Foundations of Data Science

Lecture 11: Streaming Data

MING GAO

DaSE@ECNU
(for course related communications)
mgao@sei.ecnu.edu.cn

Dec. 9, 2016

Outline

Motivations

Sensor data

Most of the algorithms are mining a database, where all our data is available when and if we want it. But...

- Imagine a temperature sensor bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour (not very interesting since the data rate is so low).
- Given the GPS sensor, and let it report surface height instead of temperature. If it sends a 4-byte real number every tenth of a second, then it produces 3.5 megabytes per day (not very interesting since # sensors is only one).
- To learn something about ocean behavior, we might want to deploy a million sensors (is not very many since there would be one for every 150 square miles), each sending back a stream, at the rate of ten per second. Now we have 3.5 terabytes arriving every day.

Motivations cont.

Image data

- Satellites often send down to earth streams consisting of many terabytes of images per day.
- Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second.
- London has six million such cameras, each producing a stream.

Internet traffic

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its output.
- Normally, the job of it is to transmit data and not to retain it or query it.
- But there is a tendency to put more capability into the switch, e.g., the ability to detect denial-of-service attacks or the ability to reroute packets based on information about congestion in the network.

Motivations cont.

Internet traffic

- Web sites receive streams of various types. Many interesting things can be learned from these streams.
- For example, an increase in queries like "sore throat" enables us to track the spread of viruses.
- A sudden increase in the click rate for a link could indicate some news connected to that page, or it could mean that the link is broken and needs to be repaired.

Many applications

Videos, email messages, webpages, chats, search queries, shopping history, GPS trials, financial transactions, stock exchange data, electricity consumption, Astronomy, Physics, medical imaging, weather measurements, maps, telephony data, audio tracks and songs, etc.

Streaming data

Characters

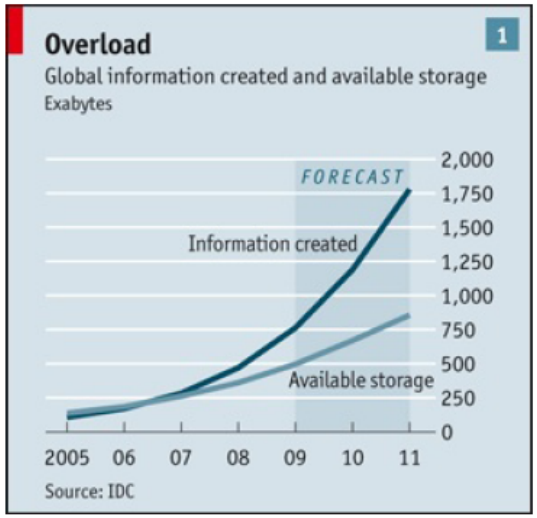
A sequence $\sigma =< a_1, a_2, \cdots, a_m >$, where the elements of the sequence are drawn from the universe $[n] := \{1, 2, \cdots, n\}$.

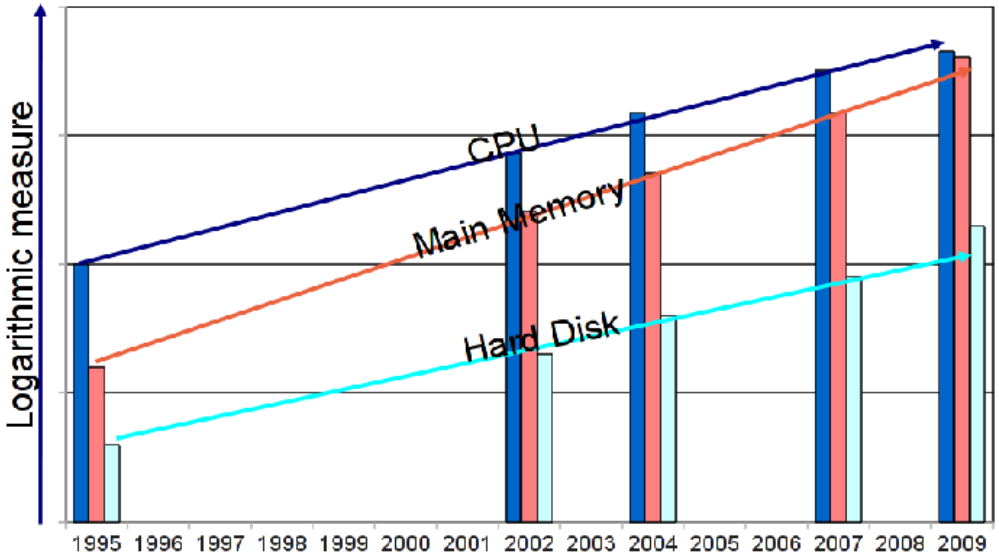
- Continuously arriving
- Large volume
- High speed

