POLITECNICO DI MILANO

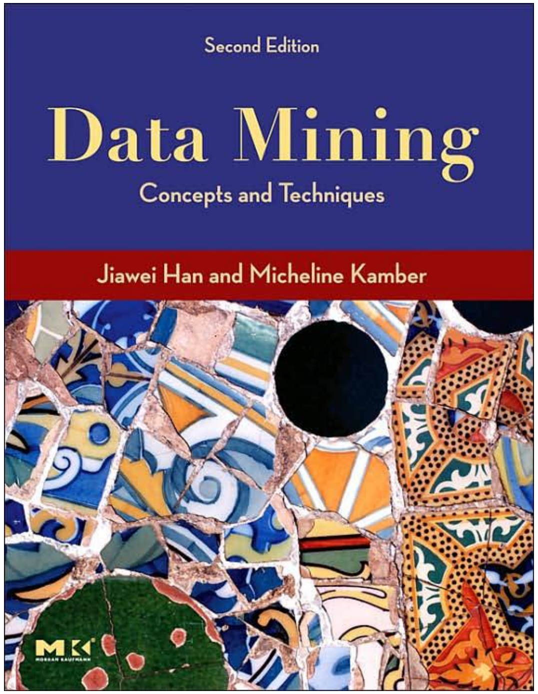
Mining Data Streams

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

Daniele Loiacono

References

- Jiawei Han and Micheline Kamber, "Data Mining: Concepts and Techniques", The Morgan Kaufmann Series in Data Management Systems (Second Edition)
  - Chapter 8

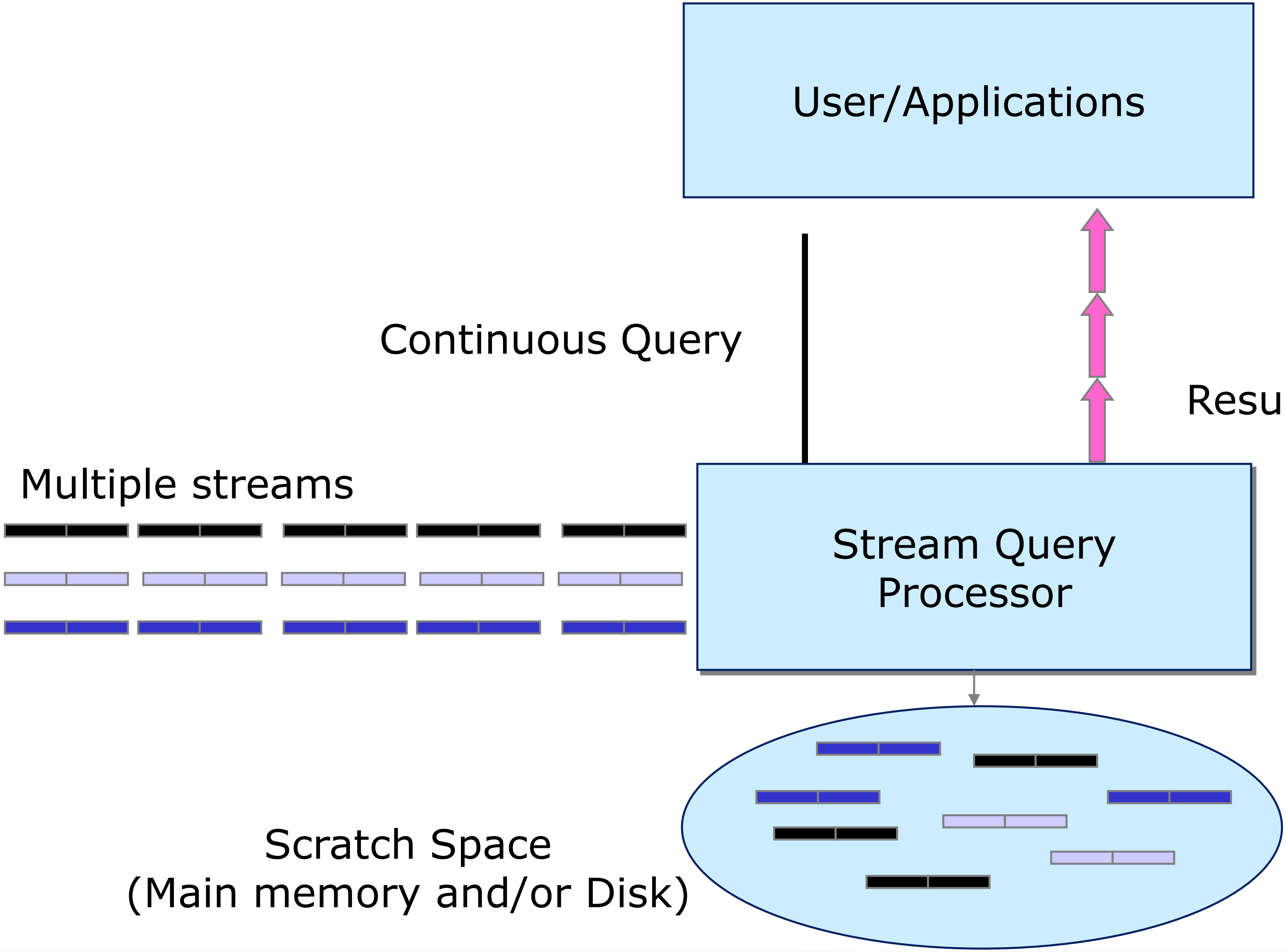


Data Streams

Examples

- 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)

Stream Query Processing



Challenges

- 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.

Stream Data Mining vs. Stream Querying

- Stream mining is 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

Processing Data

What the Methodologies for Stream Data Processing?

