More Stream-Mining

Counting Distinct Elements

Computing "Moments"

Frequent Itemsets

Elephants and Troops

Exponentially Decaying Windows

Counting Distinct Elements

- Problem: a data stream consists of elements chosen from a set of size *n*. Maintain a count of the number of distinct elements seen so far.
- Obvious approach: maintain the set of elements seen.

Applications

- How many different words are found among the Web pages being crawled at a site?
  - Unusually low or high numbers could indicate artificial pages (spam?).
- How many different Web pages does each customer request in a week?

Using Small Storage

- Real Problem: what if we do not have space to store the complete set?
- Estimate the count in an unbiased way.
- Accept that the count may be in error, but limit the probability that the error is large.

Flajolet-Martin* Approach

- Pick a hash function h that maps each of the n elements to at least $\log_2 n$ bits.
- For each stream element a, let $r(a)$ be the number of trailing 0's in $h(a)$.
- Record R = the maximum $r(a)$ seen.
- Estimate = 2^R.

* Really based on a variant due to AMS (Alon, Matias, and Szegedy)

Why It Works

- The probability that a given $h(a)$ ends in at least r 0's is 2^{-r}.
- If there are m different elements, the probability that $R \geq r$ is $1 - \boxed{(\boxed{1 - 2^{-r}})^m}$.

Prob. all h(a)'s end in fewer than *r* 0's.

Prob. a given h(a) ends in fewer than *r* 0's.

Why It Works – (2)

- Since 2^{-r} is small, $1 - (1-2^{-r})^m \approx 1 - e^{-m2^{-r}}$.
- If $2^r >> m$, $1 - (1 - 2^{-r})^m \approx 1 - (1 - m2^{-r}) \approx m/2^r \approx 0$.

  First 2 terms of the Taylor expansion of e^x
- If $2^r << m$, $1 - (1 - 2^{-r})^m \approx 1 - e^{-m2^{-r}} \approx 1$.
- Thus, 2^R will almost always be around m.

Why It Doesn't Work

- E(2^R) is actually infinite.
  - Probability halves when R -> R+1, but value doubles.
- Workaround involves using many hash functions and getting many samples.
- How are samples combined?
  - Average? What if one very large value?
  - Median? All values are a power of 2.

Solution

- Partition your samples into small groups.
- Take the average of groups.
- Then take the median of the averages.

Generalization: Moments

- Suppose a stream has elements chosen from a set of n values.
- Let m_i be the number of times value i occurs.
- The k th *moment* is the sum of $(m_i)^k$ over all i.

Special Cases

- 0th moment = number of different elements in the stream.
  - The problem just considered.
- 1st moment = count of the numbers of elements = length of the stream.
  - Easy to compute.
- 2nd moment = *surprise number* = a measure of how uneven the distribution is.

Example: Surprise Number

- Stream of length 100; 11 values appear.
- Unsurprising: 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9. Surprise # = 910.
- Surprising: 90, 1, 1, 1, 1, 1, 1, 1 ,1, 1, 1. Surprise # = 8,110.

AMS Method

- Works for all moments; gives an unbiased estimate.
- We'll just concentrate on 2nd moment.
- Based on calculation of many random variables X.
  - Each requires a count in main memory, so number is limited.

One Random Variable

- Assume stream has length *n*.
- Pick a random time to start, so that any time is equally likely.
- Let the chosen time have element *a* in the stream.
- $X = n * ($(twice the number of *a*'s in the stream starting at the chosen time) – 1).
  - Note: store *n* once, count of *a*'s for each *X*.

Expected Value of X

- 2nd moment is $\Sigma_a (m_a)^2$.
- $E(X) = (1/n)\big(\Sigma_{\text{all times } t}\, n$ * (twice the number of times the stream element at time t appears from that time on) – 1$\big)$.
- $= \Sigma_a (1/n)(n)(1+3+5+\ldots+2m_a-1)$.
- $= \Sigma_a (m_a)^2$.

