Mining Data Streams

- What is stream data? Why Stream Data Systems?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
Characteristics of Data Streams

- **Data Streams**
  - Data streams—continuous, ordered, changing, fast, huge amount
  - Traditional DBMS—data stored in finite, persistent data sets

- **Characteristics**
  - Huge volumes of continuous data, possibly infinite
  - Fast changing and requires fast, real-time response
  - Data stream captures nicely our data processing needs of today
  - Random access is expensive—single scan algorithm (*can only have one look*)
  - Store only the summary of the data seen thus far
  - Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing
Stream Data Applications

- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Engineering & industrial processes: power supply & manufacturing
- Sensor, monitoring & surveillance: video streams, RFIDs
- Security monitoring
- Web logs and Web page click streams
- Massive data sets (even saved but random access is too expensive)
DBMS versus DSMS

- Persistent relations
- One-time queries
- Random access
- “Unbounded” disk store
- Only current state matters
- No real-time services
- Relatively low update rate
- Data at any granularity
- Assume precise data
- Access plan determined by query processor, physical DB design

- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
- Historical data is important
- Real-time requirements
- Possibly multi-GB arrival rate
- Data at fine granularity
- Data stale/imprecise
- Unpredictable/variable data arrival and characteristics

Ack. From Motwani’s PODS tutorial slides
Mining Data Streams

- What is stream data? Why Stream Data Systems?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
Architecture: Stream Query Processing

SDMS (Stream Data Management System)

Scratch Space (Main memory and/or Disk)

User/Application

Continuous Query

Stream Query Processor

Results

Multiple streams
Challenges of Stream Data Processing

- Multiple, continuous, rapid, time-varying, ordered streams
- Main memory computations
- Queries are often continuous
  - Evaluated continuously as stream data arrives
  - Answer updated over time
- Queries are often complex
  - Beyond element-at-a-time processing
  - Beyond stream-at-a-time processing
  - Beyond relational queries (scientific, data mining, OLAP)
- Multi-level/multi-dimensional processing and data mining
  - Most stream data are at low-level or multi-dimensional in nature
Processing Stream Queries

- Query types
  - One-time query vs. continuous query
    - (being evaluated continuously as stream continues to arrive)
  - Predefined query vs. ad-hoc query (issued on-line)

- Unbounded memory requirements
  - For real-time response, main memory algorithm should be used
  - Memory requirement is unbounded if one will join future tuples

- Approximate query answering
  - With bounded memory, it is not always possible to produce exact answers
  - High-quality approximate answers are desired
  - Data reduction and synopsis construction methods
    - Sketches, random sampling, histograms, wavelets, etc.
Methodologies for Stream Data Processing

- Major challenges
  - Keep track of a large universe, e.g., pairs of IP address, not ages
- Methodology
  - **Synopses** (trade-off between accuracy and storage): A summary given in brief terms that covers the major points of a subject matter
  - Use *synopsis data structure*, much smaller ($O(\log^k N)$ space) than their base data set ($O(N)$ space)
  - Compute an *approximate answer* within a *small error range* (factor $\epsilon$ of the actual answer)
- Major methods
  - Random sampling
  - Histograms
  - Sliding windows
  - Multi-resolution model
  - Sketches
  - Randomized algorithms
Stream Data Processing Methods (1)

- Random sampling (but without knowing the total length in advance)
  - Reservoir sampling: maintains a set of $s$ candidates in the reservoir, which form a true random sample of the element seen so far in the stream. As the data stream flow, every new element has a certain probability ($s/N$) of replacing an old element in the reservoir.

- Sliding windows
  - Make decisions based only on recent data of sliding window size $w$
  - An element arriving at time $t$ expires at time $t + w$

- Histograms
  - Approximate the frequency distribution of element values in a stream
  - Partition data into a set of contiguous buckets
  - Equal-width (equal value range for buckets) vs. V-optimal (minimizing frequency variance within each bucket)

- Multi-resolution models
  - Popular models: balanced binary trees, micro-clusters, and wavelets
Stream Data Processing Methods (2)

- **Sketches**
  - Histograms and wavelets require multi-passes over the data but sketches can operate in a single pass
  - Frequency moments of a stream \( A = \{a_1, \ldots, a_N\} \), \( F_k \):
    \[
    F_k = \sum_{i=1}^{v} m_i^k
    \]
    where \( v \): the universe or domain size, \( m_i \): the frequency of \( i \) in the sequence
  - Given \( N \) elements and \( v \) values, sketches can approximate \( F_0, F_1, F_2 \) in \( O(\log v + \log N) \) space

