Platform with Brains. Data with Soul.

Algorithms to Sample From Streams

Jonathan Arfa, Data Scientist @ Magnetic
What if you want to store a representative sample of data from a stream, in order to understand the distribution on-the-fly?
Data Streams

- Continuous
- Unknown length
- Hard to process with algorithms designed for batch data
Get a uniformly random, fixed-size sample from a stream of events of unknown length
Motivation for Reservoir Sampling

We want:

1. Exactly \( K \) samples
2. Unbiased samples - every event in an \( N \)-length stream (\( N \) could be unknown) should have an equal chance of being in our sample
3. Fast: an extra \( O(1) \) per event in the steam
4. Low Storage: only \( K \) events at any point
1. The first $K$ events in the stream automatically enter the reservoir

2. For the $i$th event, if $i > K$: there’s a $K/i$ probability that it enters the reservoir. If so, it replaces a randomly selected event that’s already there
Reservoir Sampling

class ReservoirClassic(object):
    def __init__(self, max_size):
        self.samples = []
        self.max_size = max_size
        self.i = 0

    def add(self, element, timestamp):
        size = len(self.samples)
        if size >= self.max_size:
            spot = random.randint(0, self.i - 1)
            if spot < size:
                self.samples[spot] = (element, timestamp)
            else:
                self.samples.append((element, timestamp))
        self.i += 1
Histogram of Samples in Classic Reservoir (size=3000)
Ages of Items in Reservoir for Classic Reservoir (size=3000) Over 24 Simulated Hours
But what if you don’t want unbiased samples?
Variable Incoming Rate Biased Samplers
Collaborators: Jonathan Arfa, Dan Crosta, Sam Steingold, Vladimir Vladimirov (formerly Magnetic)
1. Specify both $K$ (max_size) and the desired mean_age

2. The first $K$ events in the stream automatically enter the reservoir

3. For any subsequent event: enter the reservoir only if the current mean age of events in the reservoir is older than the desired mean age
   a. But what event does it replace? Two versions
class ExpVIRB(BaseVIRB):
    def __init__(self, max_size, mean_age):
        self.max_size = max_size
        self.desired_mean_age = float(mean_age)
        self.current_sum Ts = 0.0
        self.samples = []

    def add(self, element, timestamp):
        if len(self.samples) < self.max_size:
            self.current_sum Ts += timestamp
            self.samples.append((element, timestamp))
        elif (timestamp - (self.current_sum Ts / self.max_size) >
              self.desired_mean_age):
            spot = random.randint(0, int(self.max_size) - 1)
            self.current_sum Ts += timestamp - self.samples[spot][1]
            self.samples[spot] = (element, timestamp)
class UnifVIRB(BaseVIRB):
    def __init__(self, max_size, mean_age):
        self.max_size = max_size
        self.desired_mean_age = float(mean_age)
        self.current_sum_ts = 0.0
        self.samples = collections.deque(maxlen=max_size)

    def add(self, element, timestamp):
        if len(self.samples) < self.max_size:
            self.current_sum_ts += timestamp
            self.samples.append((element, timestamp))
        elif (timestamp - (self.current_sum_ts / self.max_size) >
            self.desired_mean_age):
            self.current_sum_ts += timestamp - self.samples[0][1]
            self.samples.append((element, timestamp))
VIRBs

Questions

1. To what extent are these random samples?

2. What happens if the incoming rate is too low to keep $K$ events at a defined $mean\_age$?
Ages of Items in Reservoir for Exp VIRB (size=3000, age=600) Over 24 Simulated Hours
Ages of Items in Reservoir for Unif VIRB (size=3000, age=600) Over 24 Simulated Hours

![Graph showing the ages of items in a reservoir over 24 simulated hours. The graph includes lines for max, 95th percentile, 50th percentile, and mean ages. The graph shows fluctuating ages with distinct peaks and troughs.]
def exp_mean_age_from_percentile(percentile, age):
    """
    Answers the question: If <percentile> of my samples from an Exponential
distribution are within <age> seconds, what's the mean age?
We're just solving the Exponential CDF for lambda.
    """
    return -age / log(1.0 - percentile)
def unif_mean_age_from_percentile(percentile, age):
    """
    Answers the question: If <percentile> of my samples from an Uniform distribution are within <age> seconds, what's the mean age?
    """
    return age * 0.5 / percentile
Aggarwal’s Reservoir Sampler

“On Biased Reservoir Sampling in the Presence of Stream Evolution” - Charu C. Aggarwal
Algorithm 3.1
1. The event enters the reservoir with probability $p_{in}$, otherwise it’s discarded.

2. If the current size of the reservoir is $N$ out of a maximum of $K$,
   a. the event replaces a random pre-existing event with probability $N/K$.
   b. Otherwise, it’s added to the end of the reservoir, making it bigger.
class AggarwalReservoir(object):
    def __init__(self, max_size, p_in=1.0):
        self.samples = []
        self.max_size = max_size
        self.p_in = p_in

    def add(self, element, timestamp):
        if random.random() < self.p_in:
            spot = random.randint(0, self.max_size - 1)
            if spot >= len(self.samples):
                self.samples.append((element, timestamp))
            else:
                self.samples[spot] = (element, timestamp)
Ages of Items in Reservoir for Aggarwal (size=3000)
Over 24 Simulated Hours

Graph showing the age of items in a reservoir over 24 simulated hours, with lines indicating max, 95th percentile, 50th percentile, and mean values.
• Want to sample uniformly over all events?
  ○ Old-school reservoir sampling

• Want to sample from a defined period of time with a defined shape?
  ○ VIRBs, courtesy of team Magnetic
## Overly Complicated Table

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
<th>Add new event if:</th>
<th>New events replace:</th>
<th>Samples over</th>
<th>Time till full reservoir</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reservoir Sampling</td>
<td>max size</td>
<td>random() &lt; (max_size / i)</td>
<td>random event</td>
<td>events (all)</td>
<td>K events seen</td>
<td>Uniform</td>
</tr>
<tr>
<td>Aggarwal’s 3.1</td>
<td>max size, p_in</td>
<td>random() &lt; p_in</td>
<td>random (it’s complicated)</td>
<td>events (recent)</td>
<td>Longer</td>
<td>Exponential</td>
</tr>
<tr>
<td>Uniform VIRB</td>
<td>max size, mean age</td>
<td>current age &gt; desired age</td>
<td>oldest event</td>
<td>time (recent)</td>
<td>K events seen</td>
<td>Uniform</td>
</tr>
<tr>
<td>Exponential VIRB</td>
<td>max size, mean age</td>
<td>current age &gt; desired age</td>
<td>random event</td>
<td>time (recent)</td>
<td>K events seen</td>
<td>Exponential</td>
</tr>
</tbody>
</table>
http://tech.magnetic.com/2016/04/virbs-sampling-events-from-streams.html