Data Stream Mining

Péter Kiss , Assistant Professor

Eötvös Loránd University, Budapest, Hungary

September 29, 2020

Apache Flink

Definition

A data stream is an ordered (not necessarily always) and potentially infinite sequence of data points.

$$x_1, x_2, x_3, \ldots,$$

where x_i could be anything such as numbers, words, sequences, etc.

Such streams are ubiquitous and we are always around them. For example:

- Click streams
- Sensor measurements
- Satellite imaging data
- Power grid electricity distribution
- Banking/e-commerce transactions

Examples of Data Streams : YouTube comments

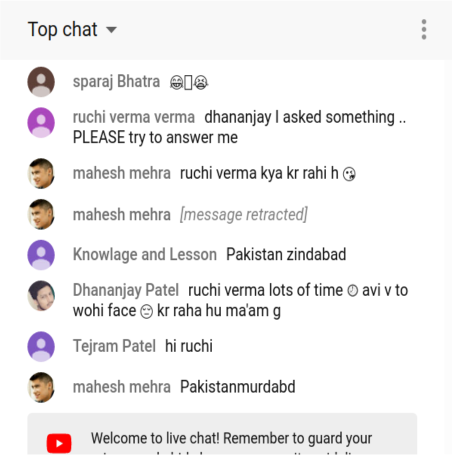


Figure: Youtube live comments

Data Stream Mining Algorithms

Various kinds of tasks:

- Classification
- Clustering
- Frequent pattern mining
- Anomaly Detection
- Change/Concept drift detection
- Mining multiple streams
- Database operations such as indexing, aggregation, and so on

Characteristics of data streams

A data stream has several distinguishing features such as:

- Unbounded Size
  - Transient (that means it lasts for only few seconds or minutes)
  - Single-pass over data
  - Only summaries can be stored
  - Real-time processing (in-memory)
- Data streams are not static
  - Incremental/decremental updates
  - Concept Drifts
- Temporal order may be important

Why can't use traditional algorithms?

- Using SQL/relational DB for storing
- K-mean for clustring?
- Naive-bayes for classification?
- Aprior algorithm for frequent pattern mining?

Relational DB and Data Streams

Table: Source:Babock et al., (2002)

Relational DB	**Data Streams**
Persistent	Transient
Multiple passes	single-pass
unlimited memory	limited memory
low update rate	stream
not real-time	real-time

Traditional Algorithms vs. DS Algorthms

	Traditional	Data Streams
passes	multiple	single
processing time	unlimited	limited
realt-time	no	yes
memory	disk	main memory
results	accurate	approximate
distributed	no	yes

Research Issues in Data Stream Management Systems (DSMS)

- Approximate query processing
- Sliding window query processing
- Sampling

Basic Streaming Methods

Most of the streaming methods are approximate. (Why?) So, need good approximation techniques such as:

- ϵ-approximation: answer is within some small fraction ϵ of the true answer.
- (ϵ, δ)-approximation: the answer is within $1 \pm \epsilon$ of the true answer with probability $1 - \delta$

Checking if a user-id is present

Can you guess how gmail checks if a user-id is available?

Checking if a user-id is present

Can you guess how gmail checks if a user-id is available?
Ans: **Bloom Filter**

Checking if a user-id is present

Can you guess how gmail checks if a user-id is available?
Ans: **Bloom Filter**
The key idea: Maintain the following:

- A bit-array of length n.
- A number of hash functions $h_1(), h_2(), \ldots, h_k()$.
- A set S of m key values.

Bloom Filter

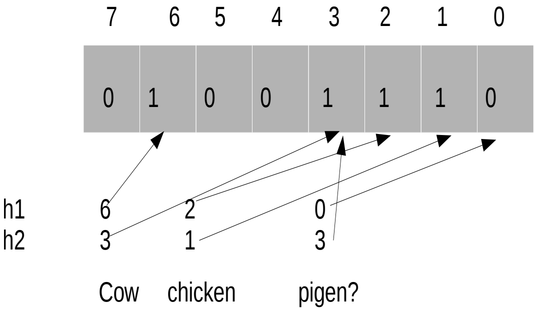


Figure: Hashing items in a bit-array. h_1 and h_2 are hash functions.

Bloom filter can answer if an item passes through it or not. That is, it gives **guaranteed answer** if a user id is available and probabilistic answer if a user-id is NOT available by measuring the collision. If at least one bit is not set, user-id is available and if all bits are set, it means either user-id *may* be available.