- **Randomized algorithms**
  - Monte Carlo algorithm: bound on running time but may not return correct result
  - Chebyshev’s inequality: \( P(\mid X - \mu \mid > k) \leq \frac{\sigma^2}{k^2} \)
    - Let \( X \) be a random variable with mean \( \mu \) and standard deviation \( \sigma \)
  - Chernoff bound: \( P[X < (1+\delta)\mu \mid] < e^{-\mu\delta^2/4} \)
    - Let \( X \) be the sum of independent Poisson trials \( X_1, \ldots, X_n \), \( \delta \) in \((0, 1]\)
    - The probability decreases exponentially as we move from the mean
Approximate Query Answering in Streams

- **Sliding windows**
  - Only over sliding windows of *recent stream data*
  - Approximation but often more desirable in applications

- **Batched processing, sampling and synopses**
  - **Batched** if update is fast but computing is slow
    - Compute periodically, not very timely
  - **Sampling** if update is slow but computing is fast
    - Compute using sample data, but not good for joins, etc.
  - **Synopsis** data structures
    - Maintain a small *synopsis* or *sketch* of data
    - Good for querying historical data

- **Blocking operators, e.g., sorting, avg, min, etc.**
  - **Blocking** if unable to produce the first output until seeing the entire input
Projects on DSMS (Data Stream Management System)

- Research projects and system prototypes
  - **STREAM** (Stanford): A general-purpose DSMS
  - **Cougar** (Cornell): sensors
  - Aurora (Brown/MIT): sensor monitoring, dataflow
  - **Hancock** (AT&T): telecom streams
  - **Niagara** (OGI/Wisconsin): Internet XML databases
  - OpenCQ (Georgia Tech): triggers, incr. view maintenance
  - **Tapestry** (Xerox): pub/sub content-based filtering
  - **Telegraph** (Berkeley): adaptive engine for sensors
  - **Tradebot** (www.tradebot.com): stock tickers & streams
  - **Tribeca** (Bellcore): network monitoring
  - **MAIDS** (UIUC/NCSA): Mining Alarming Incidents in Data Streams
Stream Data Mining vs. Stream Querying

- Stream mining—A more challenging task in many cases
  - It shares most of the difficulties with stream querying
    - But often requires less “precision”, e.g., no join, grouping, sorting
  - Patterns are hidden and more general than querying
  - It may require exploratory analysis
    - Not necessarily continuous queries

- Stream data mining tasks
  - Multi-dimensional on-line analysis of streams
  - Mining outliers and unusual patterns in stream data
  - Clustering data streams
  - Classification of stream data
Mining Data Streams

- What is stream data? Why Stream Data Systems?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
Challenges for Mining Dynamics in Data Streams

- Most stream data are at pretty low-level or multi-dimensional in nature: needs ML/MD processing

- Analysis requirements
  - Multi-dimensional trends and unusual patterns
  - Capturing important changes at multi-dimensions/levels
  - Fast, real-time detection and response
  - Comparing with data cube: Similarity and differences

- Stream (data) cube or stream OLAP: Is this feasible?
  - Can we implement it efficiently?
Multi-Dimensional Stream Analysis: Examples

- **Analysis of Web click streams**
  - Raw data at low levels: seconds, web page addresses, user IP addresses, …
  - Analysts want: changes, trends, unusual patterns, at reasonable levels of details
  - E.g., *Average clicking traffic in North America on sports in the last 15 minutes is 40% higher than that in the last 24 hours.*

- **Analysis of power consumption streams**
  - Raw data: power consumption flow for every household, every minute
  - Patterns one may find: *average hourly power consumption surges up 30% for manufacturing companies in Chicago in the last 2 hours today than that of the same day a week ago*
A Stream Cube Architecture

- A tilted time frame
  - Different time granularities
    - second, minute, quarter, hour, day, week, …

- Critical layers
  - Minimum interest layer (m-layer)
  - Observation layer (o-layer)
  - User: watches at o-layer and occasionally needs to drill-down down to m-layer

- Partial materialization of stream cubes
  - Full materialization: too space and time consuming
  - No materialization: slow response at query time
  - Partial materialization: what do we mean “partial”?
Cube: A Lattice of Cuboids

0-D(apex) cuboid
1-D cuboids
2-D cuboids
3-D cuboids
4-D(base) cuboid
**Time Dimension: A Titled Time Model**

- **Natural tilted time frame:**
  - Example: Minimal: 15min, then 4 * 15mins → 1 hour, 24 hours → day, ...

