Mining Data Streams

Barna Saha

January 21, 2016

Motivation

- Data arrives in a stream or streams

Motivation

- Data arrives in a stream or streams
- If not processed immediately or stored, then data is lost for ever.

Motivation

- Data arrives in a stream or streams
- If not processed immediately or stored, then data is lost for ever.
- Data arrives so rapidly that it is not feasible to store it all in active storage.

Motivation

- Data arrives in a stream or streams
- If not processed immediately or stored, then data is lost for ever.
- Data arrives so rapidly that it is not feasible to store it all in active storage.
- We need new algorithmic paradigm to handle data streams.

What will be covered in this part of the course?

- The algorithms for processing streams involve summarization of streams in some way.

What will be covered in this part of the course?

- The algorithms for processing streams involve summarization of streams in some way.
- How to create summary via **sampling**?

What will be covered in this part of the course?

- The algorithms for processing streams involve summarization of streams in some way.
- How to create summary via **sampling**?
- How can we **filter** data streams to eliminate undesirable elements.

What will be covered in this part of the course?

- The algorithms for processing streams involve summarization of streams in some way.
- How to create summary via **sampling**?
- How can we **filter** data streams to eliminate undesirable elements.
- How can we compute popular statistics like distinct elements, top-k frequent elements, frequency moments in a data stream?

Example of Data Streams

Sensor Data.

- A temperature sensor in the ocean sending reading every one hour–Not an interesting stream since the data rate is low. It will not stress modern technology, and the entire stream can be kept in main memory.

Example of Data Streams

Sensor Data.

- A temperature sensor in the ocean sending reading every one hour–Not an interesting stream since the data rate is low. It will not stress modern technology, and the entire stream can be kept in main memory.
- Suppose the sensor senses surface height information which changes rapidly. Now the sensor is sending data back every tenth of a second. If it sends a 4-byte real number each time, then it produces $4 * 10 * 3600 * 24 = 3456000 bytes = 3.5 Megabyte/$per day.–Still ok.

Example of Data Streams

Sensor Data.

- A temperature sensor in the ocean sending reading every one hour–Not an interesting stream since the data rate is low. It will not stress modern technology, and the entire stream can be kept in main memory.
- Suppose the sensor senses surface height information which changes rapidly. Now the sensor is sending data back every tenth of a second. If it sends a 4-byte real number each time, then it produces $4 * 10 * 3600 * 24 = 3456000 bytes = 3.5 Megabyte/$per day.–Still ok.
- We may need to employ a million sensors to learn about ocean behavior.—3.5 terabytes of data per day, million of data arriving every tenth of a second.

Example of Data Streams

Image Data.

- Satellites often send down to earth streams consisting of many terabytes of images per day.

Example of Data Streams

Image Data.

- Satellites often send down to earth streams consisting of many terabytes of images per day.
- Surveilance cameras may produce images at every second. London is said to have six millions of such cameras.

Example of Data Streams

Internet and Web Traffic.

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its outputs: denial or service attacks.

Example of Data Streams

Internet and Web Traffic.

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its outputs: denial or service attacks.
- Google receives several hundred million search queries per day.

Example of Data Streams

Internet and Web Traffic.

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its outputs: denial or service attacks.
- Google receives several hundred million search queries per day.
- Yahoo! accepts billions of clicks per day on its various sites.

Example of Data Streams

Internet and Web Traffic.

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its outputs: denial or service attacks.
- Google receives several hundred million search queries per day.
- Yahoo! accepts billions of clicks per day on its various sites.
- Many interesting things can be learnt from these streams. An increase in queries like "sore throat" may help to track the spread of viruses. A sudden increase in the click rate for a link could indicate some news connected to that page etc.

Example of Stream Queries

Example

The stream produced by the ocean-surface-temperature sensor might have a standing query to output an elect whenever the temperature exceeds 25 degrees centigrade.

Example

There may be a standing query that each time a new reading arrives, produces the average of the 100 most recent readings.

Example

There may be a standing query that returns the maximum temperature ever recorded.

Example

There may be a standing query that returns the average temperature over all time.

Example of Stream Queries

Example

Web sites often like to report the number of unique users over the past month/past year etc.

Example

Number of unique products sold by Target across the country.

Example

Denial of Service (DoS) attack: attempts to make a machine or network resource unavailable to its intended users. Attackers use forged senders IP addresses, and send large number of IP packets to the victim destination. In order to prevent such attacks, we would like to detect the **heavy-hitters** aka those IP addresses that have received extremely high traffics in the last hour or so.

Which industries are deploying stream processors?