Group times by the value seen

Time when the last a is seen

Time when the penultimate a is seen

Time when the first a is seen

Combining Samples

- Compute as many variables X as can fit in available memory.
- Average them in groups.
- Take median of averages.
- Proper balance of group sizes and number of groups assures not only correct expected value, but expected error goes to 0 as number of samples gets large.

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.

Fixups

1. The variables X have n as a factor – keep n separately; just hold the count in X.
2. Suppose we can only store k counts. We must throw some X's out as time goes on.
   - Objective: each starting time t is selected with probability k/n.

Solution to (2)

- Choose the first k times for k variables.
- When the n^{th} element arrives ($n > k$), choose it with probability k / n.
- If you choose it, throw one of the previously stored variables out, with equal probability.

New Topic: Counting Itemsets

- 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 1's for all items.

Extensions

- In principle, you could count frequent pairs or even larger sets the same way.
  - One stream per itemset.
- Drawbacks:
  1. Only approximate.
  2. Number of itemsets is way too big.

Approaches

1. "Elephants and troops": a heuristic way to converge on unusually strongly connected itemsets.
2. Exponentially decaying windows: a heuristic for selecting likely frequent itemsets.

Elephants and Troops

- When Sergey Brin wasn't worrying about Google, he tried the following experiment.
- Goal: find unusually correlated sets of words.
  - "*High Correlation*" = frequency of occurrence of set >> product of frequencies of members.

Experimental Setup

- The data was an early Google crawl of the Stanford Web.
- Each night, the data would be streamed to a process that counted a preselected collection of itemsets.
  - If $\{a, b, c\}$ is selected, count $\{a, b, c\}$, $\{a\}$, $\{b\}$, and $\{c\}$.
  - "Correlation" = $n^2 \times \#abc/(\#a \times \#b \times \#c)$.
    - n = number of pages.

After Each Night's Processing . . .

1. Find the most correlated sets counted.
2. Construct a new collection of itemsets to count the next night.
   - All the most correlated sets ("*winners*").
   - Pairs of a word in some winner and a random word.
   - Winners combined in various ways.
   - Some random pairs.

After a Week . . .

- The pair {"elephants", "troops"} came up as the big winner.
- Why? It turns out that Stanford students were playing a Punic-War simulation game internationally, where moves were sent by Web pages.

New Topic: Mining Streams Versus Mining DB's

- Unlike mining databases, mining streams doesn't have a fixed answer.
- We're really mining in the "Stat" point of view, e.g., "Which itemsets are frequent in the underlying model that generates the stream?"

Stationarity

Our assumptions make a big difference:

1. Is the model *stationary* ?
   - I.e., are the same statistics used throughout all time to generate the stream?
2. Or does the frequency of generating given items or itemsets change over time?

Some Options for Frequent Itemsets

1. Run periodic experiments, like E&T.
   - Like SON – itemset is a candidate if it is found frequent on any "day."
   - Good for stationary statistics.
2. Frame the problem as finding all frequent itemsets in an "exponentially decaying window."
   - Good for nonstationary statistics.

