CSCI8980 Algorithmic Techniques for Big Data September 12, 2013

Lecture 2

Dr. Barna Saha *Scribe: Matt Nohelty*

Overview

We continue our discussion on data streaming models where streams of elements are coming in and main memory space is not sufficient to hold all the data. We begin by discussing the Chernoff Bound and demonstrating it's proof. We then look at the Universal Hash Family and discuss pairwise, k-wise, and fully independent hash functions. Next, we dive deeper into algorithms used to count distinct items in a stream and discuss two algorithms and analyze them.

1 Chernoff Bound

The Chernoff Bound is commonly used to show randomization algorithms produce results of acceptable quality or to determine the number of runs needed to acheive a result of a certain probability. Many data streaming algorithms have components of randomization so the Chernoff Bound is frequently used with these algorithms. The Chernoff Bound produces tighter bounds than the Markov Inequality or Chebyshev Inequality but it requires assumptions that those two do not. The Chernoff Bound requires it's input to be independent Bernoulli random variables which the other two inequalities do not.

Theorem 1 (The Chernoff Bound). *Let $X_1, X_2 ... X_n$ be n independent Bernoulli random variables with $\Pr(X_i = 1) = p_i$. Let $X = \sum X_i$. Hence,*

$$E[X] = E\left[\sum X_i\right] = \sum E[X_i] = \sum \Pr(X_i = 1) = \sum p_i = \mu \textit{(say)}.$$

Then the Chernoff Bound says for any $\epsilon > 0$

$$\begin{aligned} \Pr(X > (1+\epsilon)\mu) &\leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{\epsilon}}\right)^{\mu} \textit{and} \\ \Pr(X < (1-\epsilon)\mu) &\leq \left(\frac{e^{-\epsilon}}{(1-\epsilon)^{1-\epsilon}}\right)^{\mu} \end{aligned}$$

When $0 < \epsilon < 1$ the above expression can be further simplified to

$$\begin{aligned} \Pr(X > (1+\epsilon)\mu) &\leq e^{\frac{-\mu\epsilon^2}{3}} \textit{and} \\ \Pr(X < (1-\epsilon)\mu) &\leq e^{\frac{-\mu\epsilon^2}{2}} \end{aligned}$$

Hence

$$\Pr(|X - \mu| > \epsilon\mu) \leq 2e^{\frac{-\mu\epsilon^2}{3}}$$

Proof of the Chernoff Upper Bound The upper bound of the Chernoff Bound states:

$$\Pr(X > (1+\epsilon)\mu) \quad \le \quad e^{\frac{-\mu\epsilon^2}{3}}$$

Proof.

$$\begin{aligned} Pr(e^{tx} \geq e^{t(1+\epsilon)\mu}) &\text{ for any } t > 0 \\ Pr(e^{tx} \geq e^{t(1+\epsilon)\mu}) \leq & \qquad \frac{E[e^{tx}]}{e^{t(1+\epsilon)\mu}} \text{by Markov Inequality} \end{aligned}$$

Expand x in the numerator:

$$\begin{aligned} E[e^{tx}] &= E[e^{t\sum xi}] \\ &= E[e^{tx_1}e^{tx_2}...e^{tx_n}] \text{ all are independent by base assumption in Chernoff Bound} \\ &= \prod_{i=1}^{n} E[e^{tx_i}] \\ &= \prod_{i=1}^{n}[p_ie^t + (1-p_i)] \\ &= \prod_{i=1}^{n}[1 + p_i(e^t - 1)] \\ &\leq \prod_{i=1}^{n}[e^{p_i}(e^t - 1)] \text{ because } e^x > 1 + x \\ &= e^{\sum_{i=1}^{n} p_i(e^t-1)} \\ &= e^{(e^t-1)\mu} \end{aligned}$$

Using the simplified numerator in the Chernoff Bound yields $\frac{E[e^{e^t-1}]}{e^{t(1+\epsilon)\mu}}$

Differentiating to find t where the above is minimized results in $t = ln(1+\epsilon)$

