# CSE 590
# Data Science Fundamentals

# Mining Data Streams

Klaus Mueller

Computer Science Department
Stony Brook University and SUNY Korea

| Lecture | Topic | Projects |
|---|---|---|
| 1 | Intro, schedule, and logistics | |
| 2 | Data Science components and tasks | |
| 3 | Data types | Project #1 out |
| 4 | Introduction to R, statistics foundations | |
| 5 | Introduction to D3, visual analytics | |
| 6 | Data preparation and reduction | |
| 7 | Data preparation and reduction | Project #1 due |
| 8 | Similarity and distances | Project #2 out |
| 9 | Similarity and distances | |
| 10 | Cluster analysis | |
| 11 | Cluster analysis | |
| 12 | Pattern miming | Project #2 due |
| 13 | Pattern mining | |
| 14 | Outlier analysis | |
| 15 | Outlier analysis | Final Project proposal due |
| 16 | Classifiers | |
| 17 | Midterm | |
| 18 | Classifiers | |
| 19 | Optimization and model fitting | |
| 20 | Optimization and model fitting | |
| 21 | Causal modeling | |
| 22 | Streaming data | Final Project preliminary report due |
| 23 | Text data | |
| 24 | Time series data | |
| 25 | Graph data | |
| 26 | Scalability and data engineering | |
| 27 | Data journalism | |
| | Final project presentation | Final Project slides and final report due |

# Types of Streaming data

Transaction streams

- credit card, point-of-sale transaction
- at a supermarket, or online purchase of an item

Web click-streams

Social streams

- online social networks such as Twitter
- speed and volume of the stream typically scale super-linearly with the number of actors

Network streams

- communication networks contain large volumes of traffic streams
- often mined for intrusions, outliers, or other unusual activity

# Challenges (1)

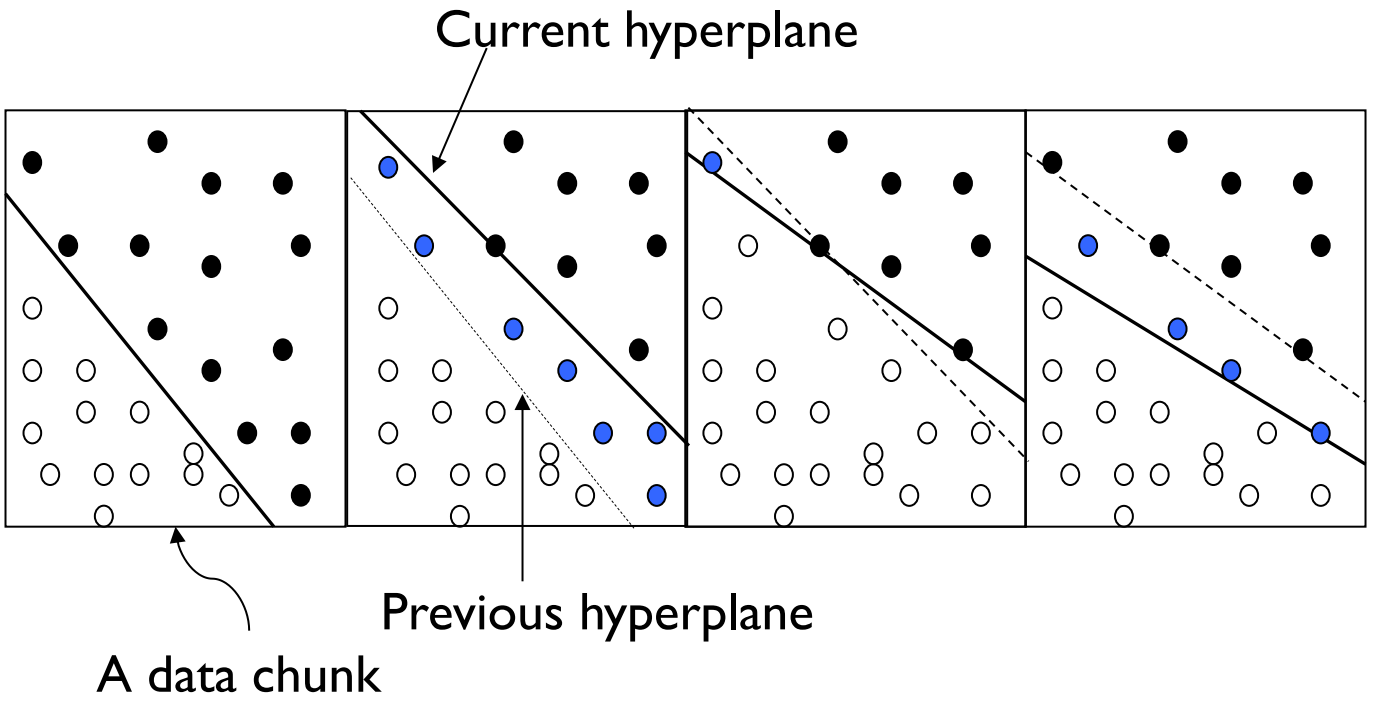
One-pass constraint

- data is generated continuously and rapidly
- it is assumed that the data can be processed only once
- archival for future processing is not possible
- prevents use of iterative mining or model building algorithms that require multiple passes over the data