- Major challenges
  - Keep track of a large universe, e.g., pairs of IP address
- Methodology
  - Synopses (trade-off between accuracy and storage)
  - 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 ε of the actual answer)
- Major methods
  - Random sampling
  - Histograms
  - Sliding windows
  - Multi-resolution model
  - Sketches
  - Radomized algorithms

Stream Data Processing Methods (1)

- Random sampling
  - Reservoir sampling: maintain 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
  - Frequency moments of a stream A = {a1, …, aN}, 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

- F_0 is the number of distinct elements
- F_1 is the number of elements
- F_2 is known as repeat rate or Gini’s index of homogeneity

- Randomized algorithms
  - Monte Carlo algorithm: bound on running time but may not return correct result
  - Chebyshev's inequality: Let X be a random variable with mean μ and standard deviation σ

$$P(|X-\mu| > k) \leq \frac{\sigma^2}{k^2}$$

  - Chernoff bound:
    - Let X be the sum of independent Poisson trials X1, …, Xn, δ in (0, 1]
    - The probability decreases exponentially as we move from the mean

$$P[X < (1+\delta)\mu] < e^{-\mu\delta^2/4}$$

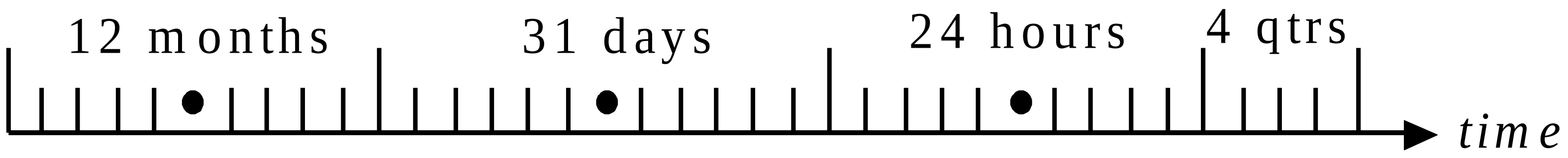
Architectures

A Stream Cube Architecture

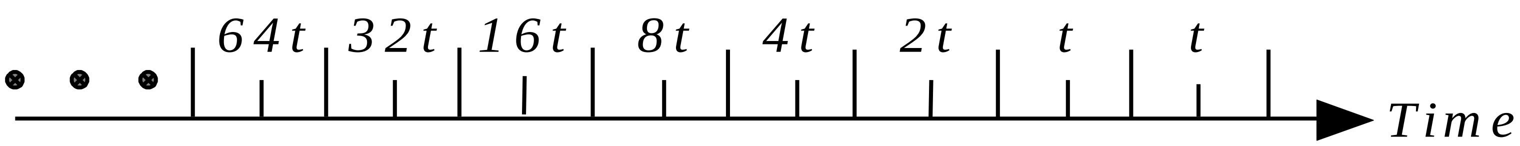
- A tilted time frame
  - Different time granularities:
    second, minute, quarter, hour, day, week, …
- Critical layers
  - Minimum interest layer (m-layer)
  - Observation layer (o-layer)
  - User: watches at o-layer and occasionally needs to drill-down down to m-layer
- Partial materialization of stream cubes
  - Full materialization: too space and time consuming
  - No materialization: slow response at query time
  - Partial materialization: what do we mean "partial"?

A Titled Time Model

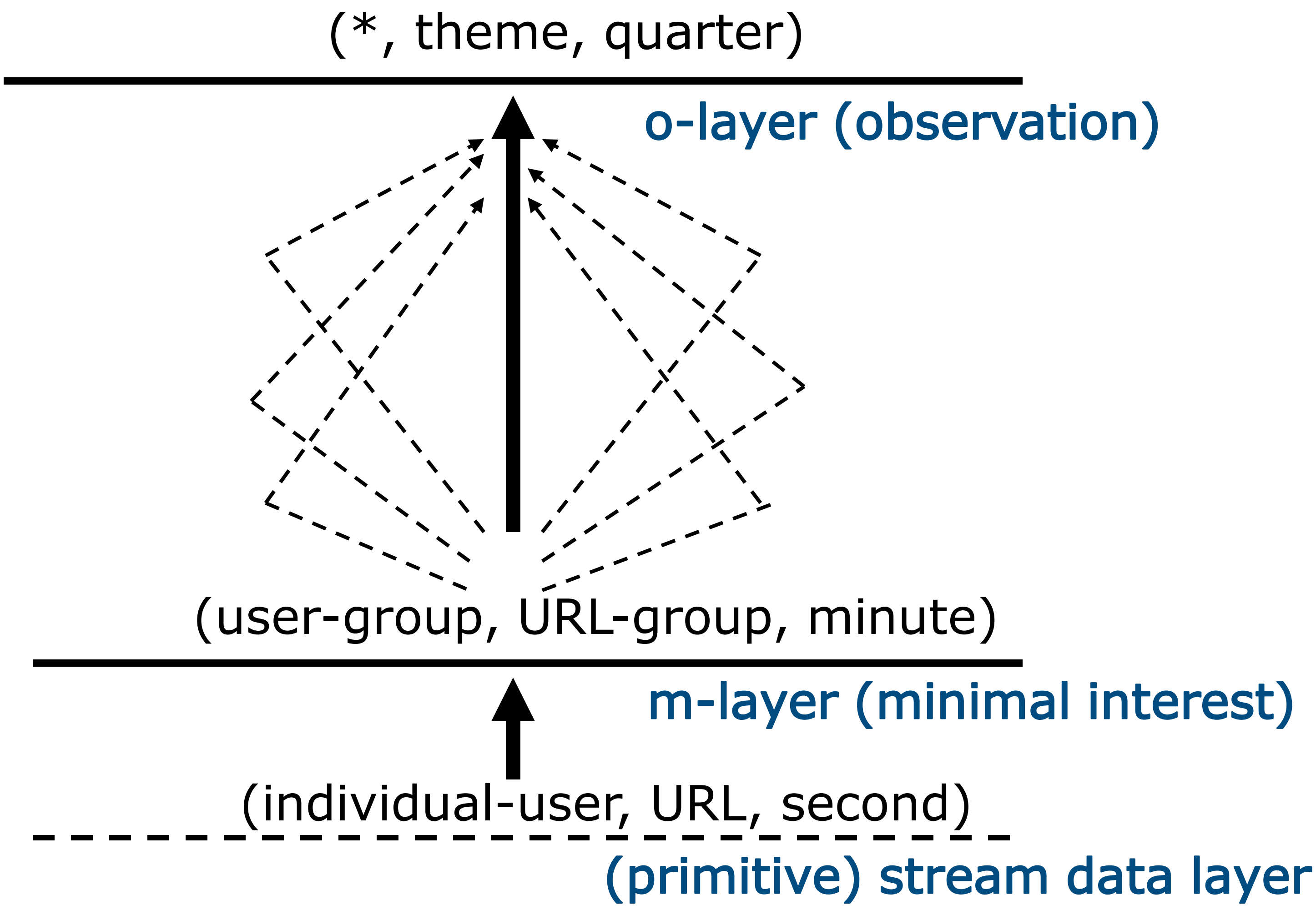
- Natural tilted time frame:
  - Example: Minimal: quarter, then 4 quarters → 1 hour, 24 hours → day, …