Returning to the upper bound with t. Expand x in the numerator:

$$\begin{aligned} Pr(x \geq (1+\epsilon)\mu) \leq & \quad \frac{e^{(e^{(ln(1+\epsilon)}-1)\mu}}{e^{1+\epsilon)ln(1+\epsilon)\mu}} \\ = & \quad \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\mu} \\ = & \quad e^{-\mu[(1+\epsilon)ln(1+\epsilon)-\epsilon]} \\ = & \quad e^{-\mu\left[(1+\epsilon)\left[\epsilon-\frac{\epsilon^2}{2}+\frac{\epsilon^3}{3}...\right]-\epsilon\right]} \\ = & \quad e^{-\mu\left[\frac{\epsilon^2}{2}-\frac{\epsilon^3}{6}...\right]} \\ \leq & \quad e^{-\mu\left[\frac{\epsilon^2}{2}-\frac{\epsilon^3}{6}\right]} \\ = & \quad e^{-\mu\frac{\epsilon^2}{2}\left(1-\frac{\epsilon}{2}\right)} \\ \leq & \quad e^{-\mu\frac{\epsilon^2}{3}} \text{ which is the upper bound of the Chernoff Bound} \end{aligned}$$

□

The proof of the lower bound of the Chernoff Bound can be found using similar logic as the proof of the upper bound.

2 Universal Hash Family

The Univeral Hash Family is a family of hash functions $H = \{h | h : [N] \Rightarrow [M]\}$ is called a pairwise independent family of hash functions if for all $i \neq j \in [N]$ and any $k, l \in [M]$

$$Pr_{h \Leftarrow H}[h(i) = k \wedge h(j) = l] = \frac{1}{M^2} \text{ is a strongly universal hash family} \tag{1}$$

A hash function is pairwise independent if property 1 holds. This definition can be extend to form k-wise hash functions as well. K-wise hash functions are important because they allow for efficient construction of hash families. Fully independent hash functions generally require large space requirements.

Hash functions are uniform over $[M]$

$$Pr_{h \Leftarrow H}[h(i) = k] = \frac{1}{M} \tag{2}$$

$$Pr_{h \Leftarrow H}[h(i) = h(j)] = \frac{1}{M} \text{ is a weakly universal hash family} \tag{3}$$

To Construct a pairwise independent hash family:
Let p be a prime.
For any $a, b \in \mathbb{Z}_p = \{0, 1, 2, ...p - 1\}$, define $h_{a,b} : \mathbb{Z}_p \Rightarrow \mathbb{Z}_p$ by $h_{a,b}(x) = ax + b mod p$.
The resulting collection of functions $H = \{h_{a,b} | a, b \in \mathbb{Z}_p\}$ is a pairwise independent hash family.

3 Counting Distinct Items

Given a stream of data a, find the total number of distinct items in the stream. For the purpose of this discussion, we assume the stream to too large to be stored in main memory.

$a = a_1 a_2 ... a_m$
$a_i = (j, \mu)$ where $j \in [1, n]$ and $\mu \geq 1$
m represents the number of elements in the stream
n represents the maximum number of distinct elements that could be in the stream.

The goal is to find the actual number of distinct elements, DE. However, because we cannot store a in main memory, we must approximate DE. This approximation will be denoted DE'. We want to find DE' such that the following constraint holds with probablilty $\geq (1 - \delta)$.

$$(1-\epsilon)DE \leq DE' \leq DE(1+\epsilon) \text{ for } \epsilon > 0 \tag{4}$$

4 Algorithm - Count Distinct Items

The following algorithm attempts to guess the actual value of DE by looping through exponentially growing values of t. For each guess, the algorithm calls $ESTIMATE$ which returns YES if there are at least t distinct values, otherwise it returns NO. $ESTIMATE$ returns the correct answer with probability $\geq (1-\delta)$ as we will see later.

Following the for loop, we have a list of YES/NO values corresponding to each t. The algorithm returns the largest value of t which has a value YES.

