, \ldots, a_N$ should appear in your sample with probability exactly n/N. Note that you clearly do not know N in advance and that you cannot reread the stream.

2. Prove the correctness of your algorithm.

3. What is the complexity of processing each element in your algorithm?

4. Bound the probability that your algorithm samples more than $2n$ elements.

Solution

There are a lot of possible solutions for this problem and anything that works will be deemed correct.

1. Before describing the algorithm, lets provide some intuition about how it works. Imagine we choose for each element a_i a random number r_i uniformly and independently at random from the range $[0, 1]$. We can keep all the elements for which $r_i \leq n/N$ we clearly do the right thing. The question is, how do we do that when we don't know N in advance? The idea is to assume that the current element is the last one in the stream at every point.

   More precisely, when receiving an element a_i the algorithm generates a random number r_i uniformly and independently at random from the range $[0, 1]$. If $r_i > n/i$ the element is ignored. Otherwise, it is kept. Also, in every step, we go over all the elements a_j such that $1 \leq j \leq i-1$ and discard those for which $r_j > n/i$. Note that we had to have kept all the values r_j as well.

2. Clearly, doing this gives the same intuitive solution as above. In other words, when the stream ends ($i = N$) we have that our sample contains

all the elements a_j for which $r_j \leq n/N$. Since r_j are uniformly and independently chosen from $[0,1]$, each element is in the sample *independently* with probability n/N.

3. A naive implementation of this algorithm requires $O(n)$ operations per read element. This is because, in every step we have to go over all the sampled elements and discard the ones for which $r_j > n/i$. However, if we keep the picked elements in a sorted list, we can go to the end of the list and remove only those. This requirs only $O(1)$ operations per removed element. Or, discarding d_i elements in step i requires $O(d_i)$ operations. This comes with the price of having to insert picked elements into the sorted list in $O(s_i)$ operations. Here s_i denoted the number of samples that we kept in step $i-1$. We denote by z_i the indicator variable for the event that element a_i is picked. Let us compute the expected running time of this algorithm $T(N,n)$.

$$\begin{aligned} E[T(N,n)] &= \sum_{i=1}^{N} E[\text{Cost of step } i] && (1)\\ &= \sum_{i=1}^{N} [E[O(s_i)z_i] + O(d_i)] && (2)\\ &= O(\sum_{i=1}^{N} \frac{n^2}{i} + N) && (3)\\ &= O(n^2 \log(N) + N) && (4) \end{aligned}$$

We used here that s_i and z_i are independent so $E[O(s_i)z_i] = O(E(s_i))E[z_i]$. Also, $\sum_{i=1}^{n} d_i \leq N$ and $\sum_{i=1}^{n} \frac{1}{i} \approx \log(N)$. Amortizing over the length of the stream we get that the expected cost per element is $T(N,n)/N = O(\frac{n^2 \log(n)}{N} + 1)$ which is $O(1)$ if the stream is long enough, i.e. $\frac{n^2 \log(N)}{N} \leq 1$.

Alternatively, we can keep all the picked elements in a sorted heap according to the r values which results in $O(\log(n))$ operations per insertion and deletion (amortized)

4. Bounding the probability of sampling more than $2n$ elements can be done using the Chernoff bound.

$$\Pr[X > (1+\varepsilon)\mu] \leq e^{-\mu\varepsilon^2/4}$$

We have that $\mu = E[s_N] = n$ and $\varepsilon = 1$. Thus:

$$\Pr[s_N > 2n] \leq e^{-n/4} \ .$$

We can also make sure that in no point in the algorithm we have $s_i > 2n$ by the union bound. Making sure that in all N steps of the algorithm this

happens with probability at least $1 - \delta$.

$$\sum_{i=1}^{n} \Pr[s_i > 2n] \leq Ne^{-n/4} \leq \delta \ .$$

This is true if $n > 4 \log(\frac{N}{\delta})$.