- Smart Cities - real-time traffic analytics, congestion prediction and travel time apps.
- Oil & Gas - real-time analytics and automated actions to avert potential equipment failures.
- Security intelligence for fraud detection and cybersecurity alerts. For example, detecting Smart Grid consumption issues, and SIM card misuse.
- Industrial automation, offering real-time analytics and predictive actions for patterns of manufacturing plant issues and quality problems.
- For Telecoms, real-time call rating, fraud detection and QoS monitoring from CDR (call detail record) and network performance data.
- Cloud infrastructure and web clickstream analysis for IT Operations.

Few Stream Processing Systems

- SQLstream `http://www.sqlstream.com/blaze/`: use standards-compliant SQL for querying live data streams
- Spark Streaming: to build streaming applications in Apache Spark. Apache Spark is a general framework for large-scale data processing that supports concepts such as MapReduce, stream processing, graph processing or machine learning.
- IBM InfoSphere Streams: IBM's flagship product for stream processing.
- Apache Storm: an open source framework that provides massively scalable event collection.

Developing Streaming Algorithm

- The main hardle is the space. To handle rapid rate of stream, external algorithms are not feasible.
- Often, it is much more efficient to get an approximate answer to a problem than an exact solution.
- Often, the algorithm uses randomization, like hashing and sampling.

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- A search engine receives a stream of queries, and it would like to study the behavior of typical users.
- Each stream consists of tuples $(user, query, time)$
- Question: "What fraction of the typical user's queries were repeated (occurred more than once) over the past month?"
- Space constraint: we can only keep $1/10$th of the stream elements.

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- Space constraint: we can only keep 1/10th of the stream elements.
- Sample 1/10th of the queries of each user!!!

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- Space constraint: we can only keep 1/10th of the stream elements.
- Sample 1/10th of the queries of each user!!!
  - Suppose a user has s queries that occurred only once, and d queries that occurred twice.

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- Space constraint: we can only keep 1/10th of the stream elements.
- Sample 1/10th of the queries of each user!!!
  - Suppose a user has s queries that occurred only once, and d queries that occurred twice.
  - Expected number of queries that occur twice in the sample is $\frac{d}{100}$

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- Space constraint: we can only keep 1/10th of the stream elements.
- Sample 1/10th of the queries of each user!!!
  - Suppose a user has s queries that occurred only once, and d queries that occurred twice.
  - Expected number of queries that occur twice in the sample is $\frac{d}{100}$
  - Expected number of queries that occur only once in the sample is $\frac{s}{10} + \frac{18d}{100}$

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- Space constraint: we can only keep 1/10th of the stream elements.
- Sample 1/10th of the queries of each user!!!
  - Suppose a user has s queries that occurred only once, and d queries that occurred twice.
  - Expected number of queries that occur twice in the sample is $\frac{d}{100}$
  - Expected number of queries that occur only once in the sample is $\frac{s}{10} + \frac{18d}{100}$
  - Correct answer: $\frac{d}{s+d}$

A Motivating Example

We want to select a subset of a stream so that we can ask queries about the selected subset, and have the answer representative of the entire stream.

- Space constraint: we can only keep 1/10th of the stream elements.
- Sample 1/10th of the queries of each user!!!
  - Suppose a user has s queries that occurred only once, and d queries that occurred twice.
  - Expected number of queries that occur twice in the sample is $\frac{d}{100}$
  - Expected number of queries that occur only once in the sample is $\frac{s}{10} + \frac{18d}{100}$
  - Correct answer: $\frac{d}{s+d}$
  - Answer found: $\frac{d}{10s+19d}$

Obtaining a Representative Sample

Instead of keeping $\frac{1}{10}$th of queries of each user do the following:

- Sample 1/10th of the users and for each user keep the entire list of queries.
- To sample 1/10th of the users, use a random hash function that hashes every user to a bucket of size 10, and keep the query list for those which are hashed to the first bucket (say).

Obtaining a Representative Sample

- As the number of users grow, keeping a $\frac{1}{10}$ of users may become prohibitive.
- We can only afford to have a sample that fits in the main memory.

Reservoir Sampling

You have a stream of items of large and unknown length that we can only iterate over once. Create an algorithm that randomly chooses an item from this stream such that each item is equally likely to be selected.

Reservoir Sampling

You have a stream of items of large and unknown length that we can only iterate over once. Create an algorithm that randomly chooses an item from this stream such that each item is equally likely to be selected.

- Upon seeing the **first** element—keep it.
  - The first element is chosen with probability 1.
- Upon seeing the **second** element—select the second element with probability $\frac{1}{2}$. If the second element is selected discard the first element.
  - The probability that the second item is sampled$=\frac{1}{2}$
  - The probability that the first item is sampled$=\frac{1}{2}$

Reservoir Sampling

You have a stream of items of large and unknown length that we can only iterate over once. Create an algorithm that randomly chooses an item from this stream such that each item is equally likely to be selected.