- **Logarithmic tilted time frame:**
  - Example: Minimal: 1 minute, then 1, 2, 4, 8, 16, 32, ...

---

![Time Dimension Diagram](image)
Pyramidal tilted time frame:

Example: Suppose there are 5 frames and each takes maximal 3 snapshots

Given a snapshot number N, if $N \mod 2^d = 0$, insert into the frame number $d$. If there are more than 3 snapshots, “kick out” the oldest one.

<table>
<thead>
<tr>
<th>Frame no.</th>
<th>Snapshots (by clock time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>69 67 65</td>
</tr>
<tr>
<td>1</td>
<td>70 66 62</td>
</tr>
<tr>
<td>2</td>
<td>68 60 52</td>
</tr>
<tr>
<td>3</td>
<td>56 40 24</td>
</tr>
<tr>
<td>4</td>
<td>48 16</td>
</tr>
<tr>
<td>5</td>
<td>64 32</td>
</tr>
</tbody>
</table>
Two Critical Layers in the Stream Cube

(*, theme, quarter)

m-layer (minimal interest)

(user-group, URL-group, minute)

o-layer (observation)

(individual-user, URL, second)

(primitive) stream data layer
OLAP Operation and Cube Materialization

- OLAP (Online Analytical Processing) operations:
  - Roll up (drill-up): summarize data
    - by climbing up hierarchy or by dimension reduction
  - Drill down (roll down): reverse of roll-up
    - from higher level summary to lower level summary or detailed data, or introducing new dimensions
  - Slice and dice: project and select
  - Pivot (rotate): reorient the cube, visualization, 3D to series of 2D planes

- Cube partial materialization
  - Store some pre-computed cuboids for fast online processing
On-Line Partial Materialization vs. OLAP Processing

- On-line materialization
  - Materialization takes precious space and time
    - Only incremental materialization (with tilted time frame)
  - Only materialize “cuboids” of the critical layers?
    - Online computation may take too much time
  - Preferred solution:
    - *popular-path* approach: Materializing those along the popular drilling paths
    - *H-tree structure*: Such cuboids can be computed and stored efficiently using the H-tree structure

- Online aggregation vs. query-based computation
  - Online computing while streaming: aggregating stream cubes
  - Query-based computation: using computed cuboids
Stream Cube Structure: From m-layer to o-layer

(A1, *, C1)
(A1, *, C2)
(A1, B1, C1)
(A2, *, C1)
(A1, B1, C2)
(A1, B2, C1)
(A1, B2, C2)
(A2, B1, C1)
(A2, B2, C1)
(A2, B2, C2)
(A2, B1, C2)
(A2, B2, C2)
An H-Tree Cubing Structure

Observation layer

Minimal int. layer

Chicago | Urbana | Springfield

Elec | Chem | Elec | Bio

6:00AM-7:00AM 156
7:00AM-8:00AM 201
8:00AM-9:00AM 235
.....
Benefits of H-Tree and H-Cubing

- H-tree and H-cubing
  - Developed for computing data cubes and ice-berg cubes
    - J. Han, J. Pei, G. Dong, and K. Wang, “Efficient Computation of Iceberg Cubes with Complex Measures”, SIGMOD'01
  - Fast cubing, space preserving in cube computation
- Using H-tree for stream cubing
  - Space preserving
    - Intermediate aggregates can be computed incrementally and saved in tree nodes
  - Facilitate computing other cells and multi-dimensional analysis
  - H-tree with computed cells can be viewed as stream cube
Mining Data Streams

- What is stream data? Why Stream Data Systems?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
What Is Frequent Pattern Analysis?

