ECEN 689
Special Topics in Data Science for Communications Networks

Nick Duffield
Department of Electrical & Computer Engineering
Texas A&M University

Lecture 6
Order Sampling
Order Sampling

- Approaches to sampling so far
  1. Event (e.g. sample or discard) has some probability $p$
     - E.g. parameter for packet sampling, or in IPPS
     - Generate random $r$ Uniform in $[0,1]$
     - Event occurs if $r \leq p$
  2. Distribution (e.g. of the skip in skip counting)
     - Generate random variate drawn from distribution

- Order Sampling
  - Generate a single random number $r_i$ for each item $i$
  - Events, distributions implicitly indicated from the $\{r_i\}$
Order Sampling $k$ from $n$ items uniformly

- Each item $i$ generates rank $r_i$ independently from Unif[0,1]

- Order the items increasing in $r_i$ and sample $k$ smallest

- The distribution of \{${r_i}$\} is **permutation invariant**
  \[
  \Pr[ r_1 \in A_1 , \ldots , r_n \in A_n ] = \Pr[ r_{\pi(1)} \in A_1 , \ldots , r_{\pi(n)} \in A_n ]
  \]
  for any permutation $\pi$ of \{1, $\ldots$, $n$\}

- Each selection of $k$ objects is equally likely
  \[
  \Pr[ \{r_1,\ldots,r_k\} < \{r_{k+1},\ldots,r_n\} ]
  = \Pr[ \{r_{\pi(1)},\ldots,r_{\pi(k)}\} < \{r_{\pi(k+1)},\ldots,r_{\pi(n)}\} ]
  = \Pr[\{\pi(1),\ldots,\pi(k)\} \text{ sampled}]
  \]

- Desired result: uniform sampling
Reservoir Sampling via Order Sampling

- Stream of \( n \) objects, reservoir capacity \( m \)
- Each arriving item \( i \), generate rank \( r_i \) in \( \text{Unif}[0,1] \)
- Take first \( m \) items
- Each subsequent item
  - Provisionally add to reservoir
  - Discard largest rank item

Example:
\( m=9 \)

\[\begin{array}{cccccccccc}
\text{x1} & \text{x3} & \text{x9} & \text{x7} & \text{x2} & \text{x8} & \text{x1} & \text{x4} \\
\end{array}\]
Computational Properties of Order Sampling

• Order Sampling can be simply parallelized
  – Divide data into subsets and order sample k items from each subset
  – Take union of outputs, then order sample k items from union

• Previous slide suggested maintaining items in linear order
  – Computationally expensive: O(m) per insertion if reservoir size m

• Only need abstract properties of a priority queue
  – Each item $i$ is equipped with a priority $r_i$ that does not change
  – Items can be inserted in the queue
  – The item of highest priority can be retrieved from the queue
    • Or equivalently, lowest priority

• There are more efficient a priority queue implementations
  – Well known: a heap, which is $O(\log m)$ per insertion, removal
Heaps

- Heap (low priority version)
  - Each node has at most two children
  - Parents have lower priority than children
  - Each depth of is filled in order (top to bottom, left to right)

- Two possible heaps storing the numbers \{1,..9\}

- Heaps of same size have same topology; balanced binary
Heap Insertion

- Insert new item at next free position
- Bubble up by swapping with until a heap is obtained
- Have to do at most h swaps, where \( h = O(\log n) \) is tree depth

- Insert(8)
Heap Removal

• Only the root node is removable
  – Lowest priority item when implementing a (low)-priority queue
• Remove root node
• Move node in last position to root
• Bubble down by swapping with smaller child until heap formed
  – Takes at most h swaps, where h = O(log n) is tree depth
Order sampling in databases

- Table of records \( \{i, k_i \}, i=1,...,n \), \( k_i \) = column values of record \( i \)
- Generate ranks, but sample later when needed
- Initialization
  - Generate \( r_i \) in \( \text{Unif}[0,1] \) for all \( i \)
  - Sort the records increasing order in \( r_i \)
    - How to do this using a heap? What is the computational cost?
- Sampling
  - Want \( m \) samples from selection subset \( S \subset \{1,2,...,n\} \)
  - Take \( m \) lowest rank matching records in \( S \)
- Reuse: repeat for any \( (S,m) \) using same order
- Costs:
  - One time sort cost + sequential access
Weighted Order Sampling

• Suppose each item $i$ has a non-uniform weight $x_i$
• Can we do **weighted order sampling**?
• General idea:
  – Choose rank $r_i$ as function $R(x_i, u_i)$ of weight $x_i$ and random $u_i$ in $\text{Unif}[0,1]$
• Different choice of rank function $R$ can be used to realize different statistical objectives:
  – Desired sampling probability of $i$ as a function of $x_i$
• We’ll look at two cases
  – “Weighted random sampling”: yields sampling probability $x_i / \Sigma_i x_i$
  – “Priority sampling”: sampling similar to IPPS
A weighted sampling scheme

