



# Introduction to Data Mining

## Mining Data Streams-3

**U Kang**  
**Seoul National University**



# Outline

- ➔  Estimating Moments
- Counting Frequent Items



# Generalization: Moments

- Suppose a stream has elements chosen from a set  $A$  of  $N$  values
- Let  $m_i$  be the number of times value  $i$  occurs in the stream
- The  $k^{\text{th}}$  *moment* is 
$$\sum_{i \in A} (m_i)^k$$
- E.g., for a stream  $(x, y, x, y, z, z, z, x, z)$ ,
  - The 2<sup>nd</sup> moment is  $3^2 + 2^2 + 4^2 = 29$
  - (x appears 3 times, y appears 2 times, z appears 4 times)



# Special Cases

$$\sum_{i \in A} (m_i)^k$$

- **0<sup>th</sup> moment** = number of distinct elements
  - The problem considered in the last lecture
- **1<sup>st</sup> moment** = count of the numbers of elements  
= length of the stream
  - Easy to compute
- **2<sup>nd</sup> moment** = *surprise number S* =  
a measure of how uneven the distribution is



# Example: Surprise Number

- Stream of length 100
- 11 distinct values
- Item counts: **10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9** Surprise  
 **$S = 910$**
- Item counts: **90, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1** Surprise  
 **$S = 8,110$**



# Problem Definition

- **Q: Given a stream, how can we estimate k-th moments efficiently, with small memory space?**
- **A: AMS method**



# AMS Method

- AMS method works for all moments
- Gives an unbiased estimate
- We first concentrate on the 2<sup>nd</sup> moment  $S$
- We pick and keep track of many variables  $X$ :
  - For each variable  $X$  we store  $X.el$  and  $X.val$ 
    - $X.el$  corresponds to the item  $i$
    - $X.val$  corresponds to the **count** of item  $i$
  - Note this requires a count in main memory, so number of  $X$ s is limited
- Our goal is to compute  $S = \sum_i m_i^2$



# One Random Variable ( $X$ )

## ■ How to set $X.val$ and $X.el$ ?

- Assume stream has length  $n$  (we relax this later)
- Pick some random time  $t$  ( $t < n$ ) to start, so that any time is equally likely
- If the stream has item  $i$  at time  $t$ , **we set  $X.el = i$**
- Then we maintain count  $c$  ( $X.val = c$ ) of the number of  $i$ s in the stream starting from the chosen time  $t$

## ■ Then the estimate of the 2<sup>nd</sup> moment ( $\sum_i m_i^2$ ) is:

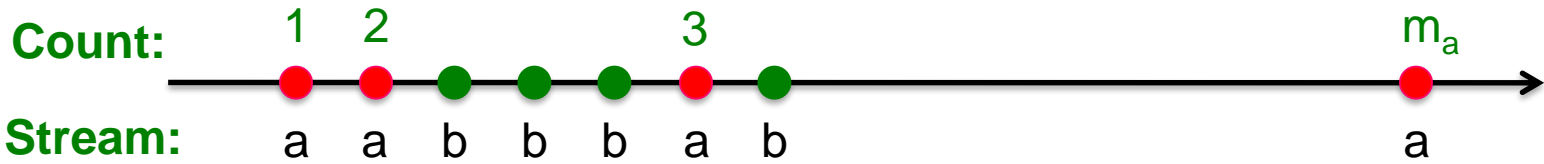
$$S = f(X) = n(2 \cdot c - 1)$$

- Note, we will keep track of multiple  $X$ s, ( $X_1, X_2, \dots, X_k$ ) and our final estimate will be  $S = \frac{1}{k} \sum_{j=1}^k f(X_j)$





# Expectation Analysis



- 2<sup>nd</sup> moment is  $S = \sum_i m_i^2$
- $c_t$  ... number of times item at time  $t$  appears from time  $t$  onwards ( $c_1=m_a, c_2=m_a-1, c_3=m_b$ )

$$E[f(X)] = \frac{1}{n} \sum_{t=1}^n n(2c_t - 1)$$

$$= \sum_i (1 + 3 + 5 + \dots + 2m_i - 1)$$

$m_i$  ... total count of item  $i$  in the stream (we are assuming stream has length  $n$ )

Group times by the value seen

Time  $t$  when the last  $i$  is seen ( $c_t=1$ )

Time  $t$  when the penultimate  $i$  is seen ( $c_t=2$ )

Time  $t$  when the first  $i$  is seen ( $c_t=m_i$ )