- Upon seeing the **third** element—select it with probability $\frac{1}{3}$, if selected then discard the element that was previously selected.
  - The probability that the third item is sampled$=\frac{1}{3}$.
  - The probability that the second item is sampled$=\frac{2}{3} * \frac{1}{2} = \frac{1}{3}$
  - The probability that the first item is sampled$=\frac{2}{3} * \frac{1}{2} = \frac{1}{3}$
- Upon seeing the **fourth** element—select it with probability $\frac{1}{4}$, if selected then discard the element that was previously selected.
  - The probability that the fourth item is sampled$=\frac{1}{4}$.
  - The probability that the third item is sampled$=\frac{3}{4} * \frac{1}{3} = \frac{1}{4}$
  - The probability that the third item is sampled$=\frac{3}{4} * \frac{1}{3} = \frac{1}{4}$ The probability that the third item is sampled$=\frac{3}{4} * \frac{1}{3} = \frac{1}{4}$

Reservoir Sampling

You have a stream of items of large and unknown length that we can only iterate over once. Create an algorithm that randomly chooses an item from this stream such that each item is equally likely to be selected.

- Can you generalize the algorithm to any i?

Reservoir Sampling

You have a stream of items of large and unknown length that we can only iterate over once. Create an algorithm that randomly chooses an item from this stream such that each item is equally likely to be selected.

- Can you generalize the algorithm to any i?
- Upon seeing the ith item—select it with probability $\frac{1}{i}$, if selected, discard the element that was previously selected.
  - The probability that the ith item is sampled$=\frac{1}{i}$.
  - The probability that the jth $j < i$ item is sampled$=\frac{i-1}{i} * \frac{1}{i-1} = \frac{1}{i}$

Reservoir Sampling

Can you generalize it to the case when the reservoir has size s ?

Reservoir Sampling of Size s

- As long as the stream has size smaller than s, main the entire stream in the reservoir.
- Upon seeing the $s+1$th element, select it with probability $\frac{s}{s+1}$, if selected discard one item from the current reservoir with probability $\frac{1}{s}$.
  - The probability that the $s+1$th item is sampled$=\frac{s}{s+1}$
  - The probability that the jth $j \leq s$th item is sampled$=\frac{1}{s+1} + \frac{s}{s+1} * \frac{s-1}{s} = \frac{1}{s+1} + \frac{s-1}{s+1} = \frac{s}{s+1}$

Reservoir Sampling of Size s

- As long as the stream has size smaller than s, main the entire stream in the reservoir.
- Upon seeing the $s+1$th element, select it with probability $\frac{s}{s+1}$, if selected discard one item from the current reservoir with probability $\frac{1}{s}$.
  - The probability that the $s+1$th item is sampled$=\frac{s}{s+1}$
  - The probability that the jth $j \leq s$th item is sampled$=\frac{1}{s+1}+\frac{s}{s+1}*\frac{s-1}{s}=\frac{1}{s+1}+\frac{s-1}{s+1}=\frac{s}{s+1}$
- **EXERCISE.** Complete the analysis.

Reservoir Sampling

According to Cloudera, they provide the world’s fastest, easiest, and most secure Hadoop platform

cloudera

Cloudera Engineering Blog

Best practices, how-tos, use cases, and internals from Cloudera Engineering and the community

Algorithms Every Data Scientist Should Know: Reservoir Sampling

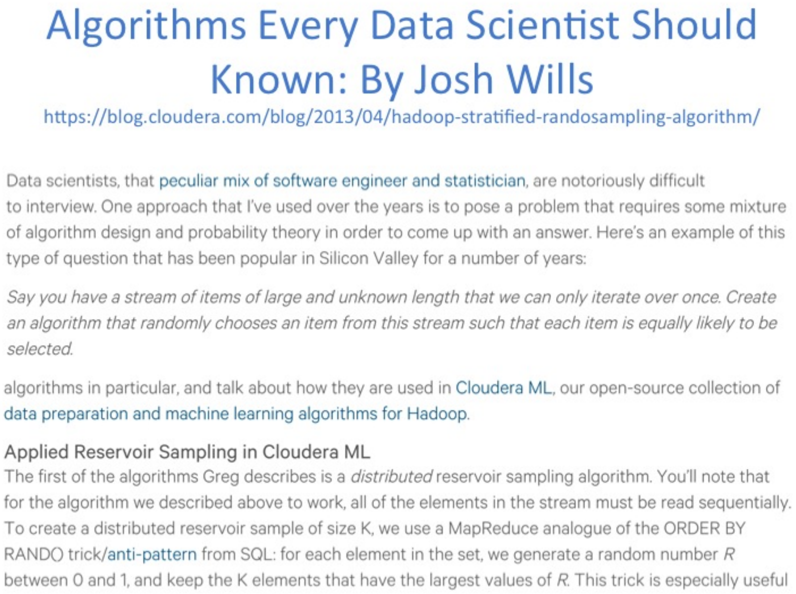
April 23, 2013 | By Josh Wills (@josh_wills) (@josh_wills) | 3 Comments

Tweets

Cloudera Engineering @ClouderaEng

Our bad. The Hive meetup at Clo
on Thurs 1/21 (not Weds)
twitter.com/thejasn/status...