Concept drift, concept evolution, feature evolution

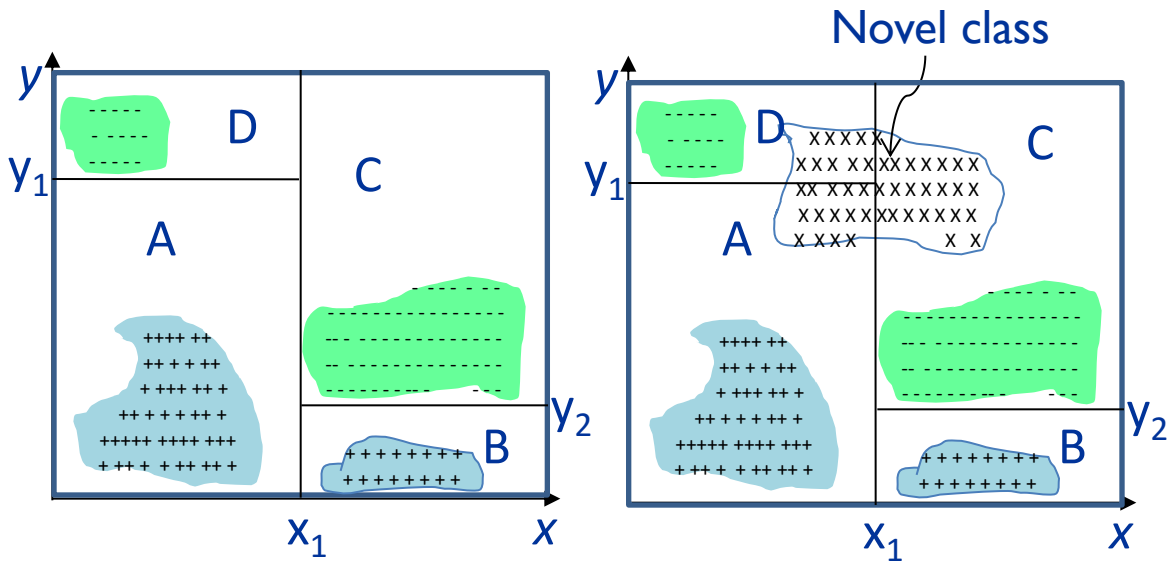
- data may evolve over time
- various statistical properties, such as correlations between attributes, correlations between attributes and class labels, and cluster distributions may change over time

# Concept Drift




Latifur Khan, et al. IBM

# Concept Evolution




Classification rules:

R1. if ($x > x_1$ and $y < y_2$) or ($x < x_1$ and $y < y_1$) then class = +

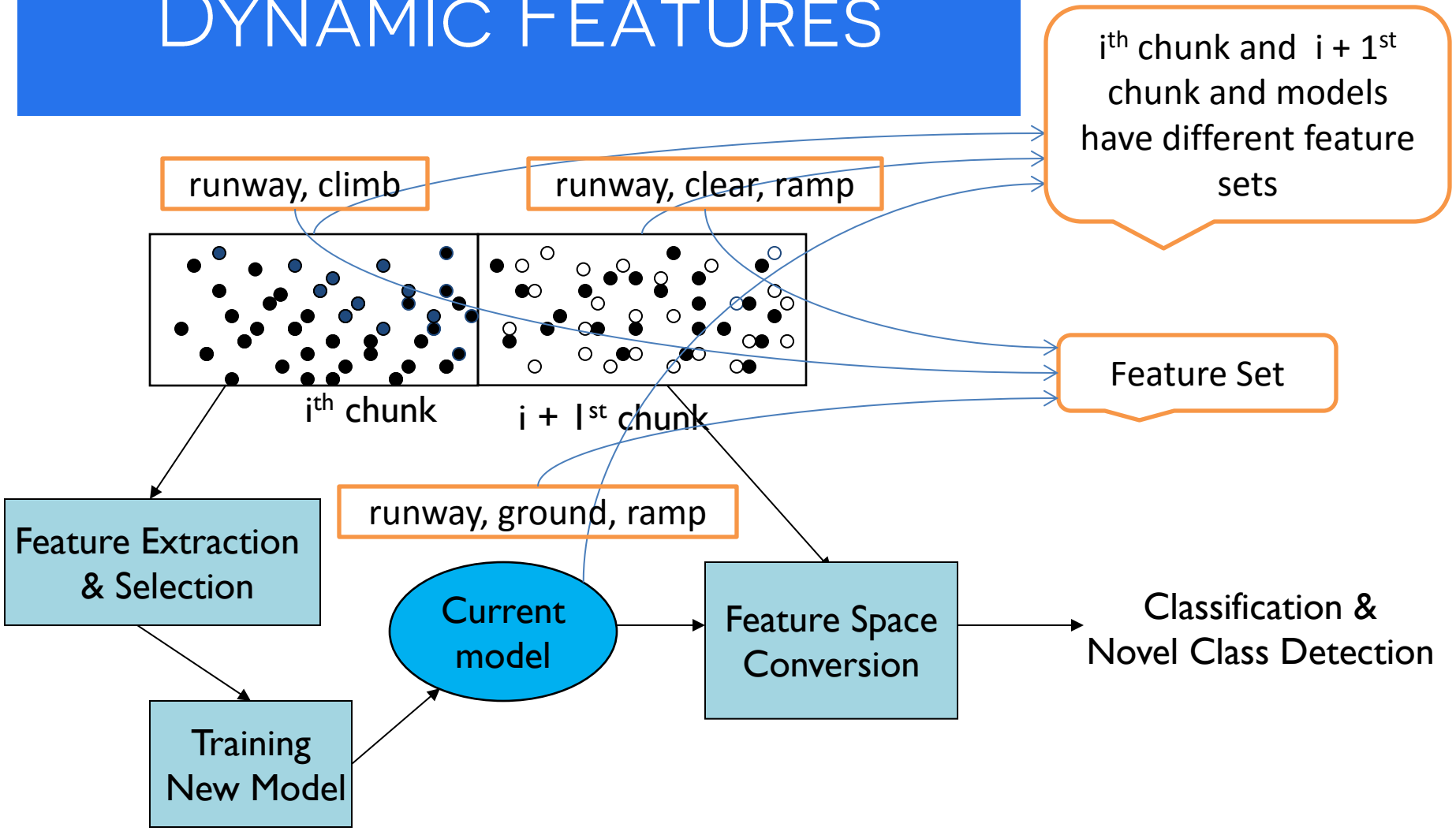
R2. if ($x > x_1$ and $y > y2$) or ($x < x_1$ and $y > y_1$) then class = -