- **Frequent pattern**: A pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set.
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining.
- **Motivation**: Finding inherent regularities in data.
  - What products were often purchased together?—Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- **Applications**
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.
Frequent Patterns for Stream Data

- Frequent pattern mining is valuable in stream applications
  - e.g., network intrusion mining (Dokas et al., ’02)
- Mining precise freq. patterns in stream data: unrealistic
  - Even store them in a compressed form, such as FPtree
- How to mine frequent patterns with good approximation?
  - Approximate frequent patterns (Manku & Motwani, VLDB’02)
  - Keep only current frequent patterns? No changes can be detected
- Mining evolution freq. patterns (C. Giannella, J. Han, X. Yan, P.S. Yu, 2003)
  - Use tilted time window frame
  - Mining evolution and dramatic changes of frequent patterns
- Space-saving computation of frequent and top-k elements (Metrally, Agrawal, and El Abbadi, ICDT'05)
Mining Approximate Frequent Patterns

- Mining **precise** freq. patterns in stream data: unrealistic
  - Even store them in a compressed form, such as FPtree
- **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

Bucket 1 | Bucket 2 | Bucket 3

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
Approximation Guarantee

- Given: (1) support threshold: $\sigma$, (2) error threshold: $\epsilon$, and (3) stream length $N$
- Output: items with frequency counts exceeding $(\sigma - \epsilon)N$
- How much do we undercount?
  - If stream length seen so far = $N$ and bucket-size = $1/\epsilon$
  - then frequency count error $\leq$ buckets
    $$= N/bucket-size = N/(1/\epsilon) = \epsilon N$$
- Approximation guarantee
  - No false negatives
  - False positives have true frequency count at least $(\sigma-\epsilon)N$
  - Frequency count underestimated by at most $\epsilon N$
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

If we put 3 buckets of data into main memory one time, then decrease each frequency count by 3
Update of Summary Data Structure

summary data + 3 bucket data in memory → summary data

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

- **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

- A space-saving method for stream frequent item mining
  - Metwally, Agrawal, and El Abbadi, ICDT'05
Mining Evolution of Frequent Patterns for Stream Data

- Approximate frequent patterns (Manku & Motwani VLDB’02)
  - Keep only current frequent patterns—No changes can be detected
- Mining evolution and dramatic changes of frequent patterns (Giannella, Han, Yan, Yu, 2003)
  - Use tilted time window frame
  - Use compressed form to store significant (approximate) frequent patterns and their time-dependent traces
- Note: To mine precise counts, one has to trace/keep a fixed (and small) set of items
Mining Data Streams

- What is stream data? Why Stream Data Systems?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
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
  - Other methods
- Are they all good for stream classification?
Classification for Dynamic Data Streams

- Decision tree induction for stream data classification
  - VFDT (Very Fast Decision Tree)/CVFDT (Domingos, Hulten, Spencer, KDD00/KDD01)

- 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
    - Naïve Bayesian
    - Ensemble (Wang, Fan, Yu, Han. KDD’03)
    - K-nearest neighbors (Aggarwal, Han, Wang, Yu. KDD’04)
    - Classifying skewed stream data (Gao, Fan, and Han, SDM'07)

- Evolution modeling: Tilted time framework, incremental updating, dynamic maintenance, and model construction
  - Comparing of models to find changes
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: $r$: random variable, $R$: range of $r$, $N$: # 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

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 VFDT)

- 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
Classifying Data Streams with Skewed Distribution

- Stream Classification:
  - Construct a classification model based on past records
  - Use the model to predict labels for new data
  - Help decision making

- Concept drifts:
  - Define and analyze concept drifts in data streams
  - Show that expected error is not directly related to concept drifts

- Classify data stream with skewed distribution (i.e., rare events)
  - Employ both *biased sampling* and *ensemble* techniques
  - Results indicate the proposed method reduces classification errors on the minority class
Concept Drifts

- Changes in $P(x, y)$ x-feature vector y-class label $P(x,y) = P(y|x)P(x)$

- Four possibilities:
  - No change: $P(y|x)$, $P(x)$ remain unchanged
  - Feature change: only $P(x)$ changes
  - Conditional change: only $P(y|x)$ changes
  - Dual change: both $P(y|x)$ and $P(x)$ changes

- Expected error:
  $$Err = \int_{(x,y) \in P(x,y)} P(x)(1 - P(y_p|x))dx$$

- No matter how concept changes, the expected error could increase, decrease, or remain unchanged

- Training on the most recent data could help reduce expected error
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
Mining Skewed Data Stream

- Skewed distribution
  - Seen in many stream applications where positive examples are much less popular than the negative ones.
  - Credit card fraud detection, network intrusion detection…

- Existing stream classification methods
  - Evaluate their methods on data with balanced class distribution

- Problems of these methods on skewed data:
  - Tend to ignore positive examples due to the small number
  - The cost of misclassifying positive examples is usually huge, e.g., misclassifying credit card fraud as normal
Stream Ensemble Approach (1)