Algorithms for data stream

- Our central goal will be to process the input stream using a small amount of space s.
- We do not have random access to the tokens. We can only scan the sequence following the given order in p passes (some "small" integer p, for example $p = 1$).

The need for streaming data algorithm





Data growth

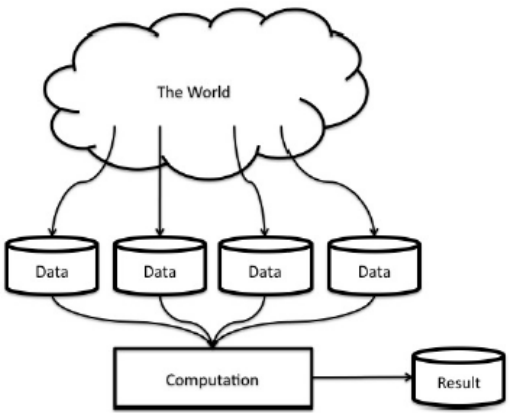
- IDC reports that enterprise data stores will grow an average of 60% annually.
- Moore law is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.

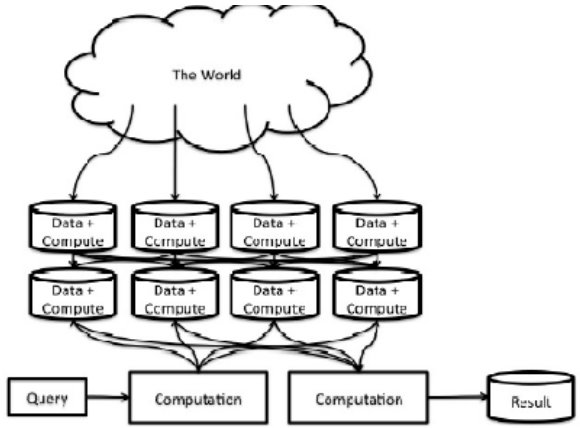
Streaming algorithm VS. traditional algorithm

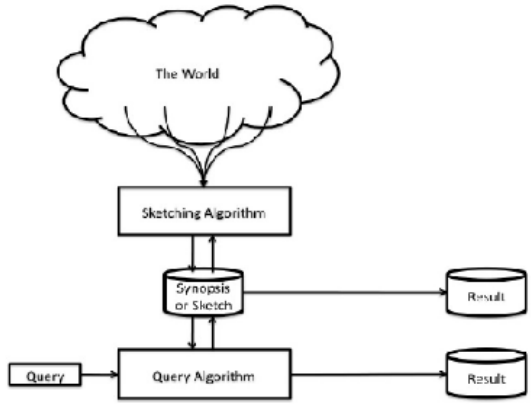
Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute, ϕ given access only to $C(\sigma)$.

- There exist exact solutions for some trivial tasks, such as count items, sum values, sample, find min or max.
- For non-trivial task, different problems maybe design different sketches to solve them.
- We are expected to give a bound for the estimation.







The quality of an algorithm's answer

Algorithm

We shall typically seek to compute only an approximation of the true value of $\phi(\sigma)$, because many basic functions can be provably not be computed exactly using sublinear space.