Algorithm 1 COUNT DISTINCT ITEMS$[a, \epsilon, \delta]$

```
$\epsilon' = \epsilon/2$
for $t = 1, \lceil (1+\epsilon') \rceil, \lceil (1+\epsilon')^2 \rceil, ... \lceil (1+\epsilon')^{log_{1+\epsilon'} n} \rceil$ do
    $\delta' = \frac{\epsilon' \delta}{logn}$ {Run in parallel}
    $b_t = ESTIMATE(a, t, \epsilon', \delta')$ {$b_t$ is a boolean variable YES/NO}
end for
return the smallest value of $t$ such that $b_{t-1}$ = YES and $b_t$ = NO if no such $t$ exists, return $n$
```

Below is an example of the output produced by the for loop in Algorithm 1. This is the likely output produced in the case where $(1+\epsilon') \leq DE \leq (1+\epsilon')^2$.

$t = 1 \Rightarrow$ YES
$t = (1+\epsilon') \Rightarrow$ YES
$t = (1+\epsilon')^2 \Rightarrow$ NO
$t = (1+\epsilon')^3 \Rightarrow$ NO
...
$t = n \Rightarrow$ NO

As the example illustrates, the resulting DE' satisfies the constraint: $(1-\epsilon)DE \leq DE' \leq DE(1+\epsilon)$

Proof. For each t, we get the correct result with probability $\delta' = \frac{\epsilon' \delta}{logn}$ and there are $log_{1+\epsilon'} n$ different values for t.

$$\begin{aligned}
Pr(\text{Error for any } t) \leq \quad & \delta' \\
Pr(\text{Error in at least one } t) \leq \quad & \sum_t Pr(\text{Error for any } t) \\
\leq \quad & log_{1+\epsilon'} n \delta' \\
\approx \quad & \frac{1}{\epsilon'} \text{log} n \delta' \\
= \quad & \delta \\
Pr(\text{No error in any } t) < \quad & 1-\delta
\end{aligned}$$

□

5 Algorithm - ESTIMATE

$ESTIMATE$ randomly selects $\frac{c}{\epsilon'^2} \log \frac{1}{\delta'}$ hash functions from a fully-independent hash family. The hash function h is of the form $h : [1...n] \Rightarrow [1...t]$.

We then compute the hash value for every value of in the stream for each hash function. If the hash function ever returns 1, use YES for this t, otherwise use NO. Finally, count the number of NO values and if it's greater than or equal to $\frac{c}{\epsilon'^2} \log \frac{1}{\delta'}$, return NO, otherwise return YES.

$ESTIMATE$ returns the correct answer with probabily $\geq (1-\delta)$ because there are $\frac{c}{\epsilon'^2} \log \frac{1}{\delta'}$ hash functions used and the most common answer wins. This minimizes the impact of the randomization in the hash functions.

Algorithm 2 [ESTIMATE(a, t, ϵ', δ')]

```
count ⇐ 0
for $t = 1, \frac{c}{\epsilon'^2} \log \frac{1}{\delta'}$ do
  Select a hash function $h_i$ uniformly and randomly from a fully-independent hash family $H$
  {run in parallel}
  $b_t^i \Leftarrow$ NO
  repeat
    Consider the current element in the stream $a$, say $a_i = (j, \mu)$
    if $h_i(j) == 1$ then
      $b_t^i \Leftarrow$ YES, BREAK
    end if
  until $a$ is exhasted
  if $b_t^i ==$ NO then
    $count = count + 1$
  end if
end for
if $count \geq \frac{1}{e} \frac{c}{\epsilon'_2} \log \frac{1}{\delta'}$ then
  return NO
else
  return YES
end if
```

Proof. The goal is to return YES when $DE > (1+\epsilon)t$ and to return NO when $DE < (1-\epsilon)t$.

