DATA STREAMS MINING
Mining Data Streams

From Data-Streams Management System Queries to Knowledge Discovery from continuous and fast-evolving Data Records.

Hammad Haleem
Xavier Plantaz
APPLICATIONS

Example: Freeboard.io - Dashboards For the Internet Of Things - https://freeboard.io/

Sensors
Weather
Stock Exchange
Self Driving Cars
Trends
Tweets
 Logs
Articles/News
...

Wikipedia Edits
ATM Transactions
Chats
Television
Seisms
Music similarities
CO2 Level
Car Tracking
...
Background
DATA STREAM MANAGEMENT SYSTEM: Recap

- KWAN, Victor Wing-chuen, LEE, Dexter - Data Streams
- TAN, Pin Siang - Time Series DB
- LI, Qingjie, YEUNG, Chun Kit - Big Data Computation
- LINGYS, Justinas, SIDDIQUE, Farhad Bin - Knowledge Discovery in DB

Example: Cubism.js (Open Sourced by Square) on random Data - https://square.github.io/cubism/
WHAT IS STREAM MINING?

Data Stream Mining is the process of extracting knowledge structures from continuous, rapid data records.

A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities.
TYPICAL DATA STREAMING ARCHITECTURE

- **Stream Processor**
  - Input Streams
  - Output Streams
  - Standing Queries
  - Ad-hoc queries
  - Limited Working Storage
  - Archival Storage
N users

What fraction of the typical user’s queries were repeated over the past month?
SAMPLING DATA IN A STREAM

MOTIVATING EXAMPLE

What fraction of the typical user’s queries were repeated over the past month?

Let’s say that we want to process 1/10th of the data
What fraction of the typical user's queries were repeated over the past month?
Let's say that we want to process 1/10th of the data

Naive approach

```scala
scala> val r = scala.util.Random
scala> r.nextFloat
res1: Float = 0.50317204

scala> r.nextFloat < 0.1
res2: true
```
What fraction of the typical user's queries were repeated over the past month?
Let's say that we want to process 1/10th of the data
What fraction of the typical user’s queries were repeated over the past month?

Let’s say that we want to process 1/10th of the data.

New Request!

hash(IP) corresponds to bucket 0? Yes Keep the query
How to take a sample of size $\frac{a}{b}$?
Key Challenges with data stream mining

- Infinite length
- Concept-drift
- Concept-evolution
- Feature Evolution
Infinite Stream length

1. Impractical to store and use all historical data, that would require infinite storage

1. In many data mining situations, we do not know the entire data set in advance. Stream Management is important when the input rate is controlled externally
   1. Google queries
   2. Twitter or Facebook status updates

1. We can think of the data as infinite and non-stationary (the distribution changes over time)

1. And infinite running time to build a machine learning model
Concept-Drift

Current hyperplane

Previous hyperplane

A data chunk

Positive instance \( \circ \)
Negative instance \( \bullet \)
Instances victim of concept-drift \( \circ \)
1. Concept-evolution occurs when a new class arrives in the stream.
2. In this example, we again see a data chunk having two dimensional data points.
3. There are two classes here, + and -. Suppose we train a rule-based classifier using this chunk.
4. Suppose a new class x arrives in the stream in the next chunk.
5. If we use the same classification rules, all novel class instances will be mis-classified as either + or -.

**Classification rules:**

R1. if \((x > x_1 \text{ and } y < y_2)\) or \((x < x_1 \text{ and } y < y_1)\) then class = +

R2. if \((x > x_1 \text{ and } y > y_2)\) or \((x < x_1 \text{ and } y > y_1)\) then class = -

Existing classification models misclassify novel class instances.
Dynamic Features

- **Reasons for new features being added**
  - **Infinite data stream**
    - Normally, global feature set is unknown
    - New features may appear
  - **Concept drift**
    - As concept drifting, new features may appear
  - **Concept evolution**
    - New type of class normally holds new set of features
    - Different chunks may have different feature sets
Dynamic Features

Example: A classification problem on input series of words.

Existing classification models need complete fixed features and apply to all the chunks. Global features are difficult to predict. One solution is using all English words and generate vector. Dimension of the vector will be too high.
HANDS-ON!

This section describes a problem and then we illustrate streaming based solution for the problem.