$S_1$  $S_2$  $S_m$  $S_{m+1}$

Classification Model

$S_m$ as training data? Positive examples not sufficient!
Stream Ensemble Approach (2)

Sampling

\[ f^E(x) = \frac{1}{k} \sum_{i=1}^{k} f^i(x) \]
Analysis

- **Error Reduction:**
  - Sampling: \( f_c(x) = P(c|x) + \beta_c + \eta_c(x) \)
  \[ \sigma_b^2 = (\sigma_{\eta+}^2 + \sigma_{\eta-}^2) / s^2 \]
  - Ensemble: \( f^E_c(x) = P(c|x) + \bar{\beta}_c + \bar{\eta}_c(x) \)
  \[ \sigma_{\eta_c}^2 = \frac{1}{k^2} \sum_{i=1}^{k} \sigma_{\eta_i}^2 \]

- **Efficiency Analysis:**
  - Single model: \( O(d(n_p + n_q) \log(n_p + n_q)) \)
  - Ensemble: \( O(dk(n_p + n_q) \log(n_p + n_q)) \)
  - Ensemble is more efficient
Experiments: Mean Squared Error on Synthetic Data

- Test on concept-drift streams

**Table 2: Mean Squared Error on Deterministic Stream Data**

<table>
<thead>
<tr>
<th>Changes</th>
<th>Decision Trees</th>
<th>Naive Bayes</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SE</td>
<td>NS</td>
<td>SS</td>
</tr>
<tr>
<td>Feature</td>
<td>0.1275</td>
<td>0.9637</td>
<td>0.6446</td>
</tr>
<tr>
<td>Conditional</td>
<td>0.0943</td>
<td>0.9805</td>
<td>0.5500</td>
</tr>
<tr>
<td>Dual</td>
<td>0.0854</td>
<td>0.9521</td>
<td>0.5174</td>
</tr>
</tbody>
</table>

**Table 3: Mean Squared Error on Stochastic Stream Data**

<table>
<thead>
<tr>
<th>Changes</th>
<th>Decision Trees</th>
<th>Naive Bayes</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SE</td>
<td>NS</td>
<td>SS</td>
</tr>
<tr>
<td>Feature</td>
<td>0.0847</td>
<td>0.6823</td>
<td>0.4639</td>
</tr>
<tr>
<td>Conditional</td>
<td>0.0552</td>
<td>0.6421</td>
<td>0.4463</td>
</tr>
<tr>
<td>Dual</td>
<td>0.0684</td>
<td>0.6758</td>
<td>0.4107</td>
</tr>
</tbody>
</table>
Experiments: Mean Squared Error on Real Data

- Test on real data

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Decision Trees</th>
<th>Naive Bayes</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SE</td>
<td>NS</td>
<td>SS</td>
</tr>
<tr>
<td>Thyroid1</td>
<td>0.0000</td>
<td>0.0799</td>
<td>0.0003</td>
</tr>
<tr>
<td>Thyroid2</td>
<td>0.0001</td>
<td>0.0266</td>
<td>0.0047</td>
</tr>
<tr>
<td>Opt</td>
<td>0.0147</td>
<td>0.1160</td>
<td>0.0414</td>
</tr>
<tr>
<td>Letter</td>
<td>0.0191</td>
<td>0.2290</td>
<td>0.0653</td>
</tr>
<tr>
<td>Covtype</td>
<td>0.0003</td>
<td>0.2500</td>
<td>0.0834</td>
</tr>
</tbody>
</table>
Experiments: Model Accuracy

- Model accuracy

Table 5: Decision Tree as Base Learner

<table>
<thead>
<tr>
<th></th>
<th>SE</th>
<th>NS</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic1</td>
<td>0.9464</td>
<td>0.5175</td>
<td>0.6944</td>
</tr>
<tr>
<td>Synthetic2</td>
<td>0.9337</td>
<td>0.4840</td>
<td>0.6611</td>
</tr>
<tr>
<td>Thyroid1</td>
<td>1.0000</td>
<td>0.9999</td>
<td>0.9999</td>
</tr>
<tr>
<td>Thyroid2</td>
<td>0.9998</td>
<td>0.9998</td>
<td>0.9996</td>
</tr>
<tr>
<td>Opt</td>
<td>0.9942</td>
<td>0.9495</td>
<td>0.9777</td>
</tr>
<tr>
<td>Letter</td>
<td>0.9931</td>
<td>0.9467</td>
<td>0.9782</td>
</tr>
<tr>
<td>Covtype</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