# Higher-Order Moments

- For estimating  $k^{\text{th}}$  moment we essentially use the same algorithm but change the estimate:
  - For  $k=2$  we used  $n (2 \cdot c - 1)$
  - For  $k=3$  we use:  $n (3 \cdot c^2 - 3c + 1)$  (where  $c=X.\text{val}$ )
- Why?
  - For  $k=2$ : Remember we had  $(1 + 3 + 5 + \dots + 2m_i - 1)$  and we showed terms  $2c-1$  (for  $c=1, \dots, m$ ) sum to  $m^2$ 
    - $m^2 = \sum_{c=1}^m c^2 - \sum_{c=1}^m (c-1)^2 = \sum_{c=1}^m (2c-1)$
    - So:  $2c-1 = c^2 - (c-1)^2$
  - For  $k=3$ :  $c^3 - (c-1)^3 = 3c^2 - 3c + 1$
- Generally: Estimate =  $n (c^k - (c-1)^k)$



# Combining Samples

## ■ In practice:

- Compute  $f(X) = n(2c - 1)$  for as many variables  $X$  as you can fit in memory
- Average them

## ■ Problem: Streams never end

- We assumed there was a number  $n$ , the number of positions in the stream
- But real streams go on forever, so  $n$  is a variable – the number of inputs seen so far



# Streams Never End: Fixups

- **(1)**  $f(\mathbf{X}) = n(2c-1)$  has  $n$  as a factor – keep  $n$  separately; just hold the count  $c$  in  $\mathbf{X}$
- **(2)** Suppose we can only store  $k$  counts. We must throw some  $\mathbf{X}$ s out as time goes on:
  - **Objective:** Each starting time  $t$  is selected with probability  $k/n$
  - **Solution: (fixed-size sampling = reservoir sampling!)**
    - Choose the first  $k$  elements for  $k$  variables
    - When the  $n^{\text{th}}$  element arrives ( $n > k$ ), choose it with probability  $k/n$
    - If you choose it, throw one of the previously stored variables  $\mathbf{X}$  out, with equal probability



# Outline

Estimating Moments

  **Counting Frequent Items**



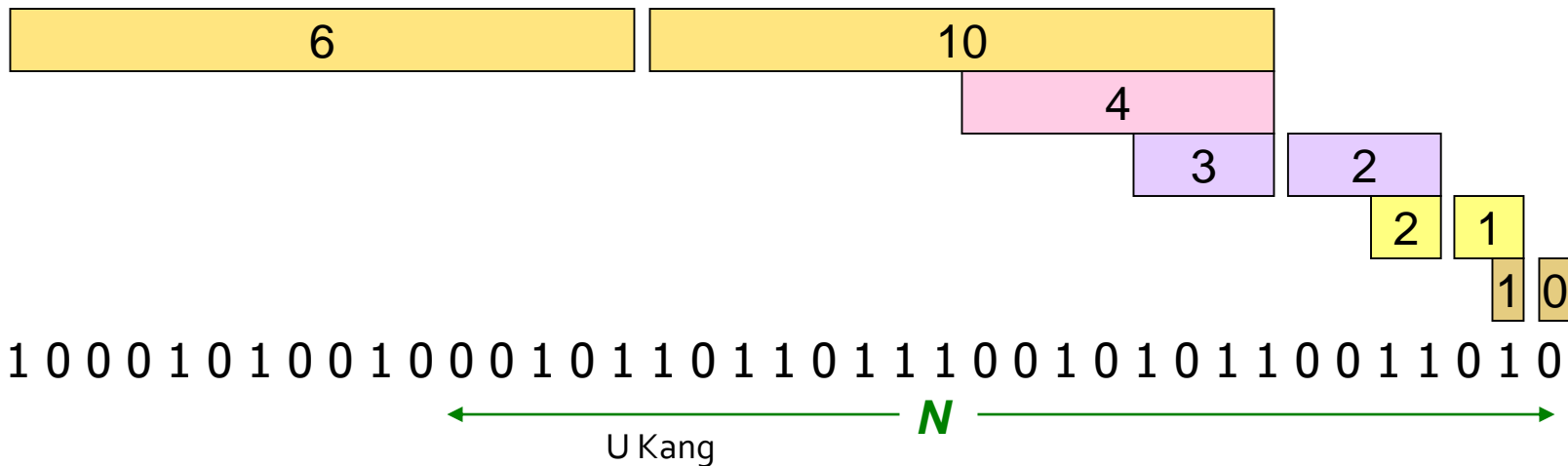
# Counting Itemsets

- New Problem: Given a stream, how can we find **recent frequent items** (= which appear more than  $s$  times in the window) efficiently?