**Problem:** We want to have a real time calculation of stock prices for 3 stocks and correlate these prices with twitter sentiment for these stocks.

**Solution:**
1. Create streams for both Stock data and twitter data.
2. Calculate a moving average of each stock price over a small window.
3. Calculate the sentiment value of stock over the same window.
4. If positive sentiment, prediction is buy, else prediction is sell.
Workflow

1. Compute moving average price for stock on 20, 60 second windows.
2. Compute average polarity in 20, 60 second windows.
3. Use a simple logic to either buy or sell
4. Create an output stream

Frameworks
1. All streams are Kafka streams, Apache Kafka is an open-source stream processing platform. It provides with a unified, high-throughput, low-latency platform for handling real-time data feeds.
2. Processing is done in Spark, which is a general purpose cluster computing framework.

Algorithm
1. Compute moving average price for stock on 20, 60 second windows.
2. Compute average polarity in 20, 60 second windows.
3. Use a simple logic to either buy or sell
4. Create an output stream
Stream Mining Algorithms

1. Stream Frequent pattern mining
1. Stream Classification
1. Stream Cluster analysis
Frequent Patterns for Stream Data

- Frequent pattern mining is valuable in stream applications

- Mining precise freq. patterns in stream data: unrealistic
  - Most frequent pattern algorithms are approximations, some store patterns in a compressed form, such as FPtree

- Common methods for frequent pattern mining on streams:
  - Approximate frequent patterns
  - Mining evolution of frequent patterns
  - Space-saving computation of frequent and top-k elements
Mining Approximate Frequent Patterns

- **Approximate answers** are often sufficient (e.g., trend/pattern analysis)
  - Example: A router is interested in all flows:
    - whose frequency is at least 1% (\( \sigma \)) of the entire traffic stream seen so far
    - and feels that 1/10 of \( \sigma \) (\( \epsilon = 0.1\% \)) error is comfortable

- How to mine frequent patterns with **good approximation**?
  - Lossy Counting Algorithm (Manku & Motwani, VLDB’02)
    - **Major ideas**: not tracing items until it becomes frequent
    - **Adv**: guaranteed error bound
    - **Disadv**: keep a large set of traces
Lossy Counting for **Frequent Single Items**

Divide stream into ‘buckets’ (bucket size is $1/\varepsilon = 1000$)
First Bucket of Stream

Empty (summary) + 

At bucket boundary, decrease all counters by 1
Next Bucket of Stream

At bucket boundary, decrease all counters by 1
Lossy Counting For Frequent Itemsets

Divide Stream into ‘Buckets’ as for frequent items.
But fill as many buckets as possible in main memory one time.

Bucket 1
Bucket 2
Bucket 3

If we put 3 buckets of data into main memory one time, then decrease each frequency count by 3.
Itemset (■□) is deleted.
That’s why we choose a large number of buckets – delete more
Pruning Itemsets – Apriori Rule

If we find itemset ( ) is not frequent itemset, then we needn’t consider its superset.
Summary of Lossy Counting

**Strength**
- A simple idea
- Can be extended to frequent itemsets
- A space-saving method for stream frequent item mining

**Weakness**
- Space bound is not good
- For frequent itemsets, they do scan each record many times
- The output is based on all previous data. But sometimes, we are only interested in recent data
Classification Methods

- **Classification**: Model construction based on training sets
- Typical classification methods
  - Decision tree induction
  - Bayesian classification
  - Rule-based classification
  - Neural network approach
  - Support Vector Machines (SVM)
  - Associative classification
  - K-Nearest neighbor approach
Classification for Dynamic Data Streams

- One of the most commonly used methods:
  - Decision tree induction for stream data classification
    - Is decision-tree good for modeling fast changing data, e.g., stock market analysis?

