Streaming (realtime) and big geospatial data

GEOG 384

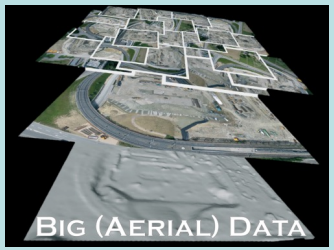
RE Sieber, McGill University

Big Data

datasets that are so large, fast and various that traditional data processing is inadequate. Challenges include data capture, curation, search, storage, transfer, visualization, and abductive analysis. Also privacy (see fb & Cambridge Analytica).

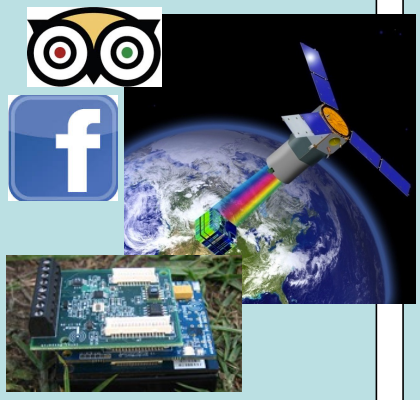
4 V's of Geospatial Big Data

Volume



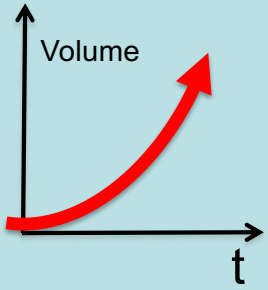
Large imagery data, due to the increasing spatial, spectral and temporal resolutions of sensing systems

Variety



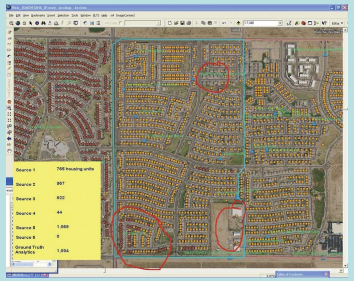
Heterogeneous resolutions; Different data formats; images; text

Velocity



Increasing speed of incoming geospatial data

Veracity



Ground truthing of big data

Xing 2016

Example

- A source of big data: 330 million monthly active **users** > 30 million tweets/hour
- A fundamental problem: Too much information to store or transmit
- A solution: process data as it arrives, one pass, small space.
- There's lots of geography in Twitter

Streaming Data

real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items. It is impossible to control the order in which items arrive, nor is it feasible to locally store a stream in its entirety.

Golab & Oszu (2003)

Streaming is not the same as big data. It doesn’t have to be big

Streaming Characteristics

Continuous (& appears to be infinite)

Bandwidth dependent

Generally, a buffer/cache requirement

Delay sensitive

No retransmission (you get what you get)

Differences in downloading/ data handling

Traditional (e.g., remote sensed imagery)

End-user must wait until the content is fully downloaded.

If there is an interruption during download of data, the whole file is likely corrupted.

For downloads, web server+app server or ftp site.

You work with 100% of dataset (and do statistical inferences based on all the data)

Streaming (e.g., geolocated tweets)

No need to wait until the file is fully downloaded

Use the file (aka tuples of data) as it comes in

Need high data rate and consume significant bandwidth

May require the same amount of storage space

Application: Stock Monitoring

Notify me when the price of ZM (Zoom) is above $150, and the first CSCO (Cisco, makers of Webex) price afterwards is below $40.

Notify me when some stock goes up by at least 5% from one transaction to the next.

Notify me when the price of any stock increases for >30 min.