Existing classification models misclassify novel class instances

Latifur Khan, et al. IBM

Latifur Khan, et al. IBM

# Dynamic Features




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.

# Challenges (2)

Resource constraints – the infinite storage problem

- data stream is typically generated by an external process
- user may have very little control over the process and also the arrival rate of the stream
- difficult to do online processing continuously during peak periods
- will be necessary to drop tuples that cannot be processed in a timely fashion → *load-shedding*

Massive-domain constraints

- may have a large number of distinct values
- foe example, the number of distinct pairs of e-mail addresses in an e-mail network with $10^8$ participants is of the order of $10^{16}$
- storing even simple statistics such as the counts or the number of distinct stream elements can be impossible

# Key Element – Online Synopsis

Virtually all streaming methods use an online synopsis (summary) construction approach in the mining process

- create an online synopsis that is then leveraged for mining

Many different kinds of synopsis approaches

- the nature of a synopsis highly influences the type of insights that can be mined from it.

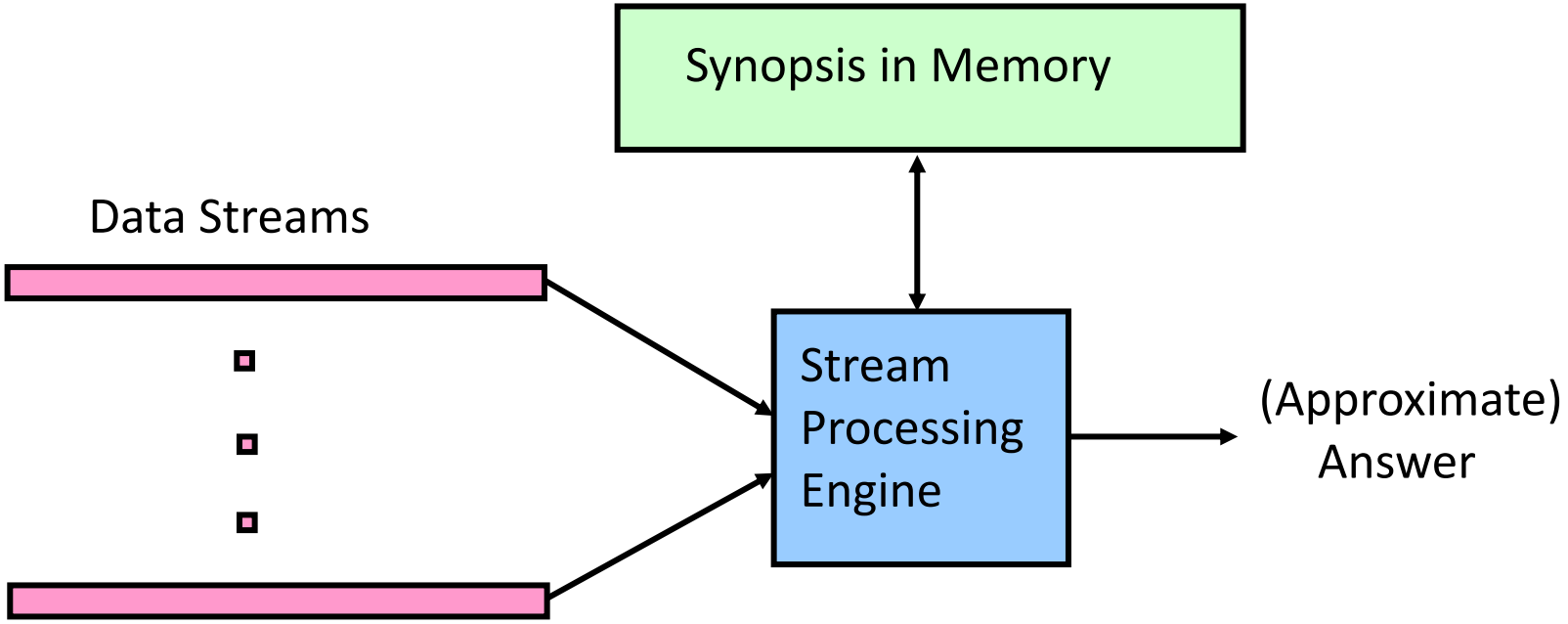
Some examples of synopsis structures:

- random samples
- bloom filters, sketches
- distinct element-counting data structures
- traditional data mining applications, such as clustering

# SYNOPSIS

Stream processing requirements

- single pass: each record is examined (sampled) at most once
- bounded storage: limited Memory (M) for storing synopsis
- real-time: per record processing time (to maintain synopsis) must be low

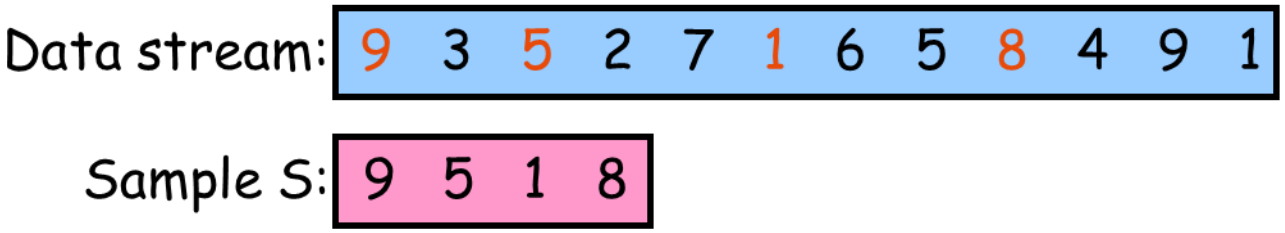


# SAMPLES

A data stream is a (massive) sequence of elements $e_1$, $e_2$, … $e_n$

Idea:

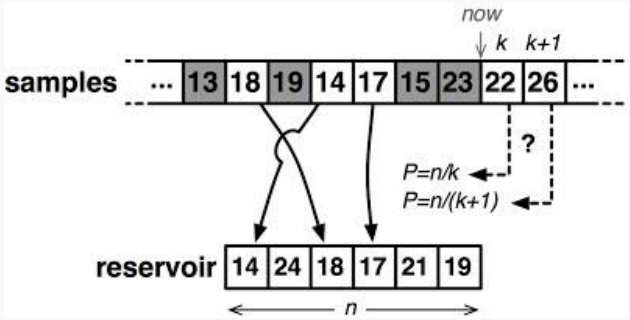
- a small random sample *S* of the data often well represents all the data
- many different ways to obtain this sample



# Reservoir Sampling

```
/*
  S has items to sample, R will contain the result
*/
ReservoirSample(S[1..n], R[1..k])
  // fill the reservoir array
  for i = 1 to k
      R[i] := S[i]

  // replace elements with gradually decreasing probability
  for i = k+1 to n
    j := random(1, i)   // important: inclusive range
    if j <= k
        R[j] := S[i]
```




Probabilities

- $k/i$ for the $i^{th}$ sample to go into the reservoir
- $1/k \cdot k/i = 1/i$ for the $j^{th}$ reservoir element to be replaced
- $k/n$ for all elements in the reservoir after $n$ has been reached
- can be shown via induction

A good algorithm to use for streaming data when $n$ is growing

# Handle Concept Drift

Use a decay-based framework to regulate the relative importance of data points in the reservoir

- achieved with the use of a *bias function*

$$f(r, n) = e^{-\lambda(n-r)}$$

*n* is the current data point, *r* is a data point acquired in the past

λ defines the bias rate and typically lies in the range [0, 1]