# Counting Itemsets

- **New Problem:** Given a stream, which items appear more than  $s$  times in the window?
- **Possible solution:** Think of the stream of baskets as one binary stream per item
- **1** = item present; **0** = not present
- Use **DGIM** to estimate counts of **1s** for all items







# Extensions

- In principle, you could count frequent pairs or even larger sets the same way
  - One stream per itemset
  - E.g., for a basket  $\{i, j, k\}$ , assume 7 independent streams:  $(i)$   $(j)$   $(k)$   $(i, j)$   $(i, k)$   $(j, k)$   $(i, j, k)$
- Drawback:
  - Number of itemsets is way too big



# Exponentially Decaying Windows

- **Exponentially decaying windows: A heuristic for selecting likely frequent item(sets)**
  - **What are “currently” most popular movies?**
    - Instead of computing the raw count in last  $N$  elements
    - Compute a **smooth aggregation** over the whole stream
- If stream is  $a_1, a_2, \dots$  and we are taking the sum of the stream, take the answer at time  $t$  to be: =  
$$\sum_{i=1}^t a_i (1 - c)^{t-i}$$
  - $c$  is a constant, presumably tiny, like  $10^{-6}$  or  $10^{-9}$
- **When new  $a_{t+1}$  arrives:**  
Multiply current sum by  $(1-c)$  and add  $a_{t+1}$



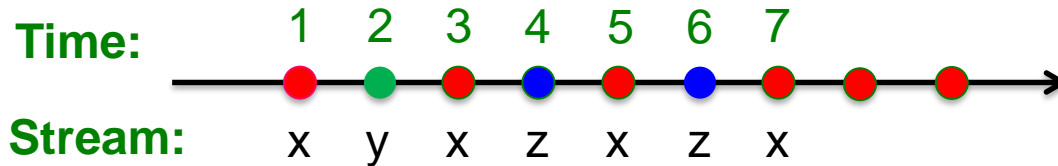
# Example: Counting Items

- If each  $a_i$  is an “item” we can compute the **characteristic function** of each possible item  $x$  as an Exponentially Decaying Window
  - That is:  $\sum_{i=1}^t \delta_i \cdot (1 - c)^{t-i}$   
where  $\delta_i = 1$  if  $a_i = x$ , and  $0$  otherwise
  - Imagine that for each item  $x$  we have a binary stream (**1** if  $x$  appears, **0** if  $x$  does not appear)
  - **New item  $x$  arrives:**
    - Multiply all counts by **(1-c)**
    - Add **+1** to count for element  $x$
    - *Remove all items whose weights  $< s$*
- Call this sum the “weight” of item  $x$

Note: Assume we are interested in items with weights  $\geq s$



# Example: Counting Items



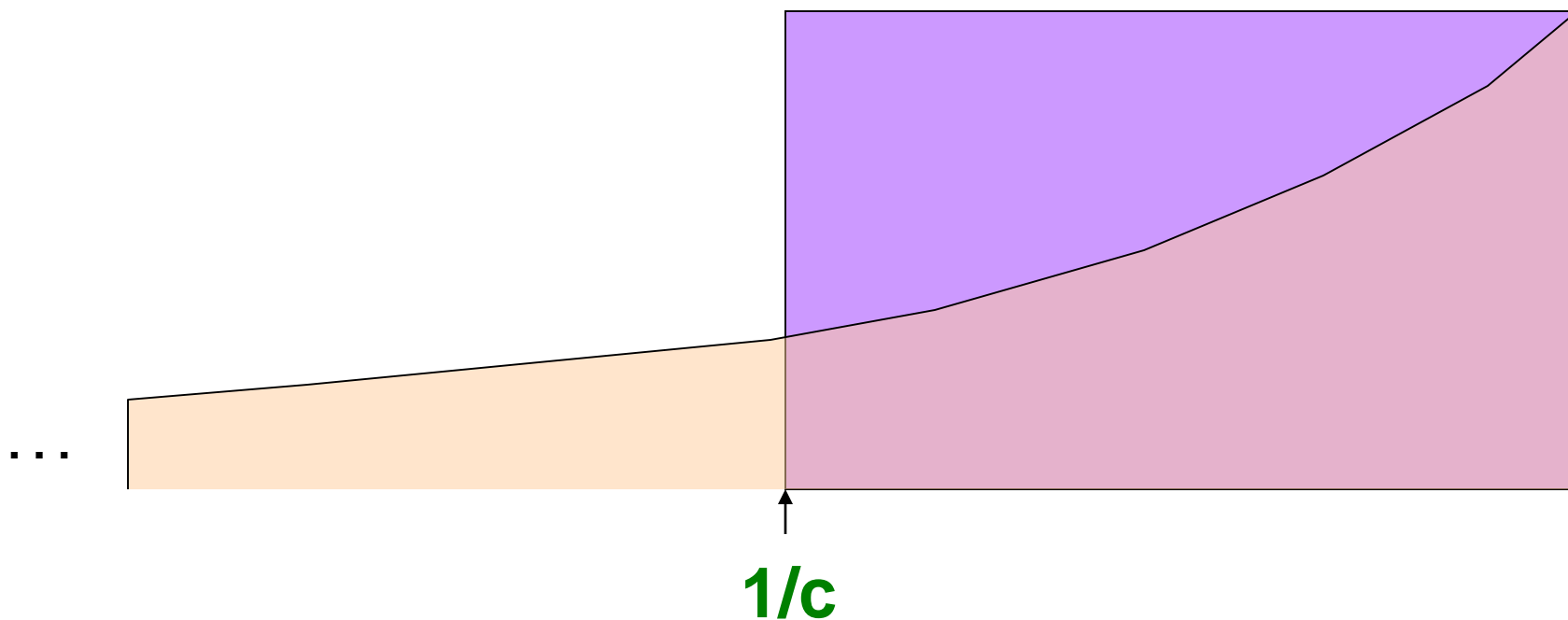
$$\sum_{i=1}^t \delta_i \cdot (1 - c)^{t-i}$$