Notify me when the difference between the current price of a stock and its 10-day moving average is greater than some threshold value

Notify me when a stock drops by 10% that day

Challenges over all

- Classical algorithms do not scale up to data stream
  - Most need entire data set for analysis
  - Random access (or multiple passes) to the data
- Difficult to compute answers accurately with limited memory
- Considerable noise (bad sensors, redundancies like retweets, outliers)
- Trying to determine what data is stale and what should be kept
- Remember you must store some of this data and provide analysis space

Structure of Stream

In a potentially infinite sequence of elements…

One element: structured information, i.e., tuple or object

Same structure for all elements in a stream

Timestamp

- << explicit >> (date field in data)
- << implicit >> (timestamp given when items arrive)

Content

Geography

Challenges of Data Handling

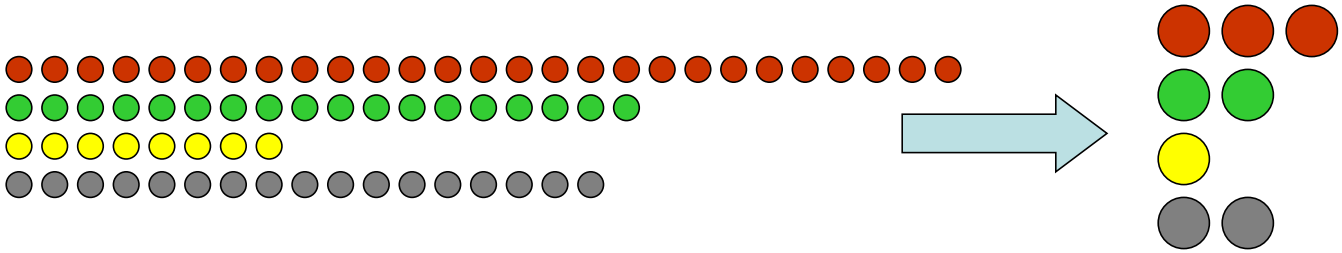
- How many tuples or how long a time period?
- What if stream is skewed (e.g., uneven in time)?
- What if stream is clustered by a single poster, sensor, area?
- Frequency problems
  - Could be lots of redundancies (e.g., retweets)
  - If few items are frequent then a long tail of infrequent items
  - Skews are prevalent in word frequency, paper citations, and city sizes (follow the leader)
- Possibilities: assigning weights, ratings
  - Assigning time delay function: weight to each item as a function of its age

One solution: Sampling

But what are you sampling? How often something is mentioned? What is trending? What is being said? For the latter →

As with any sample, it needs to represent the whole dataset. IOW, it retains the properties of the whole data set. Important for drawing the right conclusions from the data

Also have a limited amount of time to do this selection & don't want to waste storage space in this method. Both methods have you select an n.



