

Algorithms for Big Data
Prof. John Ebenezer Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 21
Reservoir Sampling

(Refer Slide Time: 00:31)

Stream of n items
↳ Not known in advance

$a_1 a_2 \dots a_i \dots a_n$

If: $\Pr(a_i \text{ is chosen}) = \frac{1}{n}$

In this segment we are going to talk about sampling from a stream. So, we are going to assume that our stream has distinct items. This is not essential, but it helps to keep the problem clean and simple. So, we have a stream of some n items that are showing up, but we do not know n . Our goal is to sample from the stream so that we output an item from the stream and the probability that each of these items is output must be equally likely. So, in other words, for all i the probability that a_i is chosen must be 1 over n .

And so, this without the knowledge of n is somewhat tricky, but as it turns out, there is a nice clean way to sample from a stream and here is how this algorithm goes.



(Refer Slide Time: 01:20)

Reservoir Sampling

$s \leftarrow \perp$ $\square \xrightarrow{s}$

When i^{th} item arrives, with prob $1/i$, $s \leftarrow a_i$

S.T.
 $\Pr(s = a_i) = \frac{1}{n}$



We have a memory cell, a single cell because we want to sample a single item and let us call that as s . s is initialized to bottom, so initially s does not have anything useful. And now, let us see how we update s as the stream flows by.

So, every time we look at the i th item, we toss a coin with bias 1 over i and if that coin comes up heads, in other words, probability 1 over i , we replace the current contents of s with a_i and of course, a_i is the i th item. So, now, and that is it. This is the algorithm. We need to now prove this algorithm actually outputs a correct sample that we want. In other words, at the end of the stream when all n items have been seen the probability that the memory cell s contains a_i should be equal to $1/n$ and this must be true for all.

So, now let us go about proving that our algorithm, that reservoir sampling algorithm is in fact, correct. So, how do we do that? Let us look at the probability with which the i th item is finally output.

(Refer Slide Time: 02:51)

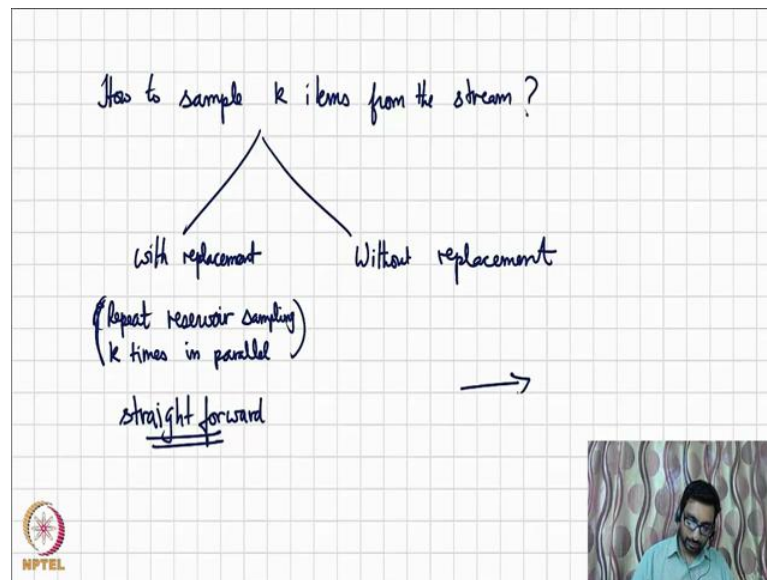
$$\begin{aligned} \forall_i \Pr(s = a_i) &= \frac{1}{i} \left(1 - \frac{1}{i+1}\right) \left(1 - \frac{1}{i+2}\right) \dots \left(1 - \frac{1}{n}\right) \\ &= \frac{1}{i} \left(\frac{i+1-1}{i+1}\right) \left(\frac{i+2-1}{i+2}\right) \dots \left(\frac{n-1}{n}\right) \\ &= \frac{1}{i} \left(\frac{\cancel{i+1}-1}{\cancel{i+1}}\right) \left(\frac{\cancel{i+2}-1}{\cancel{i+2}}\right) \dots \left(\frac{\cancel{n-1}}{n}\right) \\ &= \frac{1}{n} \end{aligned}$$

First of all, the i th item must have had a chance to, must have gotten into the memory cell and that at the point in time when the i th item has been seen, the i th item gets copied into the memory cell with probability 1 over i and furthermore, if I , finally, outcome must be i th item. None of the subsequent items can replace i because once it is replaced, it is gone, that is the way the algorithm works.

So, what is the probability with which the next i th item, the i plus 1 th item does not replace the i th item from s ? It replaces the i th item with probability 1 over i plus 1 and therefore, it does not replace the i th item with probability 1 minus 1 over i plus 1 and this same pattern has to continue. The i plus 2 th item also should not be able to, should not have replaced item i . We get these, this sequence of multiplicative terms.