- \( n \) items with weights \( \{x_1, x_2, \ldots, x_n\} \) and sum \( X = \sum x_i \)
- Consider weighted sampling of \( k < n \) items w/o replacement
- Sample 1 item in each of \( k \) rounds
  - 1\(^{st} \) round: sample \( i \) with \( p_i = \frac{x_i}{X} \)
  - 2\(^{nd} \) round: sample \( j \) with \( p_{ij} = \frac{x_j}{X - x_i} \), conditional on sampling \( i \) in 1\(^{st} \)
  - Generally: subtract off weights of previously sampled items from \( X \)

\[
p_i = \frac{x_i}{X} \quad p_{ij} = \frac{x_j}{X - x_i} \quad p_{ijm} = \frac{x_m}{X - x_i - x_j}
\]

\( k \) objects selected
“Weighted random sampling” via order

- n items with weights \{x_1, x_2, \ldots, x_n\}; sample k from n
- Order sampling with rank \( r_i = -\log(u_i)/x_i \)
- Pick k items of smallest rank
- Theorem: **same sampling distribution** as previous scheme
- Proof:
  1. CDF of \( r_i \) is \( F_i(y) = \Pr[r_i \leq y] = \Pr[u_i \geq \text{Exp}(-y x_i)] = 1 - \text{Exp}(-y x_i) \)
      In other words: \( r_i \) has exponential distribution with mean \( 1/x_i \)
  2. For any j, \( \Pr[x_j \text{ selected first}] = \Pr[\text{item j has min rank}] = \int_{0}^{\infty} dF_j(y) \prod_{i \neq j} (1 - F_i(y)) = \int_{0}^{\infty} dy x_j e^{-yX} = \frac{x_j}{X} \)
  3. Due to memoryless property of exponential distributions, the \( \{r_i - r_j\} \) for \( i \neq j \), conditional on \( r_j \) being smallest, are independent Exponential[1/x_i]
  4. Recurse over remaining weights.
Priority Sampling

• Resembles an order sampling version of IPPS
  – Which rank function should be used to yield sampling probability
• Each item $x_i$ has priority $z_i = x_i / r_i$ with $r_i$ random $\text{Unif}(0,1]$
• Estimation
  – Let $z^* = (k+1)^{st}$ highest priority
  – Top-k priority items: weight estimate $x'_i = \max\{ x_i, z^* \}$
  – All other items: weight estimate $x'_i=0$
• Main result
  – $x'_i$ is an unbiased estimator of $x_i$!
Priority Sampling & Unbiased Estimation

- Show that $E[x'_i] = x_i$ for any $i$.
- Let $A(z')$ denote the event that the $k^{th}$ largest of the other priorities \{z\_j: j\neq i\} takes the value $z'$
- Will show that $E[x'_i | A(z')] = x_i$, for any $z'$, and so $E[x'_i] = x_i$
- Proof:
  - Suppose $z_i < z'$: then $z_i$ is $< k^{th}$ largest priority, so is not sampled
  - Suppose $z_i > z'$: then $z'$ is $(k+1)^{st}$ largest priority: $z' = z^*$
  - So
    \[
    \Pr[i \text{ sampled } | A(z')] = \Pr[z_i > z'|A(z')] = \Pr[r_i < x_i / z'] = \min\{1, x_i / z'\}
    \]
    Looks like IPPS but with a random sampling threshold $z'$
  - And
    \[
    E[x'_i | A(z')] = E[x'_i|i \text{ sampled, } A(z')] \Pr[i \text{ sampled } | A(z')] \\
    = \max\{x_i, \, z'\} \min \{1, x_i / z'\} \\
    = x_i
Variance and Covariance

• Priority sampling is not independent sampling
  – Fixed size: \( P[j \text{ sampled} | i \text{ sampled}] < P[j \text{ sampled}] \)
• But estimates have zero covariance!
  – \( E[x'_i x'_j] = x_i x_j \)
• Consequence: additive variance (like independent IPPS)
  – \( \text{Var}[X'(S)] = \text{Var}[\sum_{i \in S} x'_i] = \sum_{i \in S} \text{Var}(x'_i) \)
• Actually variance is difficult to compute explicitly, but:
  – Upper bound: \( \text{Var}[X'(S)] \leq X(S)^2/(k-1) \)
• Compare with IPPS
  – \( \text{Var}[X'(S)] \leq X(S)^2 / k \) slightly tighter
Priority Sampling in Databases

• One Time Sample Preparation
  – Compute priorities of all items, sort in decreasing priority, no discard

• Sample and Estimate
  – Estimate subset sum $X(S) = \sum_{i \in S} x_i$ by $X'(S) = \sum_{i \in S} x'_i$
  – Method: select items in decreasing priority order
Bounds on variance vs. computation cost

1. Variance bounded
   - $S' = \text{first } k \text{ items from } S$: relative variance bounded $\leq 1/(k-1)$
   - $x'_1 = \max\{ x_i, z^* \}$ where $z^* = (k+1)^{\text{st}} \text{ highest priority in } S$

2. Computation cost bounded
   - $S' = \text{items from } S \text{ in first } k$: execution time $O(k)$
   - $x'_1 = \max\{ x_i, z^* \}$ where $z^* = (k+1)^{\text{st}} \text{ highest priority}$