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



Two Critical Layers in the Stream Cube



Frequent patterns

- Frequent pattern mining is valuable in stream applications
  - e.g., network intrusion mining
- Many existing algorithms require to scan the dataset more than once.
- Multiple scans are not feasible in data streams, where there are two main approaches:
  - Focus on a set of predefined set of items
  - Provide an approximate answer
    - E.g., exploiting the Lossy Counting Algorithm

Predefined set of items

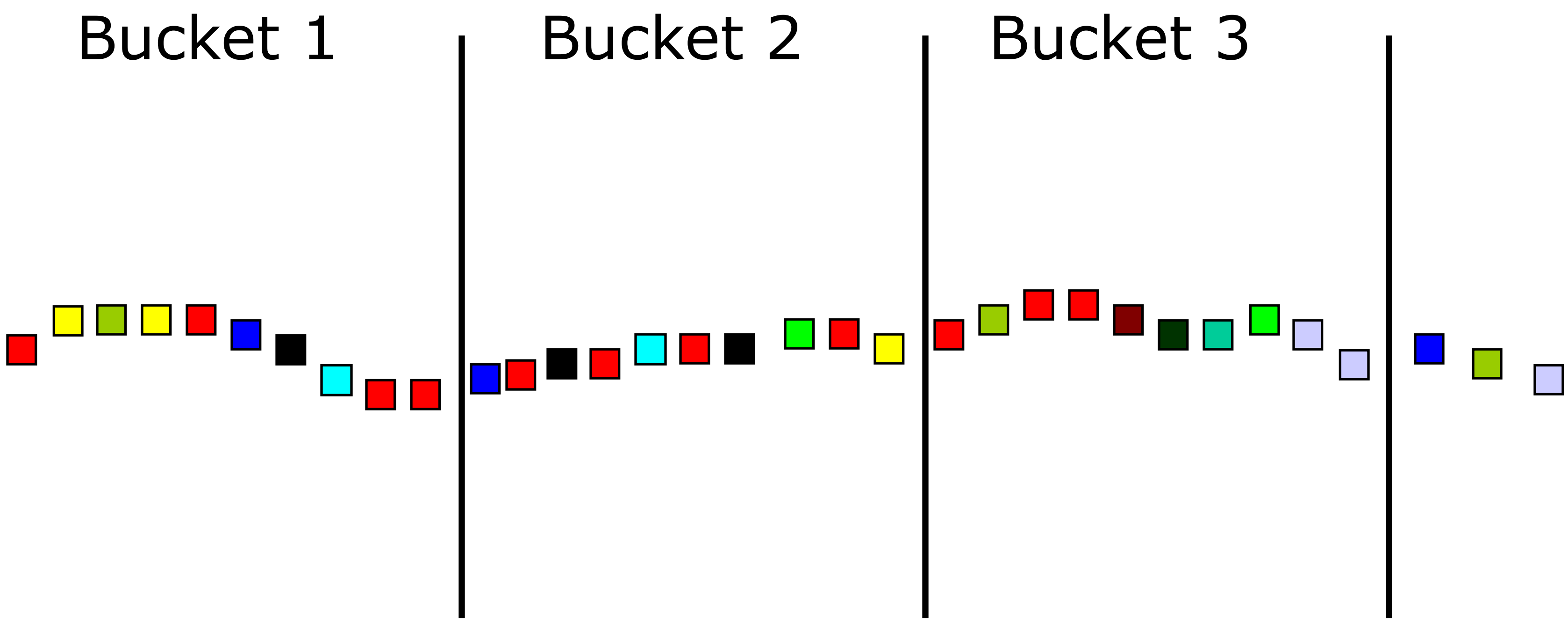
- The algorithm keeps track of a predefined set of items
- It requires a single scan of data to compute the exact frequency of each item
- How to choose the predefined set of items?
  - Focus on a set of "interesting" items
  - Focus on a set of item known to be frequent in the past
- This approach cannot be often used in practice:
  - A set of "interesting" items might not be available
  - Choosing items on the basis of past information does not account for future changes