λ = 0 represents the unbiased case

# Sliding Window Approach

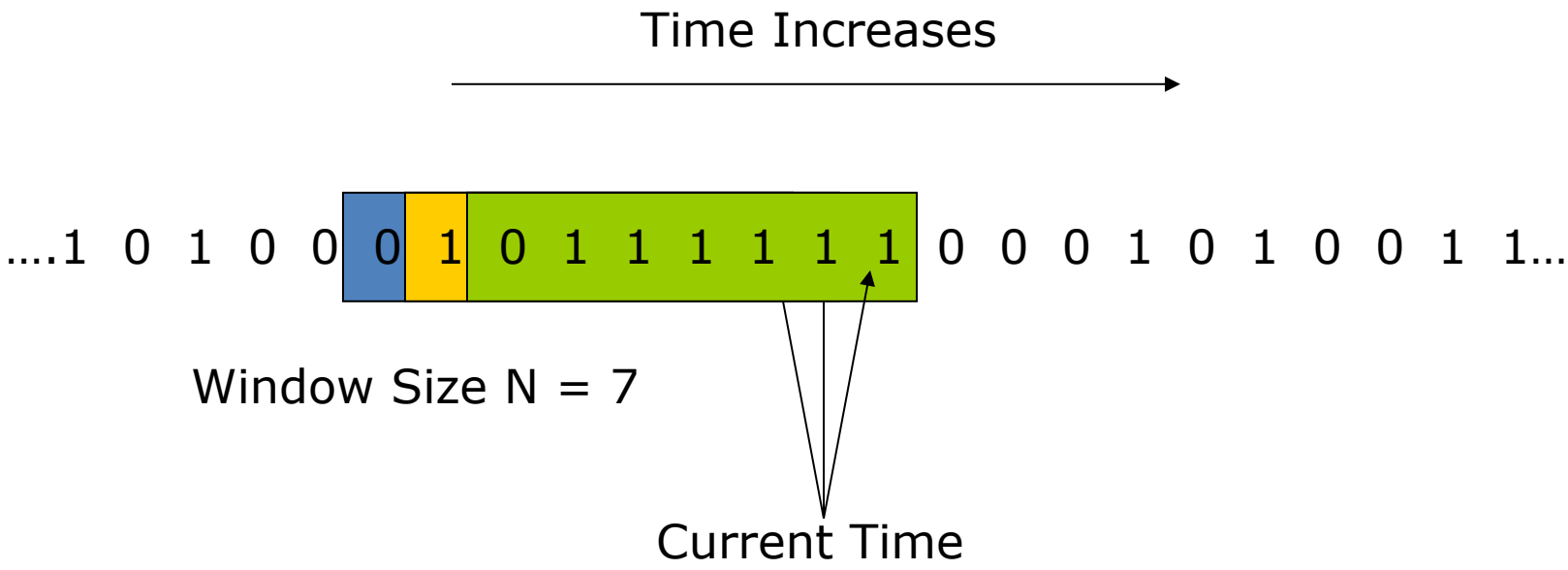
Background:

- some applications rely on ALL historical data
- but for most applications, OLD data is considered less relevant and could skew results from NEW trends or conditions
  - new processes/procedures
  - new hardware/sensors
  - new fashion trends

Sliding Windows Model

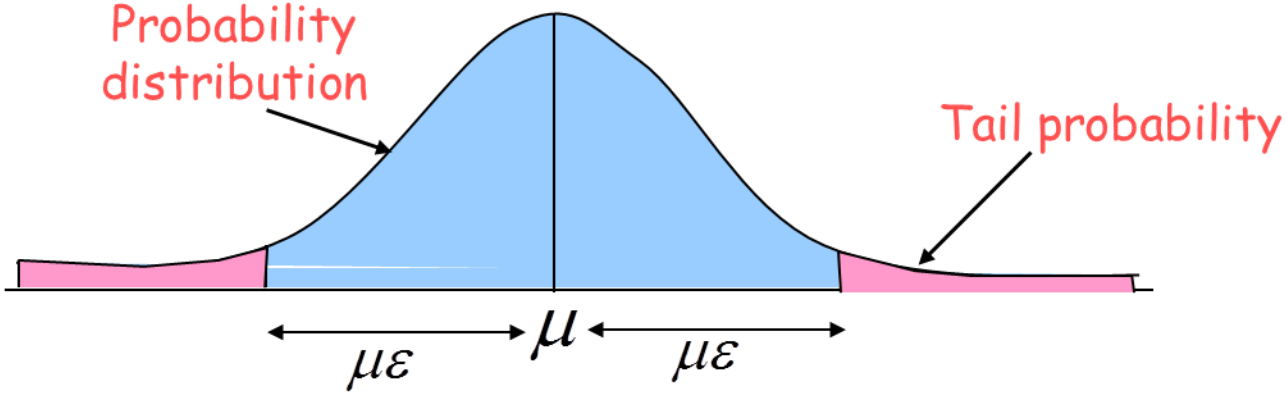
- only last "N" elements are considered
- incorporate examples as they arrive
- the record "expires" at time t+N (N is the window length)

# Sliding Window Approach



# Theoretical Bounds for Sampling

- General bounds on *tail probability* of a random variable (that is, probability that a random variable deviates far from its expectation)




- Basic Inequalities: Let X be a random variable with expectation $\mu$ and variance Var[X]. Then for any $\varepsilon > 0$

Markov: $\Pr(X \geq \varepsilon) \leq \frac{\mu}{\varepsilon}$

Chebyshev: $\Pr(|X - \mu| \geq \mu\varepsilon) \leq \frac{Var[X]}{\mu^2 \varepsilon^2}$