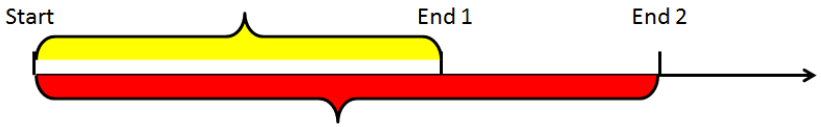
- Definition 1: let $\mathcal{A}(\sigma)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input σ; note that this is a random variable. Let ϕ be the function that $\mathcal{A}$ is supposed to compute. We say that the algorithm $(\epsilon, \delta)-$approximation ϕ if we have

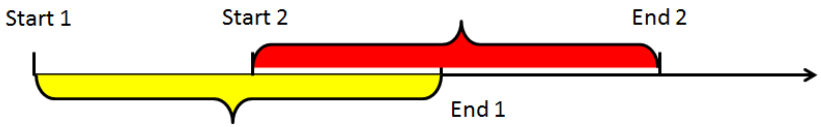
$$P\Big[|\frac{\mathcal{A}(\sigma)}{\phi(\sigma)} - 1| > \epsilon\Big] \leq \delta.$$

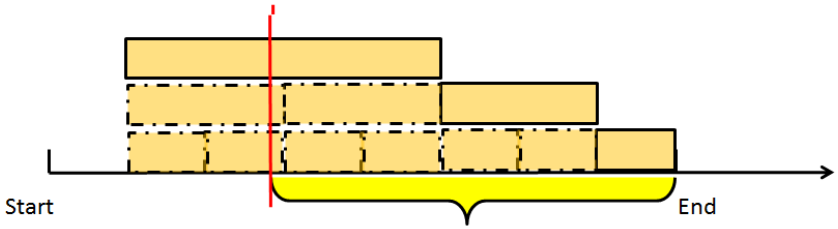
- Definition 2: In the above setup, the algorithm $(\epsilon, \delta)-$additively-approximation ϕ if we have

$$P[|\mathcal{A}(\sigma) - \phi(\sigma)| > \epsilon] \leq \delta.$$

Streaming data modeling







Model

- Landmark model:
- Sliding window model:
- Exponential model:

Counting problem

Problem definition

The algorithm must monitor a sequence of events, then at any given time output of the number of events thus far. More formally, this is a data structure maintaining a single integer n and supporting the following two operations:

- **update():** increments n by 1;
- **query():** must output (an estimate of) n.

Morris algorithm [Morris 1978]

1: initialize $X \leftarrow 0$;
2: for each update, increment X with probability $\frac{1}{2^X}$;
3: for a query, output $\widehat{n} = 2^X - 1$.

Analysis of Morris' algorithm

Analysis

Let X_N denote X in Morris' algorithm after N updates. Then, we have:

- $E2^{X_N} = N + 1$.
  Proof:

$$
\begin{aligned}
E2^{X_{N+1}} &= \sum_{i=0}^{\infty} P(X_N = j) E(2^{X_{N+1}} | X_N = j) \\
&= \sum_{i=0}^{\infty} P(X_N = j)(2^j(1 - \frac{1}{2^j}) + \frac{1}{2^j}2^{j+1}) \\
&= \sum_{i=0}^{\infty} P(X_N = j)2^j + \sum_{i=0}^{\infty} P(X_N = j) \\
&= E2^{X_{N+1}} + 1 = (N + 1) + 1
\end{aligned}
$$

- $E2^{2X_N} = \frac{3}{2}N^2 + \frac{3}{2}N + 1$.

Analysis of Morris' algorithm cont.

Bound

It is now clear why we output our estimate of n as $\widehat{n} = 2^X - 1$ since it is an unbiased estimator of n.

- Note that $E(\widehat{n} - n)^2 = \frac{1}{2}n^2 - \frac{1}{2}n - 1 < \frac{1}{2}n^2$.
- If r.v. X has mean and variance $\mu = E(X)$ and $\sigma^2 = E[(X - \mu)^2]$, then $P(|X - \mu| > a) \leq \frac{\sigma^2}{a^2}$ or $P(|X - \mu| > aE(X)) \leq \frac{\sigma^2}{a^2 E(X)^2}$.
- $P(|\widehat{n} - n| > \epsilon n) < \frac{n^2}{2\epsilon^2 n^2} = \frac{1}{2\epsilon^2}$.
- It is not particularly meaningful since the right hand side is only better than $\frac{1}{2}$ failure probability when $\epsilon \geq 1$.

