Online unsupervised learning

A: Online computation

Online unsupervised learning

- Endless stream of data: $x_1, x_2, x_3, \ldots$
- Repeat forever:
  - Get data point x
  - Update model

Space allowed: O(size of model)

Online computation of the mean

- $\mu = 0$ (mean so far)
- For $t = 1, 2, 3, \ldots$:
  - Get x_t
  - Update μ:

Why is this correct?

Online computation of the median

How to do this?

B: Maintaining a random subset

Useful primitive: Maintaining a random subset

Goal: Keep a random sample of k of the points encountered so far.

Let's start with the $k = 1$ case:

- $s = x_1$ (this is our random sample)
- For $t = 2, 3, \ldots$:
  - Get x_t
  - Update s:

Why is this correct?

Maintain k random samples (with replacement)

- $s_1 = s_2 = \cdots = s_k = x_1$
- For $t = 2, 3, \ldots$:
  - Get x_t
  - For $j = 1$ to k:
    - With probability $1/t$: Set $s_j = x_t$

Why is this correct?

Maintain k random samples (without replacement)

- $(s_1, s_2, \ldots, s_k) = (x_1, x_2, \ldots, x_k)$
- For $t = k+1, k+2, \ldots$:
  - Get x_t
  - Update:

Why is this correct?

Approximate median

- Maintain a random sample of k elements with replacement
- At any time t: let m_t be the median of these k elements

Here's what we can show using large deviation bounds:

Claim. Pick any $0 < \delta, \epsilon < 1$. If

$$k \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta},$$

then for any time t, with probability at least $1 - \delta$, the value m_t is a $(1/2 \pm \epsilon)$-fractile of $\{x_1, \ldots, x_t\}$.

C: Heavy hitters

The heavy hitters problem

- Stream of data $x_1, x_2, \ldots, x_m \in \mathcal{X}$ (where m is unknown)
- Parameter $\epsilon \in (0, 1)$

Keep track of elements whose frequency is $\geq \epsilon m$, as well as their frequencies.

How much space is needed to write down this information, if $n = |\mathcal{X}|$?

Can we design an online algorithm that uses only this much space?

Misra-Gries algorithm

Data structure:

- Hash table T of size $k = 1/\epsilon$
- Each element $x \in T$ has an associated value $V[x] \in \{1, 2, \ldots\}$

Algorithm:

- Table T is initially empty
- For $t = 1, 2, \ldots$:
  - Get x_t
  - If $x_t \in T$: increment $V[x_t]$
  - Else: If $|T| < k$: Add x_t to T, with $V[x_t] = 1$
  - Else: For each $x \in T$:
    - Decrement $V[x]$
    - If $V[x] = 0$, remove x from T

Algorithmic guarantees

Suppose that the number of times x appears in $x_1, \ldots, x_t$ is $\text{freq}_t(x)$.

Claim. The following is true at all times t, for every $x \in \mathcal{X}$:

$$\text{freq}_t(x) - t/(k+1) \leq V[x] \leq \text{freq}_t(x).$$

(Take $V[x] = 0$ for any $x \notin T$.)

Key idea:

- Think of $V[x]$ as holding the number of occurrences of each item x
- Once in a while, $k+1$ of these values are decremented
- By time t, the maximum number of such decrement-steps is $t/(k+1)$