# Counting Samples

Effective for answering hot list queries (k most frequent values)

- sample S is a set of <value, count> pairs

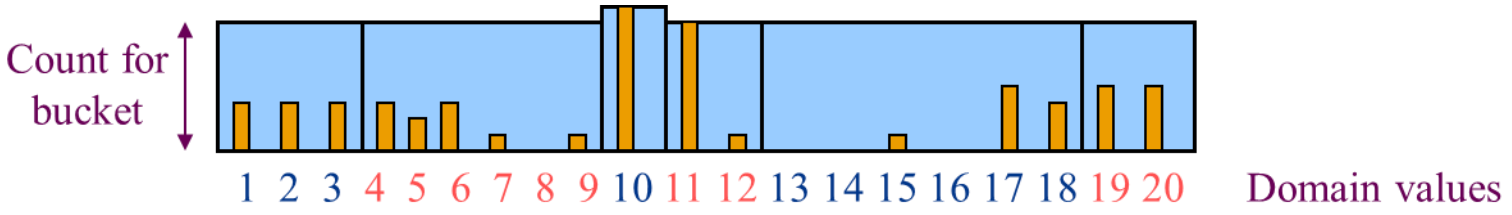
For each new stream element

- if element value in S, increment its count
- otherwise, add to S with probability 1/T
- if size of sample S exceeds M, select new threshold T' > T

# Histograms

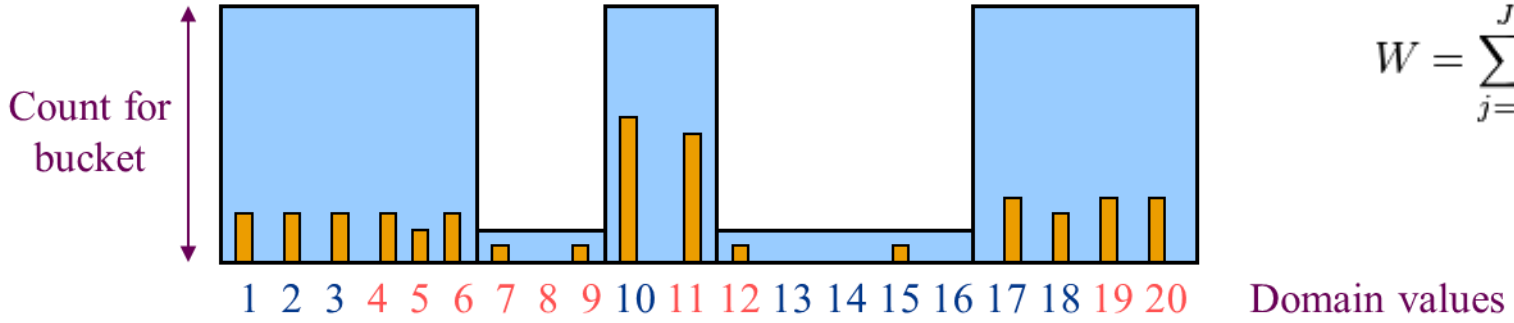
## Equi-Depth Histograms

- select buckets such that counts per bucket are equal




## V-Optimal Histograms

- select buckets to minimize frequency variance within buckets




$$W = \sum_{j=1}^{J} n_j V_j \,,$$

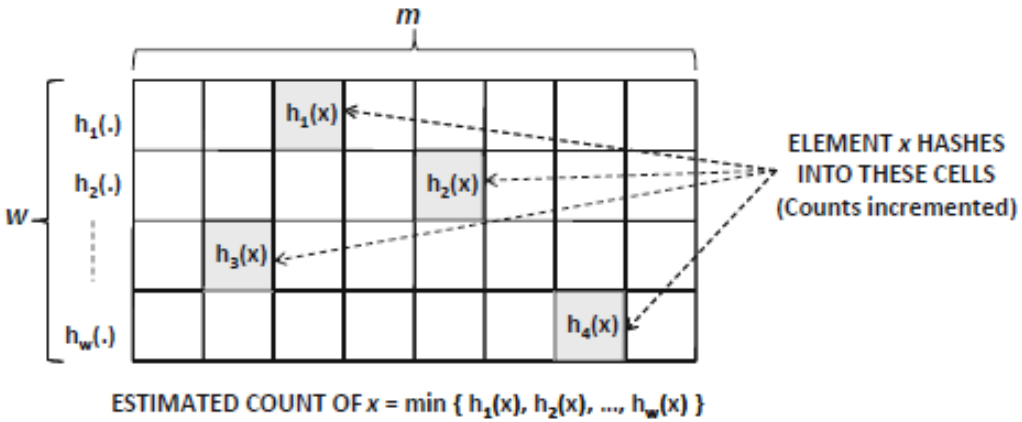
# Count-Min Sketch

Consists of a set of w different numeric arrays of length m

- space requirement is m · w cells containing numeric values
- the elements of each of the w numeric arrays are indexed starting with 0, corresponding to an index range of {0 . . m − 1}