So, let us see how this multiplicative terms, how they expand out. As it turns out, they have a nice form. When we expand them out, it is a sort of a telescopic product and most of the terms cancel out and we are, finally, left with 1 by n and this is of course, true for all i which is exactly what we wanted to prove.

(Refer Slide Time: 04:40)



Let us now move on to sampling, not just one item, but some k items. Here, we need to a bit more careful. Now, this sampling can be done in one of two ways. We can either sample with replacement or without replacement. Now, with replacement is somewhat straight forward. We can simply perform k independent sampling, reservoir sampling for single items. So, that is relatively straight forward.

Let us now come to the aspect of sampling without replacement. So, now if you consider any, any subset of k items, it should be equally likely that subset should be picked with equally likelihood as any one of the n chooses k possible, such k items subsets. So, how do we do that? Well, here we again rely on the previous intuition from the reservoir sampling. Now, we just need to be a little bit more careful. This is quite intuitive. Here is how we do this.

(Refer Slide Time: 06:01)

For the first k items, simply assign $s_i \leftarrow a_i \quad 1 \leq i \leq k$

For $i > k$
 $j = \text{random number from } 1 \dots i \text{ (UAR)}$
if $j \leq k$
 $s_j = a_i$

Instead of maintaining one, one memory cell, we maintain k memory cells. These k cells are denoted s_1, s_2 and so on, up to s_k . And for the first i , for the first k items, you simply assign the item a_i by ranging from 1 to k to the memory cell s_i .

Now, what happens if i is greater than k ? We first generate a random number and this random number is generated uniformly at random between 1 and i and if that number that randomly generated number happens to be between 1 and k , then we replace the j th memory cell. So, the random number, if it is less than or equal to k , that means, it is the, j is, s_j is somewhere here and we replace the contents of s_j with a_i and that is what we do here.

So, let me give you a little bit of intuition for why this works. Now, the first k items we have to fill them in because if the stream stops at k , there is only a single k item subset and that must be the subset that is, that is output. Now, as think of the, well, think of the stream extending past these first k items and consider the item a_i with probability k over i . So, this is, this is k and this is i with probability k over i . It replaces an item from the, from the memory cells and which memory cell should not replace it. It replaces exactly the, the index that was generated randomly.

So, why is this algorithm correct? What, actually first of all what do we need to show? It will say that this algorithm is correct. So, first of all notice that this algorithm is going to produce a subset of k items and it is not with repetition. So, that part is taken care of.

Now, what we want to show is that the probability with which item a_i occurs in the chosen subset is k over n .

(Refer Slide Time: 08:58)

Let $S = \{a_1, a_2, \dots, a_k\}$ at the end of the stream

Claim: $\Pr(a_i \in S) = \frac{k}{n}$

$\hookrightarrow = \frac{k}{i} \left(1 - \frac{1}{i+1}\right) \left(1 - \frac{1}{i+2}\right) \dots \left(1 - \frac{1}{n}\right)$

$= \frac{k}{n}$

$\Pr(\text{any fixed } k\text{-item subset is chosen}) = \frac{1}{\binom{n}{k}}$



So, more, more formally, let be the set of items that we output as the samples, basically the contents of the memory cells, when we reach the end of the stream. Now, we need to show that the probability with which an item a_i is in, that the set s is equal to k over n . So, how do we show, that well, first of all the item a_i must have been placed in the, in one of the memory cells and that happens with probability k over i and then it should remain in the memory cell in which it was placed.

Let us, without loss of generality, say that the memory cell was placed in, but one some particular location, the i plus 1th item should not have fallen in that particular location, the i plus 1th item falls in a particular location with probability 1 over i plus 1. So, it does not fall in that location with probability 1 minus 1 over i plus 1.

(Refer Slide Time: 10:19)

For the first k items, simply assign $s_i \leftarrow a_i \quad 1 \leq i \leq k$

For $i > k$
 $j = \text{random number from } 1 \dots i \text{ (UAR)}$
if $j \leq k$
 $s_j = a_i$



So, let me go back to the picture to explain exactly why this happens. So, let us assume that the i th item gets placed in this cell s_j . Now, the $i + 1$ th item can, will also generate a random number. If it generates the random number anywhere here, no problem; if it generates the random number that falls within this range, well we need to be a bit careful, if it falls anywhere in this region that is OK, it falls anywhere in this region that is ok.

The only location it should not fall into is s_j because if it falls in s_j , then our item a_i has to be evicted and the probability of that is 1 over $i + 1$ and therefore, the good probability that a_i gets retained is 1 minus 1 over $i + 1$ and of course, this has to be maintained throughout. So, you get a similar telescopic product, which finally, will evaluate to k over n , which is exactly what we wanted.

Now, as an exercise think about this. What we have shown is, that the item a_i is one of the elements of the subset that we are outputting at the end of the stream with probability k over n . Thus, this translates to what we actually started out, with what we find. What we wanted was that the probability that any fixed k item subset is chosen as the subset as output should be exactly equal to 1 over n choose k . So, have we proven this? Think about this, that is your exercise. So, that completes our segment 2.