Sampling From a Data Stream

Fundamental prob: sample *m* items uniformly from stream

- Useful: approximate possible costly computation on small sample

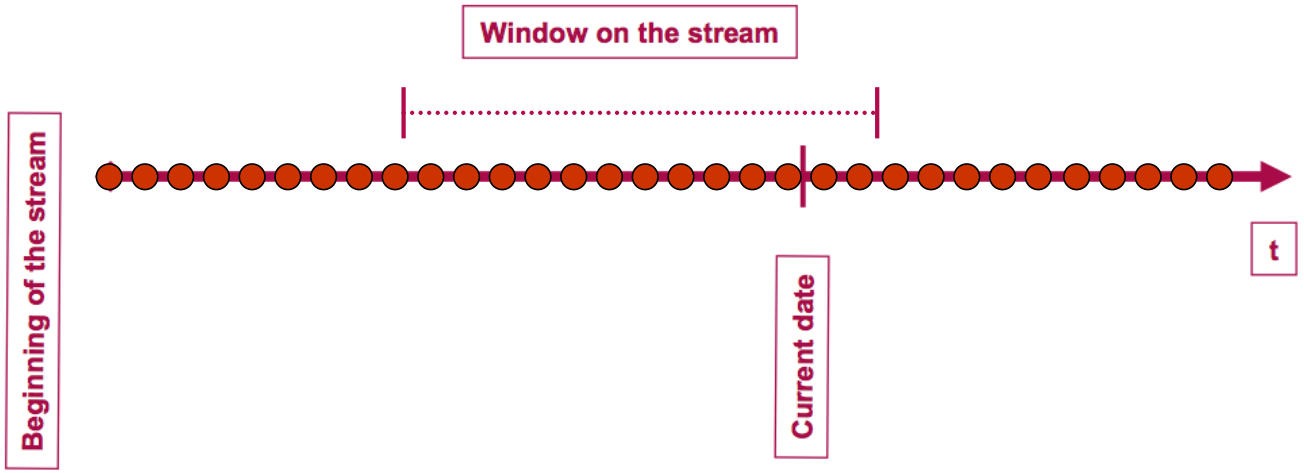
Challenge: don't know length of stream

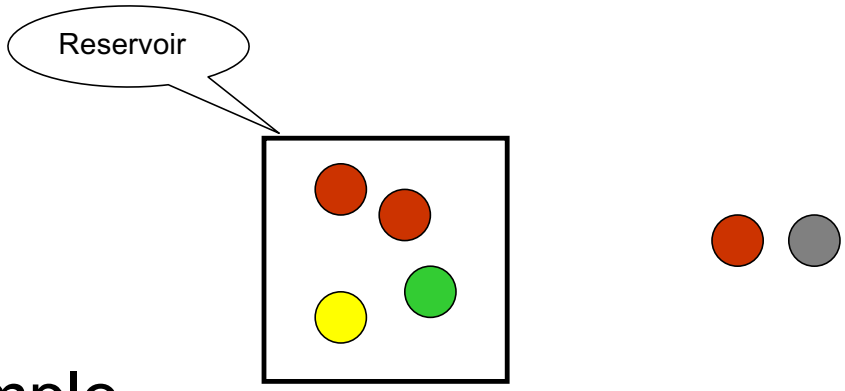
- So when/how often to sample? Width, velocity

Three solutions:

- Sliding window
- Reservoir sampling
- Min-wise sampling

Sliding Windows





Simple Reservoir Sampling

Sample first *m* items

Choose to sample item *i* (*i*>*m*)

If sampled, randomly replace a previously sampled item

Min-wise Sampling

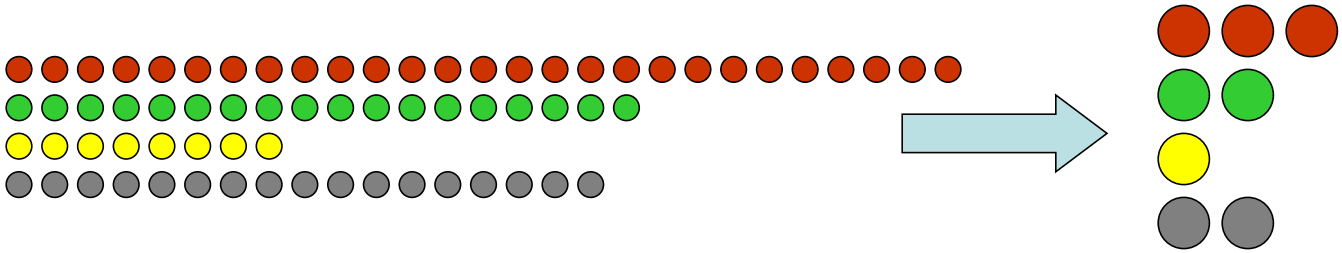
For each item, pick a random fraction between 0 and 1

Collect item(s) with the smallest random tag

Each item has same chance (probability) of smallest tag

Can do dynamically or after you collect the data

Distributed Streams: Sensors



Distributed Streams from Sensors;
Different rows above represent
different, distributed data streams

(think multiple Twitters)

Implications of integrating distributed sensors: What is the implication of the traffic flow monitoring to the air quality monitoring?

Conclusion

- Definitions
- Fundamental components and differences from traditional data
- Structure of a stream tuple (e.g., tweet)
- Part of rhetoric of big data, Internet of Things, Smart Cities
- Challenges in caching, interoperability, lags
- Types of Sampling
- Wide variety of geosensors