Mining Approximate Frequent Patterns: Lossy Counting

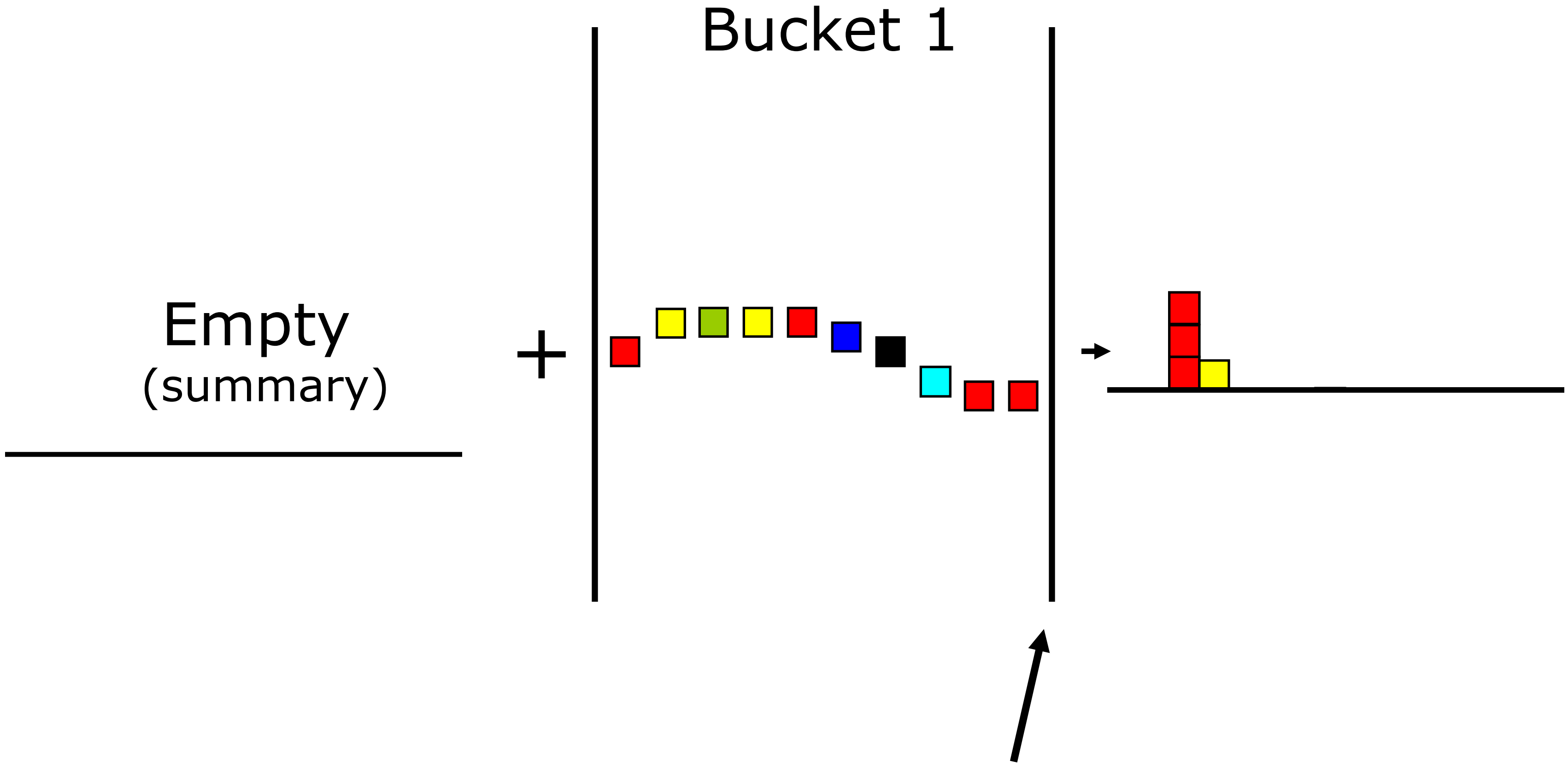
- Approximate answers are often enough (e.g., trend/pattern analysis)
- Example: a router is interested in all flows
  - whose frequency is at least 1% (σ) of the entire traffic stream seen so far
  - and feels that 1/10 of σ (ε = 0.1%) error is comfortable
- How to mine frequent patterns with good approximation?
- Lossy Counting Algorithm is able to compute the frequency of items with an error not bigger than ε

Lossy Counting for Frequent Items (1)

- Divide Stream into 'Buckets' (bucket size is 1/ ε = 1000)