Reservoir Sampling



Algorithms Every Data Scientist Should Known: By Josh Wills

https://blog.cloudera.com/blog/2013/04/hadoop-stratified-randosampling-algorithm/

Data scientists, that peculiar mix of software engineer and statistician, are notoriously difficult to interview. One approach that I've used over the years is to pose a problem that requires some mixture of algorithm design and probability theory in order to come up with an answer. Here's an example of this type of question that has been popular in Silicon Valley for a number of years:

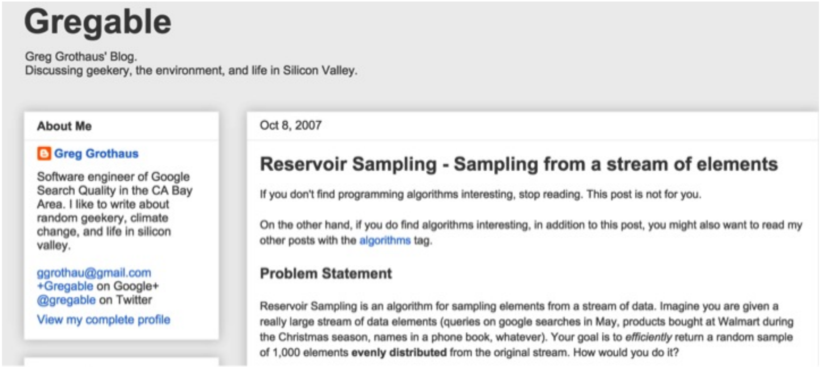
Say you have a stream of items of large and unknown length that we can only iterate over once. Create an algorithm that randomly chooses an item from this stream such that each item is equally likely to be selected.

algorithms in particular, and talk about how they are used in Cloudera ML, our open-source collection of data preparation and machine learning algorithms for Hadoop.

Applied Reservoir Sampling in Cloudera ML

The first of the algorithms Greg describes is a *distributed* reservoir sampling algorithm. You'll note that for the algorithm we described above to work, all of the elements in the stream must be read sequentially. To create a distributed reservoir sample of size K, we use a MapReduce analogue of the ORDER BY RANDO trick/anti-pattern from SQL: for each element in the set, we generate a random number *R* between 0 and 1, and keep the K elements that have the largest values of *R*. This trick is especially useful

Reservoir Sampling



Gregable

Greg Grothaus' Blog.
Discussing geekery, the environment, and life in Silicon Valley.

About Me

Greg Grothaus

Software engineer of Google Search Quality in the CA Bay Area. I like to write about random geekery, climate change, and life in silicon valley.

ggrothau@gmail.com
+Gregable on Google+
@gregable on Twitter
View my complete profile

Oct 8, 2007

Reservoir Sampling - Sampling from a stream of elements

If you don't find programming algorithms interesting, stop reading. This post is not for you.

On the other hand, if you do find algorithms interesting, in addition to this post, you might also want to read my other posts with the algorithms tag.

Problem Statement

Reservoir Sampling is an algorithm for sampling elements from a stream of data. Imagine you are given a really large stream of data elements (queries on google searches in May, products bought at Walmart during the Christmas season, names in a phone book, whatever). Your goal is to *efficiently* return a random sample of 1,000 elements **evenly distributed** from the original stream. How would you do it?

Birds of a Feather

If you are one of the handful of people interested in Reservoir Sampling and advanced software algorithms like this, you are the type of person I'd like to see working with me at Google. If you send me your resume (ggrothau@gmail.com), I can make sure it gets in front of the right recruiters and watch to make sure that it doesn't get lost in the pile that we get every day. **Update**: Despite the fact that this post was published in 2008, this offer still stands today.

- http://gregable.com/2007/10/reservoir-sampling.html

Further Reading

- Priority Sampling: Elements are weighted. Keep a sample of size k such that each subset sum queries can be answered.
  - Reference: Nick Duffield, Carsten Lund, and Mikkel Thorup. Priority sampling for estimation of arbitrary subset sums. Journal of the ACM (JACM), 54(6):32, 2007.