Morris+

Boosting success probability

To decrease the failure probability of Morris' basic algorithm, we instantiate s independent copies of Morris' algorithm and average their outputs.

- That is, we obtain independent estimators $\widehat{n}_1, \cdots, \widehat{n}_s$ from independent instantiations of Morris' algorithm, and our output to a query is

$$\widehat{n} = \sum_{i=1}^{s} \widehat{n}_i.$$

- $P(|\widehat{n} - n| > \epsilon n) < \frac{1}{2s\epsilon^2} < \delta$ for $s > \frac{1}{2\epsilon^2\delta}$.

Morris++

Tug of War

It turns out there is a simple technique to reduce the dependence on the failure probability δ from $\frac{1}{\delta}$ to $\log 1/\delta$.

- We run t instantiations of Morris+, each with failure probability $\frac{1}{3}$, that is $s = O(1/\epsilon^2)$.
- We then output the median estimate from all the t Morris instantiations.
- Note that the expected number of Morris+ instantiations that succeed is at least $\frac{2t}{3}$.
- For the median to be a bad estimate, at most half the Morris+ instantiations can succeed, implying the number of succeeding instantiations deviated from its expectation by at least $\frac{t}{6}$.

Morris++ cont.

Tug of War

- Define $Y_i = \begin{cases} 1, & \text{if the ith Morris+ instantiation succeeds;} \\ 0, & \text{otherwise.} \end{cases}$
- Let X_i be a sequence of independent r.v.s with $P(X_i = 1) = p_i$ and $P(X_i = 0) = 1 - p_i$. r.v. $X = \sum_{i=1}^{n} X_i$. Then, we have $P(|X - \mu| \geq \epsilon\mu) \leq 2\exp(-\mu\epsilon^2/3)$, where $\mu = \sum_{i=1}^{n} p_i$ and $\epsilon \in (0, 1)$ (Chernoff bound).
- Note that $\mu = E\sum_i Y_i = \frac{2t}{3}$. Then by the Chernoff bound,

$$P(\sum_i Y_i \leq \frac{t}{2}) \leq P(|\sum_i Y_i - \frac{2t}{3}| \geq \frac{t}{6}) = P(|\sum_i Y_i - \mu| \geq \frac{1}{4}\mu)$$
$$\leq 2\exp(-2t/3(1/4)^2/3) < 2\exp(-t/3) < \delta,$$

for $t = O(\log 1/\delta)$.

- Finally, we can get an $(\epsilon, \delta)-$approximation in complexity $O(\frac{\log 1/\delta}{\epsilon^2})$.

Item frequencies

Problem definition

We have a stream $\sigma =< a_1, a_2, \cdots, a_m >$, with each $a_i \in [n]$ and this implicitly defines a frequency vector

$$f = (f_1, f_2, \cdots, f_n).$$

Note that $\sum_{i=1}^{n} f_i = m$. In the Majority problem, our task is as follows: if $\exists j : f_k > \frac{m}{2}$, then output j, otherwise output $\perp$.

- General problem: given a parameter k, output the set $\{j : f_j > \frac{m}{k}\}$. Alternatively, given a parameter ψ, output the set $\{j : f_j > \psi m\}$.
- Frequency estimation problem: the task is to process σ to produce a data structure that can provide an estimate $\widehat{f_a}$ for the frequency f_a of a given token $a \in [n]$, then return the top-k results.
- For example, $\sigma =< a, b, a, c, c, a, b, d >,$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$. For $k = 4$, the frequent items are a, b, c. And for $\psi = 0.3$, , the frequent item is a.

Take-home messages

- Centrality
  - Degree
  - Eigenvector
  - Closeness
  - Betweenness
- Weighted graph centrality
- PageRank and HITS
  - PageRank
  - HITS
  - Co-HITS
- Graph robustness