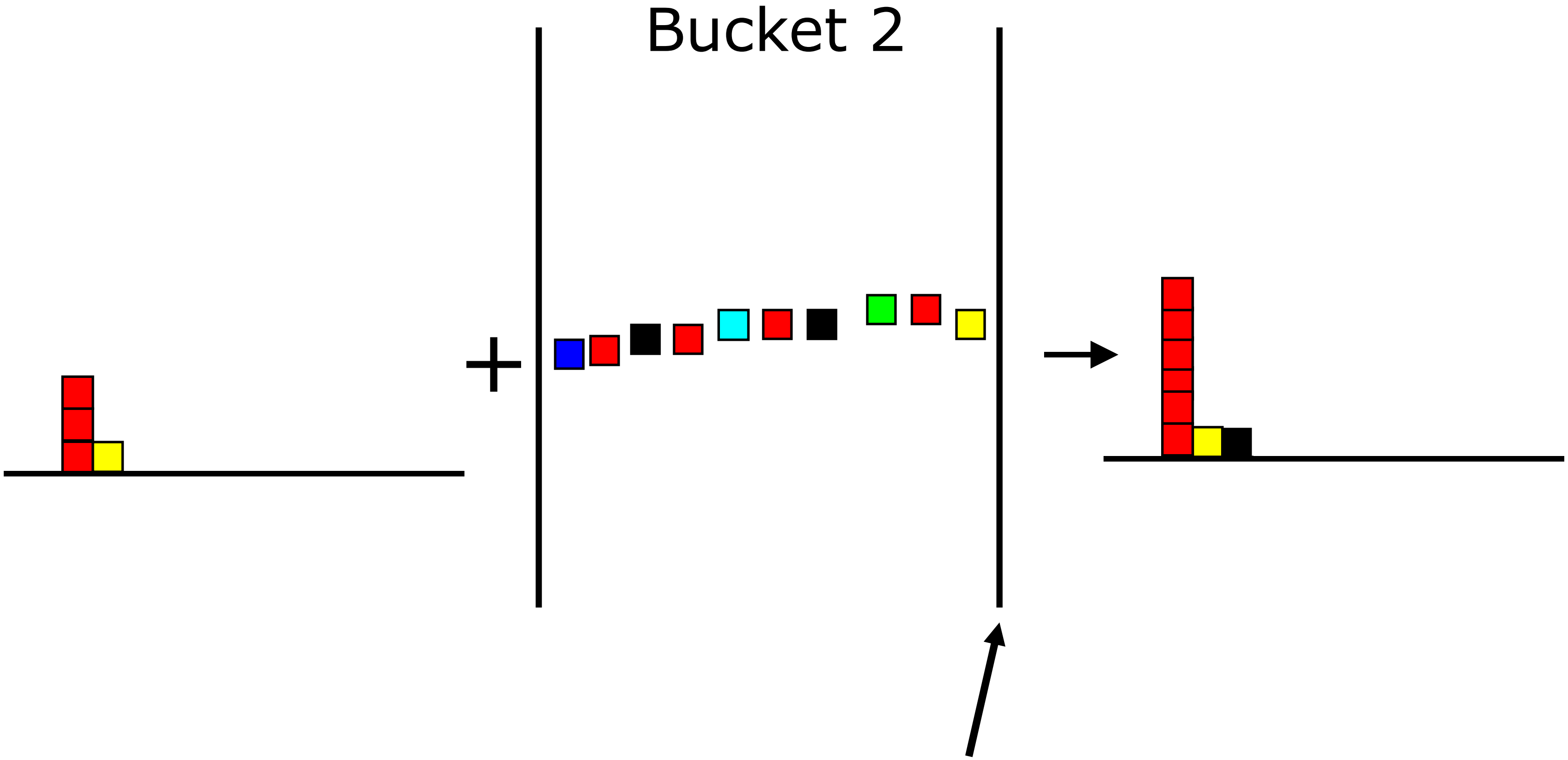


Lossy Counting for Frequent Items (2)



At bucket boundary, decrease all counters by 1

Lossy Counting for Frequent Items (1)



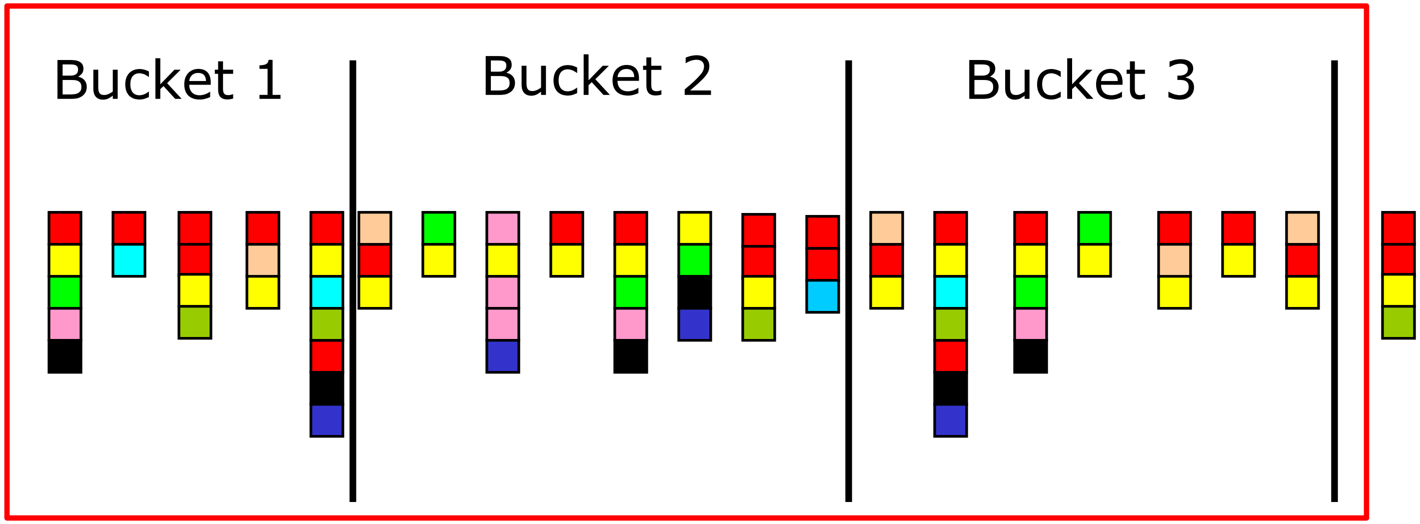
At bucket boundary, decrease all counters by 1

Lossy Counting for Frequent Items (4)