Assume  $c = 0.2$ ,  
Keep items with weights  $\geq 1/2$

- (T1) x: 1
- (T2) x:  $0.8 \cdot 1$ , y: 1
- (T3) x:  $0.8 \cdot 0.8 + 1$ , y:  $0.8 \cdot 1$
- (T4) x:  $0.8 \cdot 1.64$ , y:  $0.8 \cdot 0.8$ , z = 1
- (T5) x:  $1.312 + 1$ , y:  $0.8 \cdot 0.64 = 0.512$ , z:  $0.8 \cdot 1$
- (T6) x:  $0.8 \cdot 2.312$ , y:  $0.8 \cdot 0.512$ , z:  $0.8 \cdot 0.8 + 1$ 
  - Remove y
- (T7) x:  $0.8 \cdot 1.8496 + 1$ , z:  $0.8 \cdot 1.64$
- ...



# Sliding vs. Decaying Windows



- **Important property:** Sum over all weights  $\sum_t (1 - c)^t$  is  $1/[1 - (1 - c)] = 1/c$



# Example: Counting Items

- What are “currently” most popular movies?
- Suppose we want to find movies of weight  $> \frac{1}{2}$ 
  - **Important property:** Sum over all weights  $\sum_t (1 - c)^t$  is  $1/[1 - (1 - c)] = 1/c$
- **Thus:**
  - There cannot be more than  $2/c$  movies with weight  $> \frac{1}{2}$
- So,  $2/c$  is a limit on the number of movies being counted at any time (if we remove movies whose weight  $\leq \frac{1}{2}$ )



# Extension to Itemsets

- Assume at each time we are given an itemset
  - E.g.,  $\{i, j, k\}$ ,  $\{k, x\}$ ,  $\{i, j\}$
- Count (some) itemsets in an E.D.W.
  - **What are currently “hot” itemsets?**
  - **Problem:** Too many itemsets to keep counts of all of them in memory



# Extension to Itemsets

- Count (some) itemsets in an E.D.W.
  - What are currently “hot” itemsets?
  - **Problem:** Too many itemsets to keep counts of all of them in memory
- **When a basket **B** comes in:**
  - Multiply all counts by **(1-c)**
  - For uncounted items in **B**, create new count
  - Add **1** to count of any item in **B** and to any **itemset** contained in **B** that is already being counted
  - **Remove items and itemsets whose counts  $< \frac{1}{2}$**
  - Initiate new counts (next slide)





# Initiation of New Counts

- Start a count for an itemset  $S \subseteq B$  if every proper subset of  $S$  had a count prior to arrival of basket  $B$ 
  - **Intuitively:** If all subsets of  $S$  are being counted this means they are “frequent/hot” and thus  $S$  has a potential to be “hot”
- **Example:**
  - Start counting  $S=\{i, j\}$  iff both  $i$  and  $j$  were counted prior to seeing  $B$
  - Start counting  $S=\{i, j, k\}$  iff  $\{i, j\}$ ,  $\{i, k\}$ , and  $\{j, k\}$  were all counted prior to seeing  $B$



# Summary – Stream Mining

- Important tools for stream mining
  - Sampling from Data Stream (Reservoir Sampling)
  - Querying Over Sliding Windows (DGIM method for counting the number of 1s or sums in the window)
  - Filtering a Data Stream (Bloom Filter)
  - Counting Distinct Elements (Flajolet-Martin)
  - Estimating Moments (AMS method; surprise number)
  - Counting Frequent Itemsets (exponentially decaying windows)



# Questions?