Limitations of bloom filter

1. Can't remove items. (Why?)

Limitations of bloom filter

1. Can't remove items. (Why?)
2. Use **Count-Bloom filter**
3. Size of the bloom filter is very important. Smaller bloom filter, larger false positive (FP) and vice versa.
4. Number of hash functions? Larger number of hash functions, quicker the bloom filter fills as well as slow filter. Too few hash functions, too many FPs unless all are *good* hash functions.

Contd...

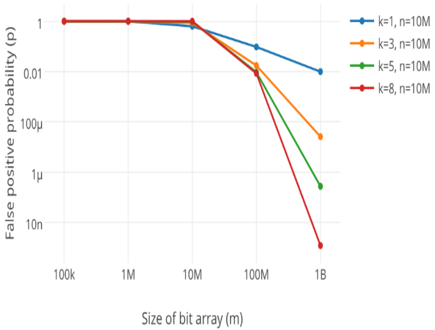


Figure: Number of hash functions vs FPR. Source: https://www.semantics3.com/blog/use-the-bloom-filter-luke-b59fd0839fc4/

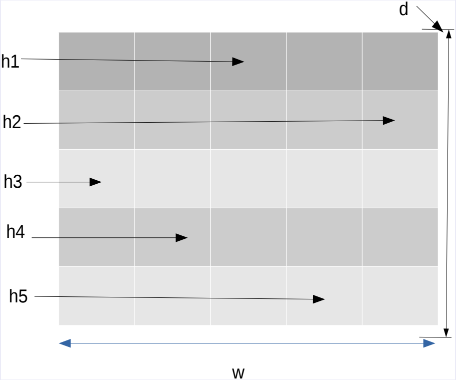
Example 1: Counting the frequency of unique items in a stream

The Problem: Count frequency of A in the stream: A, B, C, A, A, C,...?

Example 1: Counting the frequency of unique items in a stream

The Problem: Count frequency of A in the stream: A, B, C, A, A, C,...?

Count-Min Sketch



(c)

	h1	h2	h3	h4	h5
A	0	1	3	4	2
B	1	0	2	4	1
C	0	3	2	1	4

(d)

Example 2: Count unique values in a stream

Similar to numpy np.unique() function. Suppose the domain of the random variable is $\{0, 1, \ldots, M\}$ and stream looks like:

$$0, 1, 0, 0, 0, 1, 1, 2, 3, \ldots$$

Here $M = 3$. Trivial if you have space linear in M (why?). Can you do better? e.g. using space only $\log(M)$. Answer: Use Flajolet and Martin Algorithm (1985).

Flajolet and Martin Algortihm

Algorithm 1: Flajolet and Martin (FM) Algorthm

Input: stream M

Output: Cardinality of M

initialization: BITMAP OF L bits initialized to 0.;

for *each x in M* **do**

1. Calculate hash function $h(x)$.;
2. get binary representation of hash output, call it $bin(h(x))$.;
3. Calculate the index i such that $i = \rho(bin(h(x)))$, where $\rho()$ is such that it outputs the **position** of the least-significant set bit,i.e., position of 1.;
4. set BITMAP[i] $=$ 1.;

end

Let R denote the largest index i such that BITMAP[i] $=$ 1;

Cardinality of M is $2^R/\phi$ where $\phi \approx 0.77351$;

Intuition

The basic idea behind FM algorithm is that of using a hash function that maps the strings in the stream to uniformly generated random integers in the range $[0, \ldots, 2^L - 1]$. Thus we expect that:

- 1/2 of the numbers will have their binary representation end in 0 (divisible by 2)
- 1/4 of the numbers will have their binary representation end in 00 (divisible by 4)
- 1/8 of the numbers will have their binary representation end in 00 (divisible by 8)
- In general, $1/2^R$ of the numbers will have their binary representation end in 0^R

Then the number of unique strings will be approx. 2^R.(because using n bits, we can represent 2^n integers)

Working Example:

Assume stream M=$\{\ldots,1,1,2,3,1,4,2,\ldots\}$ and the hash function $h(x)=(2x+1)\%5$.
Then,

$$h(M)=\{3,3,0,2,3,4,0\}$$

and

$$\rho(h(M))=\{0,0,0,1,0,2,0\}$$

and BITMAP looks like

$$BITMAP=0|0|0|1|1|1$$

so we see the largest integer has index $i=2$. So $R=2$ and unique integers are $2^2=4$ approx.
HW: Try finding unique numbers in $M=\{5,10,15,20,\ldots$ with any hash function.
Extensions:$\log\log$ and $Hyper\log\log$

Bibliography I