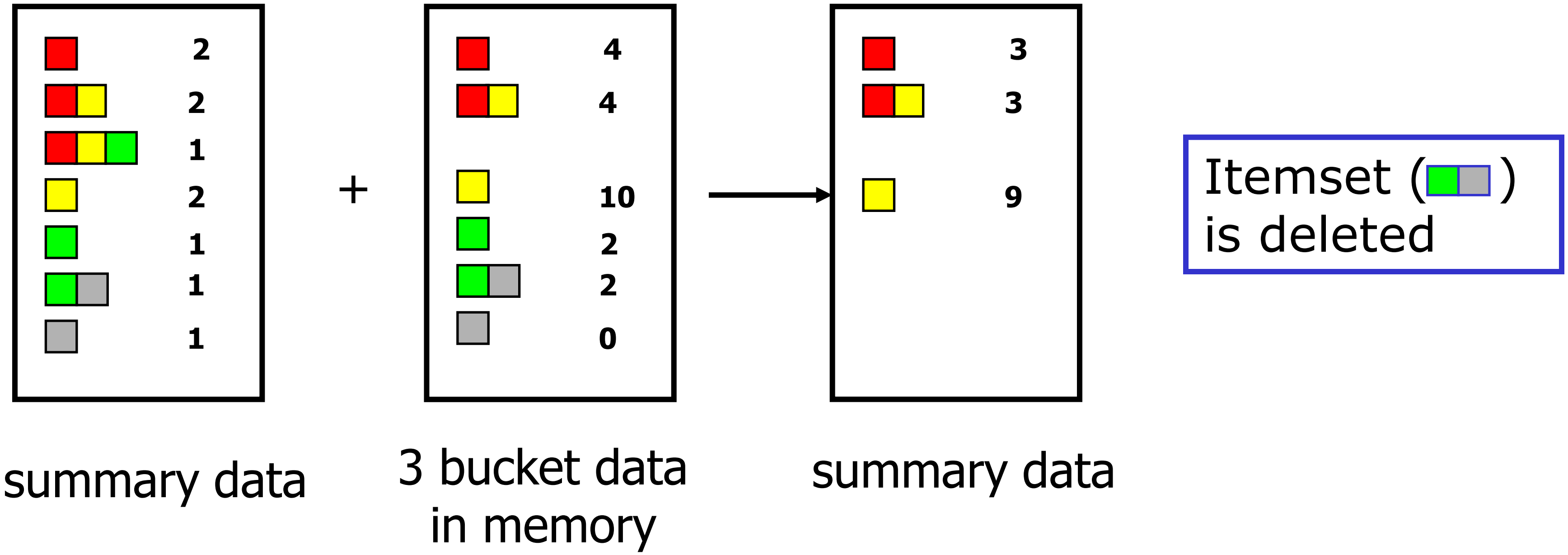
- Inputs
  - support threshold: σ
  - error threshold: ε
  - data stream of length N
- Output: items with frequency counts exceeding $(\sigma - \varepsilon)\,N$
- How much do we underestimate frequency?
  - Not more than one element is “lost” for each buket
  - The number of buckets is $N/w = \varepsilon N$
  - Frequency count underestimated by at most εN
- Approximation guarantee
  - No false negatives
  - False positives have true frequency count at least $(\sigma-\varepsilon)N$
  - The space requirement is limited to $1/\varepsilon \log(\varepsilon N)$

Lossy Counting For Frequent Itemsets

- When applied to find frequent itemsets, the list of frequencies grows exponentially
- To deal with this problem, as many buckets as possible are loaded in main memory at one time
- Example: load 3 buckets into main memory

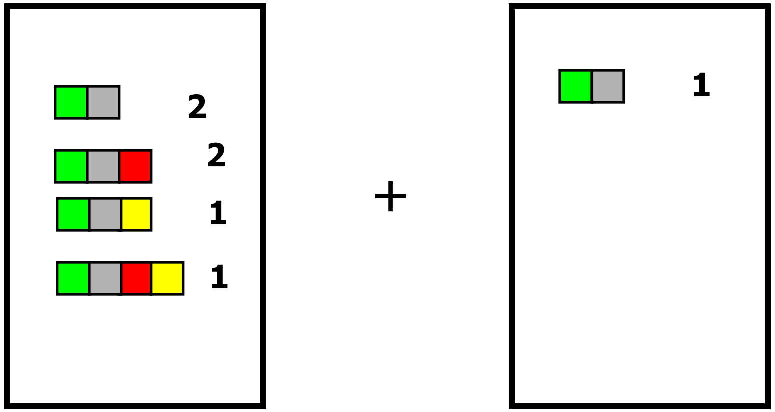


Lossy Counting For Frequent Itemsets (2)



- With large number of buckets in memory we delete more itemsets

Lossy Counting For Frequent Itemsets: Pruning Itemsets



summary data

3 bucket data in memory

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

Classification

Classification in Data Streams
What are the issues?

- It is impossible to store the whole data set, as traditional classification algorithms require
- It is usually not possible to perform multiple scans of the input data
- Data streams are time-varying! There is concept drift.

- Approaches
  - Hoeffding Trees
  - Very Fast Decision Tree
  - Concept-adapting Very Fast Decision Tree
  - Ensemble of Classifiers

Hoeffding Tree

- Initially introduced to analyze click-streams
- With high probability, lead to the same decision tree of typical algorithms
- Only uses small sample to choose optimal splitting attribute
- It is based on Hoeffding Bound principle
  - r: random variable representing the attribute selection method (e.g. information gain)
  - R: range of r
  - n: # independent observations
  - Mean of r is at least $r_{avg} - \varepsilon$, with probability $1 - \delta$

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

- The bound is used to determine, with high probability the smallest number N of examples needed at a node to select the splitting attribute

