# L7: Reservoir Sampling

scribe(s): *Kavitaben Sheth*

Streaming algorithms are process that takes data in input stream and examine in few passes typically one pass. Thease algorithms has limited memory available and limited processing time to process each data item. All sampling methods that process the input file in one pass can be characterised as reservior algorithms.

Reservoir Sampling is a family of randomized but fast algorithms for selecting a random sample of n records without replacement from a pool of N records, where value of N is unknown beforehand. Reservoir algorithm select first n records of the file into a 'reservoir' and rest of records are process sequentially.

## 7.1 Reservoir algorithm

Random sampling is basic to many computer applications in computer science, statistics, and engineering. The problem is to select without replacement a random sample of size n from a set of size N.

The first step of any reservoir algorithm is to put the first n records of the file into a reservoir. The rest of the records are processed sequentially; records can be selected for the reservoir only as they are processed. An algorithm is a reservoir algorithm if it maintains the invariant that after each record is processed a true random sample of size n can be extracted from the current state of the reservoir. This algorithm runs in O(N) because the entire file must be scanned.

So, we assume we have a continuous random variable that can be generated quickly and whose distribution approximate well. It can be generated very quickly, in constant time. So, At the ith element of S, the algorithm generates a random number j between 1 and i. If j is less than n, the jth element of the reservoir array is replaced with the ith element of S. In effect, for all i, the ith element of S is chosen to be included in the reservoir with probability n/i. Assume i+1 to m elements kept in reservior. Now, probability of not replaced element from reservior is 1-(m-i) /m = i/m. Total probability of kept and not replaced elements is (n/i)*(i/m) = n/m. So, when algorithm finished executing, each element has equal probability of being chosen from reservior.

## 7.2 Approximate Counts

The approximate counting problem counts large number of events using small amount of memory.

Consider query Interval I subset with elements [n] count(I) = $|a_i inA|a_i inI|$

Goal is to find data structure S such that for every query interval probablity = $|S(I) - count(I)| > eps * m.Probablity < delta.$

### 7.2.1 Chernoff Inequality

Let $X_1, X_2, ..., X_r$ be independent reserviors.
Let $\triangle_i = max(X_i) - min(X_i)$
Let $M = \sum_{i=1}^{r} X_i$

Probablity = $Pr[|M - \sum_{i=1}^{r} E[X_i]| >= \alpha] <= 2exp(\frac{-2\alpha^2}{\sum_i \triangle_i^2})$

Often : $\triangle = max_i \triangle_i$ and $E[X_i] = 0$ then :

Probablity = $Pr[|M| >= \alpha] <= 2exp(\frac{-2\alpha^2}{r\triangle_i^2})$

Let S be a random sample of size k = $O((\frac{1}{eps^2})log(\frac{1}{\triangle}))$

$S(I) = |ScapI| * (\frac{m}{k})$

Each $s_i$ in I with probability (count(I)/m)

Reservoir $RVY_i = \{1$ if $s_i$ in I, 0 if $s_i$ not in I$\}$
$E[Y_i] = count(I)/m$
Reservoir $RVX_i = (Y_i - count(I)/m)/k$
$E[X_i] = 0$
$\triangle < 1/k$
$M = sum_i X_i$ == error on count estimate by S
$Pr[|M| > eps] < 2exp(\frac{-2eps^2}{(k*(1/k^2)})) < \triangle$