- Other stream classification methods
  - Instead of decision-trees, consider other models
  - We can train an offline model using previous data on batch mode, rather than online mode
    - Naïve Bayesian
    - Ensemble
    - K-nearest neighbors
    - Classifying skewed stream data
Build Very Fast Decision Trees Based on Hoeffding Inequality

- Hoeffding's inequality:
  - A result in probability theory that gives an upper bound on the probability for the sum of random variables to deviate from its expected value

- Based on Hoeffding Bound principle, classifying different samples leads to the same model with high probability — can use a small set of samples

- Hoeffding Bound (Additive Chernoff Bound)
  
  Given:
  
  \[ r: \text{random variable}, \]
  \[ R: \text{range of } r, \]
  \[ N: \text{# independent observations} \]

  True mean of \( r \) is at least \( r_{\text{avg}} - \varepsilon \), with probability \( 1 - \delta \) (where \( \delta \) is user-specified)

\[
\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}}
\]

Decision-Tree Induction with Data Streams

Packets > 10
yes

Data Stream

Protocol = http

Bytes > 60K
yes

Packets > 10
yes

Protocol = ftp

Protocol = http

yes

no

no

Ack. From Gehrke’s SIGMOD tutorial slides
Hoeffding Tree: Strengths and Weaknesses

**Strengths**
- Scales better than traditional methods
  - Sublinear with sampling
  - Very small memory utilization
- Incremental
  - Make class predictions in parallel
  - New examples are added as they come

**Weakness**
- Could spend a lot of time with ties
- Memory used with tree expansion
- Number of candidate attributes
CVFDT (Concept-adapting Very Fast Decision Trees)

- Concept Drift
  - Time-changing data streams
  - Incorporate new and eliminate old

- CVFDT
  - Increments count with new example
  - Decrement old example
    - Sliding window
    - Nodes assigned monotonically increasing IDs
  - Grows alternate subtrees
  - When alternate more accurate => replace old
  - $O(w)$ better runtime than VFDT-window
Ensemble of Classifiers Algorithm

- H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining Concept-Drifting Data Streams using Ensemble Classifiers”, KDD'03.

- Method (derived from the ensemble idea in classification)
  - train K classifiers from K chunks
  - for each subsequent chunk
    - train a new classifier
    - test other classifiers against the chunk
    - assign weight to each classifier
    - select top K classifiers
Issues in stream classification

- Descriptive model vs. generative model
  - Generative models assume data follows some distribution while descriptive models make no assumptions
  - Distribution of stream data is unknown and may evolve, so descriptive model is better

- Label prediction vs. probability estimation
  - Classify test examples into one class or estimate $P(y|x)$ for each $y$
  - Probability estimation is better:
    - Stream applications may be stochastic (an example could be assigned to several classes with different probabilities)
    - Probability estimates provide confidence information and could be used in post processing
Cluster Analysis Methods

- Cluster Analysis: Grouping similar objects into clusters
- Types of data in cluster analysis
  - Numerical, categorical, high-dimensional, …
- Major Clustering Methods
  - Partitioning Methods
  - Hierarchical Methods
  - Density-Based Methods
  - Grid-Based Methods
  - Model-Based Methods
  - Clustering High-Dimensional Data
  - Constraint-Based Clustering
- Outlier Analysis: often a by-product of cluster analysis
Stream Clustering: A Mean Approach

1. Slide time window by one time step
2. Delete old features out of time window from their clusters
3. Generate features for data in this step
4. For each new features classify in old or new cluster (outlier)

Make use of distance methods to compute clusters, evolving cluster.

- If marker in common with a cluster member, assign to that cluster
- If near a cluster, assign to nearest cluster
- Otherwise it is an outlier and a candidate new cluster
Stream Clustering: A K-Median Approach

● Base on the k-median method
  ○ Data stream points from metric space
  ○ Find k clusters in the stream s.t. the sum of distances from data points to their closest center is minimized

● Constant factor approximation algorithm
  ○ In small space, a simple two step algorithm:
    ○ For each set of M records, $S_i$, find $O(k)$ centers in $S_1, ..., S_l$
      ■ Local clustering: Assign each point in $S_i$ to its closest center
    ○ Let $S'$ be centers for $S_1, ..., S_l$ with each center weighted by number of points assigned to it
      ■ Cluster $S'$ to find $k$ centers
Hierarchical Clustering Tree

level-i medians

level-(i+1) medians

data points
Hierarchical Tree and Drawbacks

- Method:
  - maintain at most m level-i medians
  - On seeing m of them, generate $O(k)$ level-$(i+1)$ medians of weight equal to the sum of the weights of the intermediate medians assigned to them

- Drawbacks:
  - Low quality for evolving data streams (register only k centers)
  - Limited functionality in discovering and exploring clusters over different portions of the stream over time
Thank You!

Questions?