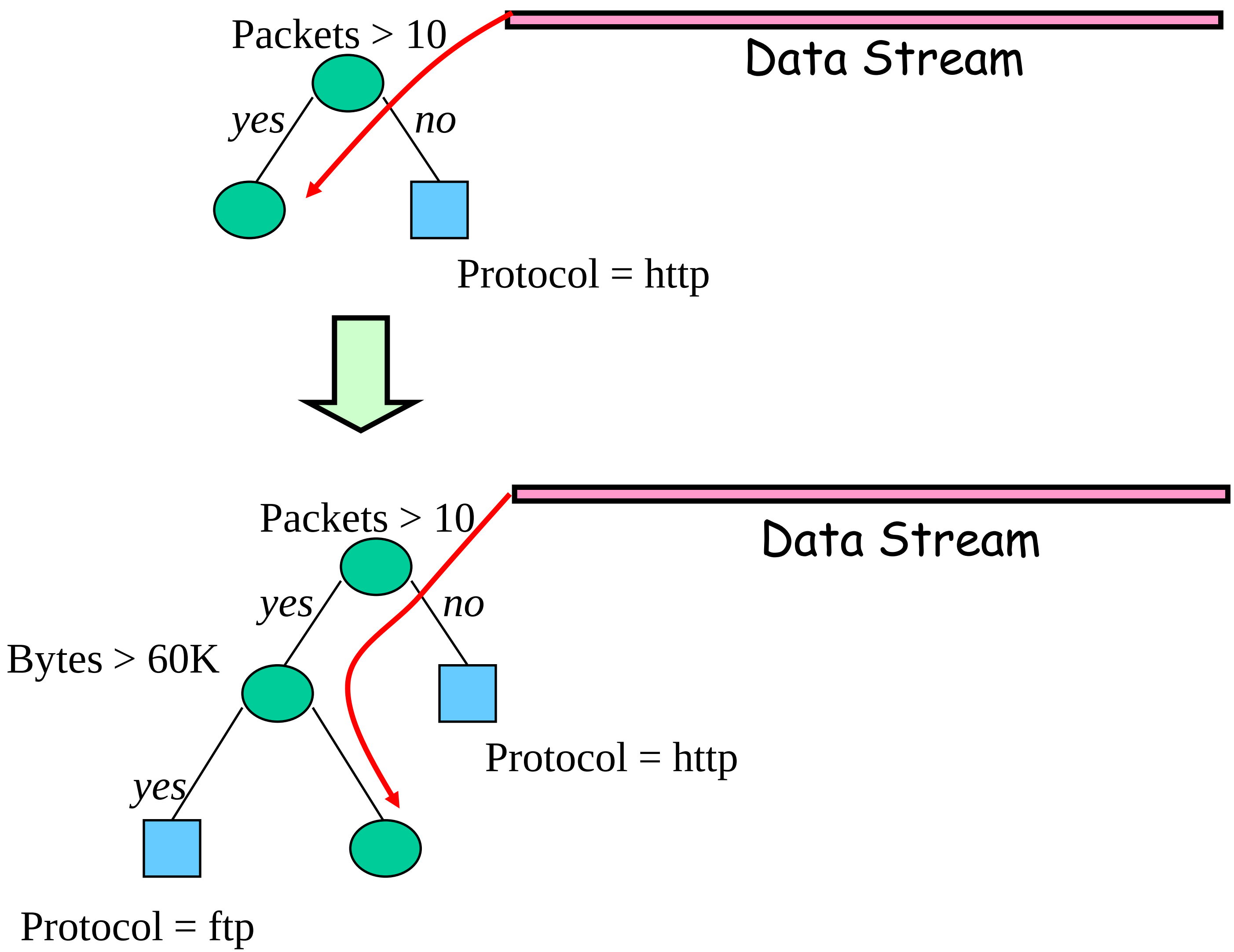
Hoeffding Tree Algorithm

- Hoeffding Tree Input
  - S: sequence of examples
  - X: attributes
  - G( ): evaluation function
  - δ: desired accuracy

```
for each example in S
   retrieve G(Xa) and G(Xb)
   if ( G(Xa) - G(Xb) > ε )
     split on Xa
     recurse to next node
     break
```

X_a and X_b are the atributes with highest values of G(), while ε is computed with the Hoeffding bound

Example



Packets > 10
yes
no
Protocol = http
Data Stream
Packets > 10
yes
no
Bytes > 60K
yes
Protocol = http
Protocol = ftp
Data Stream

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

Weaknesses

- Could spend a lot of time with ties
- Memory used with tree expansion
- Number of candidate attributes

VFDT (Very Fast Decision Tree)

- Modifications to Hoeffding Tree
  - Near-ties broken more aggressively
  - G computed every nmin
  - Deactivates certain leaves to save memory
  - Poor attributes dropped
  - Initialize with traditional learner (helps learning curve)
- Compare to Hoeffding Tree: Better time and memory
- Compare to traditional decision tree
  - Similar accuracy
  - Better runtime with 1.61 million examples
    - 21 minutes for VFDT
    - 24 hours for C4.5
- Still does not handle concept drift