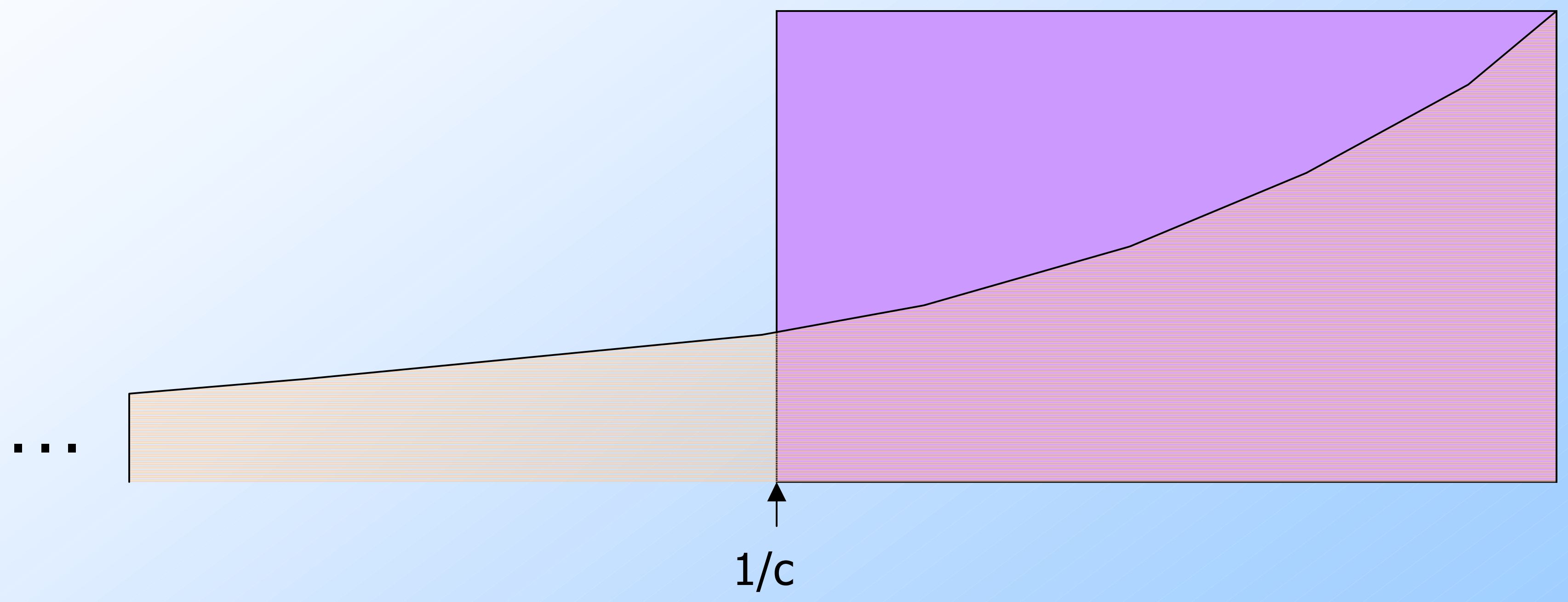
Exponentially Decaying Windows

- If stream is a_1, a_2,… and we are taking the sum of the stream, take the answer at time t to be: $\Sigma_{i = 1,2,\ldots,t}\, a_i\, e^{-c(t-i)}$.
- c is a constant, presumably tiny, like 10^{-6} or 10^{-9}.

Example: Counting Items

- If each a_i is an "item" we can compute the *characteristic function* of each possible item x as an E.D.W.
- That is: $\Sigma_{i = 1,2,\dots,t}\, \delta_i\, e^{-c(t-i)}$, where $\delta_i = 1$ if $a_i = x$, and 0 otherwise.
  - Call this sum the "*count*" of item x.

Sliding Versus Decaying Windows



Counting Items – (2)

- Suppose we want to find those items of weight at least ½.
- Important property: sum over all weights is $1/(1 - e^{-c})$ or very close to $1/[1 - (1 - c)] = 1/c$.
- Thus: at most $2/c$ items have weight at least ½.

Extension to Larger Itemsets*

- Count (some) itemsets in an E.D.W.
- When a basket *B* comes in:
  1. Multiply all counts by (1-*c*);
  2. For uncounted items in *B*, create new count.
  3. Add 1 to count of any item in *B* and to any counted itemset contained in *B*.
  4. Drop counts < ½.
  5. Initiate new counts (next slide).

* Informal proposal of Art Owen

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.
- Example: Start counting $\{i, j\}$ iff both i and j were counted prior to seeing B.
- Example: Start counting $\{i, j, k\}$ iff $\{i, j\}$, $\{i, k\}$, and $\{j, k\}$ were all counted prior to seeing B.

How Many Counts?

- Counts for single items $\leq (2/c)$ times the average number of items in a basket.
- Counts for larger itemsets = ??. But we are conservative about starting counts of large sets.
  - If we counted every set we saw, one basket of 20 items would initiate 1M counts.