Table 6: Naive Bayes as Base Learner

<table>
<thead>
<tr>
<th></th>
<th>SE</th>
<th>NS</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic1</td>
<td>0.9532</td>
<td>0.8220</td>
<td>0.9525</td>
</tr>
<tr>
<td>Synthetic2</td>
<td>0.9558</td>
<td>0.8355</td>
<td>0.9556</td>
</tr>
<tr>
<td>Thyroid1</td>
<td>0.9982</td>
<td>0.9979</td>
<td>0.9982</td>
</tr>
<tr>
<td>Thyroid2</td>
<td>0.9551</td>
<td>0.9054</td>
<td>0.9145</td>
</tr>
<tr>
<td>Opt</td>
<td>0.9926</td>
<td>0.9722</td>
<td>0.9898</td>
</tr>
<tr>
<td>Letter</td>
<td>0.9395</td>
<td>0.9389</td>
<td>0.9389</td>
</tr>
<tr>
<td>Covtype</td>
<td>0.9997</td>
<td>0.9995</td>
<td>0.9997</td>
</tr>
</tbody>
</table>

Table 7: Logistic Regression as Base Learner

<table>
<thead>
<tr>
<th></th>
<th>SE</th>
<th>NS</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic1</td>
<td>0.8801</td>
<td>0.8363</td>
<td>0.8737</td>
</tr>
<tr>
<td>Synthetic2</td>
<td>0.8992</td>
<td>0.8102</td>
<td>0.8854</td>
</tr>
<tr>
<td>Thyroid1</td>
<td>0.9977</td>
<td>0.9774</td>
<td>0.9909</td>
</tr>
<tr>
<td>Thyroid2</td>
<td>0.9949</td>
<td>0.9593</td>
<td>0.9930</td>
</tr>
<tr>
<td>Opt</td>
<td>0.9971</td>
<td>0.9940</td>
<td>0.9953</td>
</tr>
<tr>
<td>Letter</td>
<td>0.9545</td>
<td>0.9448</td>
<td>0.9517</td>
</tr>
<tr>
<td>Covtype</td>
<td>0.9995</td>
<td>0.9989</td>
<td>0.9994</td>
</tr>
</tbody>
</table>
Experiments: Efficiency

- Training time

Figure 4: Training Time
Mining Data Streams

- What is stream data? Why Stream Data Systems?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
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 K-Median Approach

- O'Callaghan et al., “Streaming-Data Algorithms for High-Quality Clustering”, (ICDE'02)
- 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:
    1. 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
    2. 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

Data points

Level-i medians

Level-(i+1) medians
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
Clustering for Mining Stream Dynamics

- Network intrusion detection: one example
  - Detect bursts of activities or abrupt changes in real time—by on-line clustering
- Our methodology (C. Agarwal, J. Han, J. Wang, P.S. Yu, VLDB’03)
  - Tilted time frame work: o.w. dynamic changes cannot be found
  - Micro-clustering: better quality than k-means/k-median
    - incremental, online processing and maintenance
- Two stages: micro-clustering and macro-clustering
- With limited “overhead” to achieve high efficiency, scalability, quality of results and power of evolution/change detection
Design goal
- High quality for clustering evolving data streams with greater functionality
- While keep the stream mining requirement in mind
  - One-pass over the original stream data
  - Limited space usage and high efficiency

CluStream: A framework for clustering evolving data streams
- Divide the clustering process into online and offline components
  - Online component: periodically stores summary statistics about the stream data
  - Offline component: answers various user questions based on the stored summary statistics
BIRCH: A Micro-Clustering Approach

Clustering Feature: \( CF = (N, LS, SS) \)

where \( N \): # data points, \( LS = \sum_{i=1}^{N} X_i \), \( SS = \sum_{i=1}^{N} X_i^2 \)
The CluStream Framework

- Micro-cluster
  - Statistical information about data locality
  - Temporal extension of the *cluster-feature vector*
    - Multi-dimensional points $\overline{X}_1 \ldots \overline{X}_k$ with time stamps $T_1 \ldots T_k$
    - Each point contains $d$ dimensions, i.e., $\overline{X}_i = (x_i^1 \ldots x_i^d)$
    - A micro-cluster for $n$ points is defined as a $(2.d + 3)$ tuple
      $$(\overline{CF2^x}, \overline{CF1^x}, CF2^t, CF1^t, n)$$
  - Pyramidal time frame
    - Decide at what moments the snapshots of the statistical information are stored away on disk
CluStream: Pyramidal Time Frame