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, then 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
```

Clustering

Clustering Evolving Data Streams
What methodologies?

- Compute and store summaries of past data
- Apply a divide-and-conquer strategy
- Incremental clustering of incoming data streams
- Perform microclustering as well as macroclustering anlysis
- Explore multiple time granularity for the analysis of cluster evolution
- Divide stream clustering into on-line and off-line processes

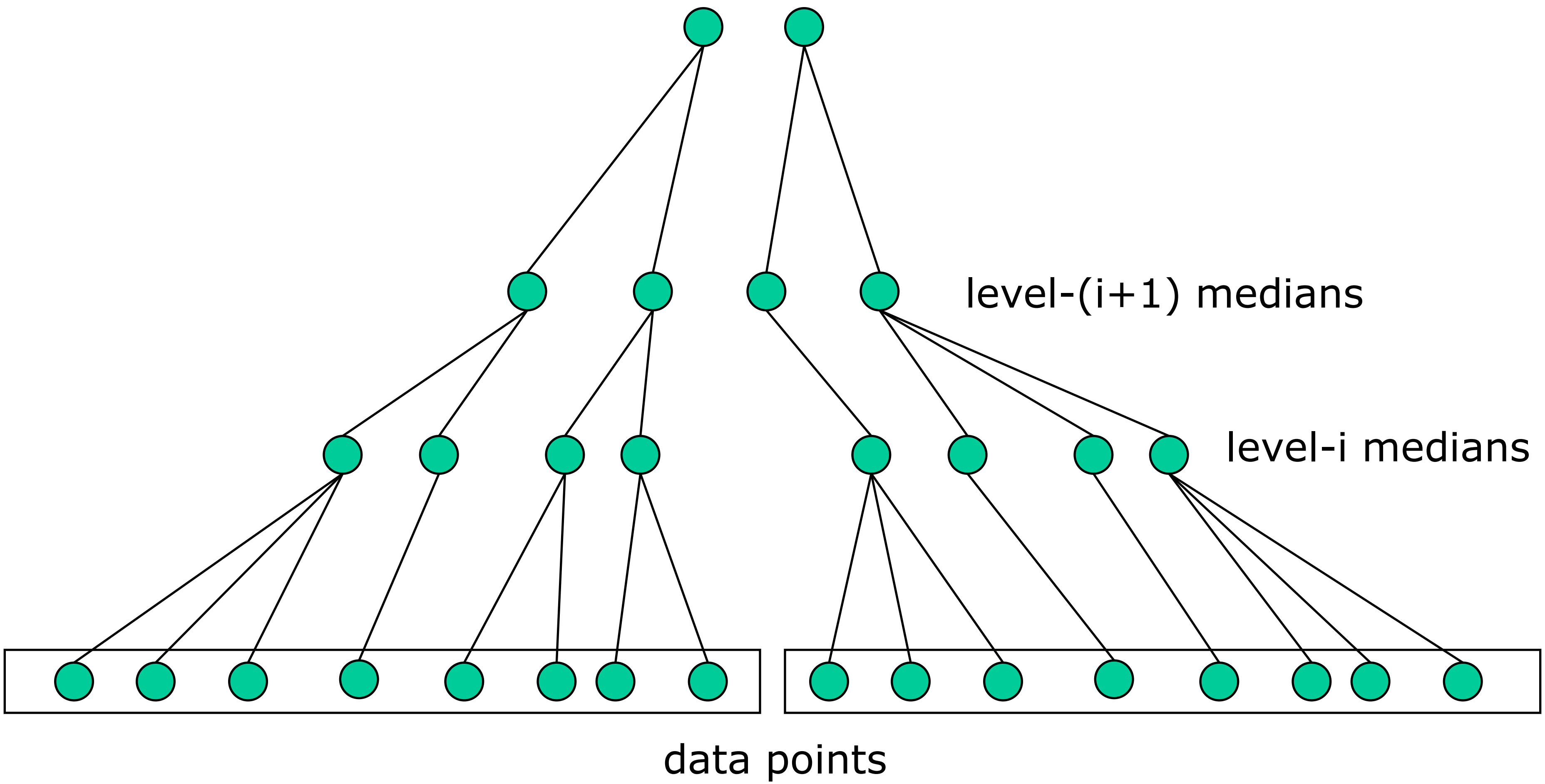
- Based on the k-median method
  - Data stream points from metric space
  - Find k clusters in the stream s.t. the sum of distances from data points to their closest center is minimized

- Two-steps approximation algorithm:

  1. For each set of M records, S_i,
     find O(k) centers in $S_1, \ldots, S_l$
     Local clustering: Assign each point in S_i to its closest center

  2. Let S’ be centers for $S_1, \ldots, S_l$ with each
     center weighted by number of points assigned to it
     Cluster S’ to find k centers

Hierarchical Clustering Tree



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

CluStream: A Framework for Clustering Evolving Data Streams

- 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

The CluStream Framework

- Micro-cluster
  - Statistical information about data locality
  - Temporal extension of the cluster-feature vector
    - Multi-dimensional points $X_1 \dots X_k \dots$
      with time stamps $T_1 \dots T_k \dots$
    - Each point contains d dimensions, i.e., $X = (x^1 \dots x^d)$
    - A micro-cluster for n points is defined as a (2.d + 3) tuple

$$\left(\overline{CF2^{x}}, \overline{CF1^{x}}, CF2^{t}, CF1^{t}, n\right)$$

- Pyramidal time frame
  - Decide at what moments the snapshots of the statistical information are stored away on disk

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

Summary

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

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
- B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom. Models and Issues in Data Stream Systems", PODS'02. (Conference tutorial)
- 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, R. Rastogi. Processing Complex Aggregate Queries over Data Streams, SIGMOD'02
- J. Gehrke, F. Korn, D. Srivastava. On computing correlated aggregates over continuous data streams. SIGMOD'01
- C. Giannella, J. Han, J. Pei, X. Yan and P.S. Yu. Mining frequent patterns in data streams at multiple time granularities, Kargupta, et al. (eds.), Next Generation Data Mining'04

References on Stream Data Mining (2)

- S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams, FOCS'00
- G. Hulten, L. Spencer and P. Domingos: Mining time-changing data streams. KDD 2001
- S. Madden, M. Shah, J. Hellerstein, V. Raman, Continuously Adaptive Continuous Queries over Streams, SIGMOD02
- 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. Muthukrishnan, Data streams: algorithms and applications, Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, 2003
- R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge Univ. Press, 1995
- 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