Can be viewed as a w × m 2-dimensional array of cells

- each of the *w* numeric arrays corresponds to a hash function
- the *i*th numeric array corresponds to the *i*th hash function $h_i(\cdot)$
- all hash functions are pair-wise independent



# Construction

**Algorithm** *CountMinConstruct*(Stream: $\mathcal{S}$, Width: $w$, Height: $m$)
**begin**
  Initialize all entries of $w \times m$ array $\mathcal{CM}$ to 0;
  **repeat**
    Receive next stream element $x \in \mathcal{S}$;
    **for** $i = 1$ to $w$ **do**
      Increment $(i, h_i(x))$th element in $\mathcal{CM}$ by 1;
  **until** end of stream $\mathcal{S}$;
  **return** $\mathcal{CM}$;
**end**

# Query

**Algorithm** *CountMinQuery*(Element: $y$, Count-min Sketch: $\mathcal{CM}$)
**begin**
  Initialize $Estimate = \infty$;
  **for** $i = 1$ to $w$ **do**
    $Estimate = \min\{Estimate, V_i(y)\}$;
  { $V_i(y)$ is the count of the $(i, h_i(y))$th element in $\mathcal{CM}$ }
  **return** $Estimate$;
**end**

Each value $V_i(y)$ is an overestimate of the true frequency of y because of potential collisions

- therefore, the tightest possible estimate may be obtained by using the minimum value $\min_i\{Vi(y)\}$ over the different hash functions

A related algorithm is the bloom filter for sets

# Flajolet-Martin Sketch

A probabilistic counting algorithm

Used to estimate number of distinct elements in a large file originally

- uses little memory
- single pass only
- can give information on unusual observations

Overall idea

- the more different elements we see in the stream, the more different hash-values we shall see
- as we see more different hash-values, it becomes more likely that one of these values will be "unusual."

# Flajolet-Martin Sketch

Hash function $h$: map $n$ elements to $\log_2 n$ bits uniformly

bit($y$, $k$) = $k^{\text{th}}$ bit in the binary representation of $y$

$$y = \sum_{k \geq 0} bit(y,k) \cdot 2^k$$

$$\rho(y) = \min_{k \geq 0}[bit(y,k)] \neq 0 \quad \text{if } y>0$$

$$\rho(y) = L \quad \text{if } y=0$$

# Flajolet–Martin Sketch

```
for (i:=0 to L-1) do BITMAP[i]:=0;
for (all x in M) do
        begin
                index:=ρ(h(x));
                if BITMAP[index]=0 then
                        BITMAP[index]:=1;
        end
```

$R$ := the largest *index* in *BITMAP* whose value equals to 1

*Estimate* := $2^R$

# Flajolet-Martin Sketch

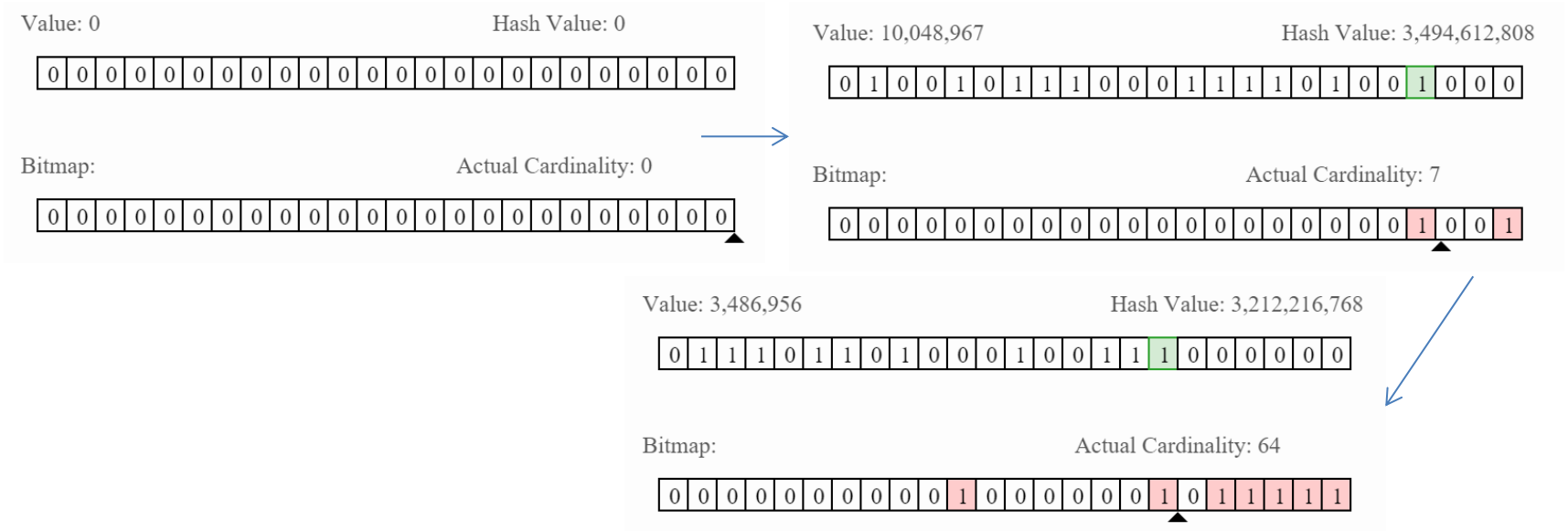
If the final *BITMAP* looks like this:

0000,0000,1100,1111,1111,1111

The left most 1 appears at position 15

We say there are around $2^{15}$ distinct elements in the stream

Example (generated with http://content.research.neustar.biz/blog/runs.html



# Flajolet-Martin Sketch

The probability that a given stream element a has h(a) ending in at least r 0's is $2^{-r}$

Given m distinct elements in the stream, the probability that none of them has tail length at least r is $(1 - 2^{-r})^m$

After some manipulations we get the following:

Assuming r is reasonably large, the probability of not finding a stream element with as many as r 0's at the end of its hash value is $e^{-m2^{-r}}$

We can conclude:

- if m is much larger than $2^r$ , then the probability that we shall find a tail of length at least r approaches 1.
- if m is much less than $2^r$ , then the probability of finding a tail of length at least r approaches 0.

This confirms that the estimate of m is $2^R$ (R being the largest tail length for any stream element)

# Frequent Pattern Mining

Can use the various synopsis structures for this

Reservoir sampling is the most flexible

1. Maintain a reservoir sample S from the data stream
2. Apply a frequent pattern mining algorithm to the reservoir sample S and report the patterns

# CluStream Clustering

The concept drift in an evolving data stream changes the clusters significantly over time

- need a clustering algorithm that can deal with this
- CluStream is such an algorithm

CluStream's online microclustering clustering stage

- processes the stream in real time to continuously maintain summarized but detailed (micro-)cluster statistics of the stream

CluStream's  offline macroclustering stage

- further summarizes these detailed clusters
- provides the user with a more concise understanding of the clusters over different time horizons and levels of temporal granularity.

# Microclustering Algorithm

There are k microclusters

- a new data point either needs to be absorbed by a microcluster, or it needs to be put in a cluster of its own

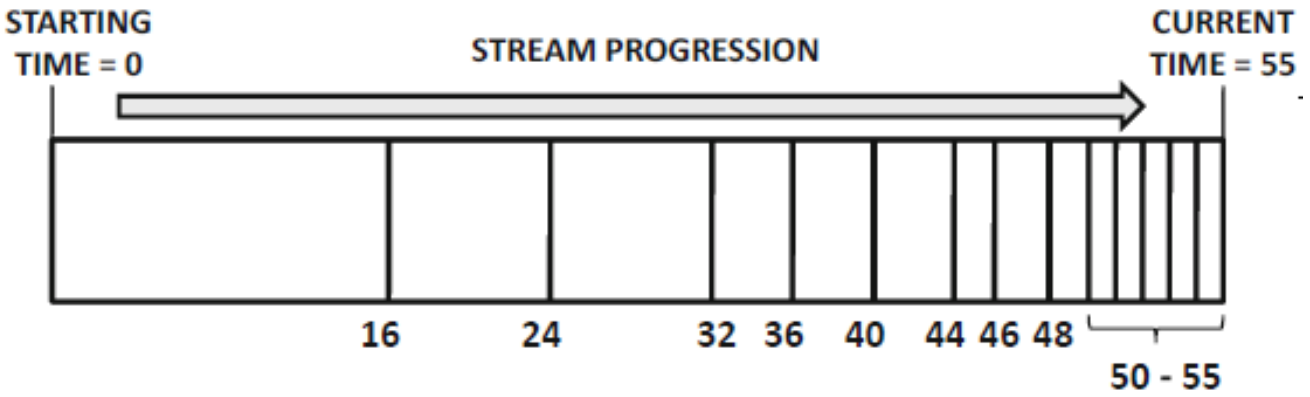
Algorithm

- determine distance of the new data point to all current microcluster centroids
- assign the point to the closest cluster and update the statistics
- if the point does not fall within the maximum boundary of any microcluster create a new microcluster
- to create this new microcluster, the number of other microclusters must be reduced by 1 to free memory availability
- achieve this by either deleting an old microcluster or merging two of the older clusters
- decide by examining the staleness (using the time stamp statistics) of the different clusters, and the number of points in them
- determine whether one of them is “sufficiently” stale to merit removal
- if no microcluster is stale, then a merging of the two microclusters is initiated

# Pyramidal Time Frame

Store microclusters statistics periodically to enable time horizon-specific analysis of the clusters

- the microcluster snapshots are stored at varying levels of granularity depending on the recency of the snapshot



# Other Stream Mining Issues

Streaming outlier (anomaly) detection

- use time windows and k-nearest neighbor scores
- new concepts or trends can manifest themselves as outliers in the onset

Streaming classifiers

- the *Hoeffding tree* is constructed incrementally by growing the tree simultaneously with stream arrival.