- Pyramidal time frame
  - Snapshots of a set of micro-clusters are stored following the pyramidal pattern
    - They are stored at differing levels of granularity depending on the recency
  - Snapshots are classified into different orders varying from 1 to $\log(T)$
    - The $i$-th order snapshots occur at intervals of $\alpha^i$ where $\alpha \geq 1$
    - Only the last $(\alpha + 1)$ snapshots are stored
CluStream: Clustering On-line Streams

- Online micro-cluster maintenance
  - Initial creation of q micro-clusters
    - q is usually significantly larger than the number of natural clusters
  - Online incremental update of micro-clusters
    - If new point is within max-boundary, insert into the micro-cluster
    - o.w., create a new cluster
    - May delete obsolete micro-cluster or merge two closest ones

- Query-based macro-clustering
  - Based on a user-specified time-horizon $h$ and the number of macro-clusters $k$, compute macroclusters using the $k$-means algorithm
Mining Data Streams

- What is stream data? Why SDS?
- Stream data management systems: Issues and solutions
- Stream data cube and multidimensional OLAP analysis
- Stream frequent pattern analysis
- Stream classification
- Stream cluster analysis
- Research issues
Stream Data Mining: Research Issues

- Mining sequential patterns in data streams
- Mining partial periodicity in data streams
- Mining outliers and unusual patterns for botnet detection
- Stream clustering
  - Multi-dimensional clustering analysis
    - Cluster not confined to 2-D metric space, how to incorporate other features, especially non-numerical properties
  - Stream clustering with other clustering approaches
  - Constraint-based cluster analysis with data streams
- Real-time stream data mining in cyberphysical systems
Summary: Stream Data Mining

- Stream data mining: A rich and on-going research field
  - Current research focus in database community:
    - DSMS system architecture, continuous query processing, supporting mechanisms
  - Stream data mining and stream OLAP analysis
    - Powerful tools for finding general and unusual patterns
    - Effectiveness, efficiency and scalability: lots of open problems
- Our philosophy on stream data analysis and mining
  - A multi-dimensional stream analysis framework
  - Time is a special dimension: Tilted time frame
  - What to compute and what to save?—Critical layers
  - Partial materialization and precomputation
  - Mining dynamics of stream data
References on Stream Data Mining (1)

- C. Aggarwal, J. Han, J. Wang, P. S. Yu, “A Framework for Clustering Data Streams”, VLDB'03
- C. C. Aggarwal, J. Han, J. Wang and P. S. Yu, “On-Demand Classification of Evolving Data Streams”, KDD'04
- C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A Framework for Projected Clustering of High Dimensional Data Streams”, VLDB'04
- S. Babu and J. Widom, “Continuous Queries over Data Streams”, SIGMOD Record, Sept. 2001
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, “Multi-Dimensional Regression Analysis of Time-Series Data Streams”, VLDB'02
- P. Domingos and G. Hulten, “Mining high-speed data streams”, KDD'00
- A. Dobra, M. N. Garofalakis, J. Gehrke, and R. Rastogi, “Processing Complex Aggregate Queries over Data Streams”, SIGMOD’02
- J. Gehrke, F. Korn, and D. Srivastava, “On computing correlated aggregates over continuous data streams”, SIGMOD'01
- J. Gao, W. Fan, and J. Han, “A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions”, SDM'07.
References on Stream Data Mining (2)

- G. Hulten, L. Spencer and P. Domingos, “Mining time-changing data streams”, KDD’01
- S. Madden, M. Shah, J. Hellerstein, V. Raman, “Continuously Adaptive Continuous Queries over Streams”, SIGMOD’02
- G. Manku, R. Motwani, “Approximate Frequency Counts over Data Streams”, VLDB’02
- A. Metwally, D. Agrawal, and A. El Abbadi. “Efficient Computation of Frequent and Top-k Elements in Data Streams”. ICDT'05
- S. Viglas and J. Naughton, “Rate-Based Query Optimization for Streaming Information Sources”, SIGMOD’02
- Y. Zhu and D. Shasha. “StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time”, VLDB’02
- H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining Concept-Drifting Data Streams using Ensemble Classifiers”, KDD'03