L2: Approximate Algorithms

Wei Wang @ HKUST(GZ)

March 25, 2025

Modified from Slides by Marta Arias, José Luis Balcázar, Ramon Ferrer-i-Cancho, Ricard Gavaldà, Department of Computer Science, UPC

Outline

1. Data streams are everywhere
2. Data stream model
3. Samping

Data streams are everywhere

- Telcos - phone calls
- Satellite, radar, sensor data
- Computer systems and network monitoring
- Search logs, access logs
- RSS feeds, social network activity
- Websites, clickstreams, query streams
- E-commerce, credit card sales

Example 1: Online shop

Thousands of visits / day

- Is this "customer" a robot?
- Does this customer want to buy?
- Is customer lost? Finding what s/he wants?
- What products should we recommend to this user?
- What ads should we show to this user?
- Should we get more machines from the cloud to handle incoming traffic?

Example 2: Web searchers

Millions of queries / day

- What are the top queries right now?
- Which terms are gaining popularity now?
- What ads should we show for this query and user?

Example 3: Phone Company

Hundreds of millions of calls/day

- Each call about 1000 bytes per switch
- I.e., about 1Tb/month; must keep for billing
- Is this call fraudulent?
- Why do we get so many call drops in area X?
- Should we reroute differently tomorrow?
- Is this customer thinking of leaving us?
- How to cross-sell / up-sell this customer?

Data Streams: Modern times data

- Data arrives as sequence of items
- At high speed
- Forever
- Can't store them all
- Can't go back; or too slow
- Evolving, non-stationary reality

https://www.youtube.com/watch?v=ANXGJe6i3G8

In algorithmic words ...

The Data Stream axioms:

1. One pass
2. Low time per item - read, process, discard
3. Sublinear memory - only summaries or sketches
4. Anytime, real-time answers
5. The stream evolves over time

Computing in data streams

- Approximate answers are often OK
- Specifically, in learning and mining contexts
- Often computable with surprisingly low memory, one pass

Main ingredients: approximation and randomization

- Algorithms use a source of independent random bits
- So different runs give different outputs
- But "most runs" are "approximately correct"

(ϵ, δ)-**approximation**

A randomized algorithm A (ϵ, δ)-approximates a function $f : X \to R$ iff for every $x \in X$, with probability $\geq 1 - \delta$

- (absolute approximation) $|A(x) - f(x)| < \epsilon$
- (relative approximation) $|A(x) - f(x)| < \epsilon f(x)$
- Often ϵ, δ are given as inputs to A, ϵ is **accuracy**, δ is **confidence**

In traditional statistics one roughly describes a random variable X by giving $\mu = E[X]$ and $\sigma^2 = Var(X)$

Obtaining (ϵ, δ)-approximation
For any X, there is an algorithm that takes m independent samples of X and outputs an estimate $\hat{\mu}$ such that

$$Pr[|\hat{\mu} - \mu| \leq \epsilon] \geq 1 - \delta$$

for

$$m = O(\frac{\sigma^2}{\epsilon^2} \ln \frac{1}{\delta})$$

This is general. (Proof omitted)
For specific X there may be more sample-efficient methods.

Some problems on data streams

- Keeping a uniform sample
- Counting total elements
- Approximating a discrete distribution
- Approximating distances

The solutions

- are interesting in streaming mode
- **reduce memory**
- **demonstrate some typical algorithmic tricks**

Sampling: dealing with velocity

$\mathbf{x} :=$ … p # q q 9 8 g r k % & q 3 4 n 3 1 i @ g a a …

- At time t, process element $\mathbf{x}[t]$ with probability α
- Compute your query on the sampled elements only
- You process about αt elements instead of t, then **extrapolate**.

Sampling: dealing with velocity AND memory

Similar problem:

- Keep a **uniform sample** S of elements of some size k
- At every time t, each of the first t elements is in S with probability k/t

Key challenge:

- How to make early elements as likely to be in S as later elements?

Reservoir Sampling [Vitter85]:

- Add the first k stream elements to S
- Choose to keep t-th item with probability k/t
- If chosen, replace one element from S at random

Reservoir sampling: why does it work?

Claim

For every t, for every $i \leq t$,

$$P_{i,t} := Pr[s_i \text{ in sample at time } t] = \frac{k}{t}$$

Proof: Suppose true at time t. At time $t + 1$,

$$P_{i+1,t+1} := Pr[s_{i+1} \text{ in sample at time } t + 1] = \frac{k}{t + 1}$$

and for $i \leq t$, s_i is in the sample S if

Reservoir sampling: why does it work?

Claim

For every t, for every $i \le t$,

$$P_{i,t} := Pr[s_i \text{ in sample at time } t] = \frac{k}{t}$$

Proof: Suppose true at time t. At time $t+1$,

$$P_{i+1,t+1} := Pr[s_{i+1} \text{ in sample at time } t+1] = \frac{k}{t+1}$$

and for $i \le t$, s_i is in the sample S if

- it was before, and
- **NOT** (s_{t+1} sampled and it kicks out exactly s_i)

Reservoir sampling: why does it work?

Claim

For every t, for every $i \leq t$,

$$P_{i,t} := Pr[s_i \text{ in sample at time } t] = \frac{k}{t}$$

Proof: Suppose true at time t. At time $t+1$,

$$P_{i+1,t+1} := Pr[s_{i+1} \text{ in sample at time } t+1] = \frac{k}{t+1}$$

and for $i \leq t$, s_i is in the sample S if

- it was before, and
- **NOT** (s_{t+1} sampled and it kicks out exactly s_i)

$$P_{i,t+1} = \frac{k}{t} \cdot (1 - \frac{k}{t+1} \cdot \frac{1}{k}) = \frac{k}{t} \cdot (1 - \frac{1}{t+1})$$
$$= \frac{k}{t} \cdot \frac{t}{t+1} = \frac{k}{t+1}$$