Let h_i be the i_{th} run through the for loop.
There are k runs where $k = \frac{c}{\epsilon'^2} \log \frac{1}{\delta'}$

$Pr(h_i(j) = 1) = \frac{1}{t}$ by definition of h

$$\begin{aligned} Pr(\text{Return NO for the } i^{th} \text{ run}) = &\quad Pr(\text{None of the distinct elements are mapped to 1 by } h_i) \\ = &\quad (1 - \frac{1}{t})^{DE} \end{aligned}$$

$\square$

Lemma 2. *Consider the* i^{th} *round of* $ESTIMATE(a, t, \epsilon', \delta')$ *for any* $i \in [\frac{c}{\epsilon^2}\ log\ \frac{1}{\delta'}]$

If $DE > (1+\epsilon)t$ *and* $\epsilon < 1$ *then* $Pr[b_t^i == NO] \leq \frac{1}{e} - \frac{\epsilon}{2e}$

$$\begin{aligned} Pr(i^{th}\ run\ returns\ NO) \leq &\ (1 - \frac{1}{t})^{(1+\epsilon)t} \\ \approx &\ e^{-(1+\epsilon)}\ when\ t\ is\ large \\ = &\ e^{-1}(1 - \epsilon + \frac{\epsilon^2}{2}...) \\ \leq &\ \frac{1}{e} - \frac{\epsilon}{e} + \frac{\epsilon^2}{2e} \\ \leq &\ \frac{1}{e} - \frac{\epsilon}{2e} \end{aligned}$$

If $DE < (1-\epsilon)t$ *and* $\epsilon < 1$ *then* $Pr[b_t^i == NO] \geq \frac{1}{e} + \frac{\epsilon}{2e}$

$$\begin{aligned} Pr(i^{th}\ run\ returns\ NO) \geq &\ (1 - \frac{1}{e})^{(1-\epsilon)t} \\ \geq &\ \frac{1}{e} + \frac{\epsilon}{2e}\ by\ the\ same\ logic\ as\ above \end{aligned}$$

Lemma 3. *Demostrates the bounds of the error in Algorithm 1.*
If $DE > (1+\epsilon')t$ *then* $Pr[b_t == NO] \leq \frac{\delta'}{2}$
If $DE < (1-\epsilon')t$ *then* $Pr[b_t == YES] \leq \frac{\delta'}{2}$

$$\begin{aligned} Pr(\text{Algorithm 1 returns NO}) = &\ Pr(x > \frac{k}{e}) \text{ because we return NO if more than } \tfrac{k}{e} \text{ runs return NO} \\ \leq &\ e^{-\epsilon'^2 ck} \end{aligned}$$

Define a random variable $x_i = 1$ if algorithm returns NO, otherwise $x_i = 0$.

$$\begin{aligned} x = &\ \sum x_i \\ E[x] = &\ \sum E[x] \\ = &\ \sum Pr(x_i = 1) \\ = &\ \sum Pr(i^{th} \text{ run returns NO}) \\ \leq &\ k(\frac{1}{e} + \frac{\epsilon}{2e}) \text{ by Lemma 2} \end{aligned}$$

Re-write $Pr(x > \frac{k}{e})$ in the form of the Chernoff Bound

$$\begin{aligned} &Pr(x > (1+\epsilon')E[x]) \\ (1+\epsilon')k(\tfrac{1}{e} - \tfrac{\epsilon'}{2e}) = \;& \frac{k}{e} \text{ by using the value of E[x] from above} \\ (1+\epsilon')(1-\tfrac{\epsilon'}{2}) = \;& 1 \\ 1+\epsilon = \;& \frac{2}{2-\epsilon'} \end{aligned}$$

$$\begin{aligned} Pr(x > \tfrac{k}{e}) \leq \;& e^{-\epsilon^2 \frac{\mu}{3}} \\ = \;& e^{-\epsilon^2 ck} \\ \leq \;& \frac{\delta'}{2} \text{ using } k = \tfrac{c}{\epsilon'^2} log\tfrac{1}{\delta'} \end{aligned}$$

The lower bound can be demonstrated with similar logic to what was done to prove the upper bound above. This shows that when run enough times, $\frac{c}{\epsilon'^2} log\frac{1}{\delta'}$, we can minimize the probability for error to a sufficient level.

Lemma 4. *If* $|DE - t| > \epsilon' t$ *then* $Pr[ERROR] \leq \delta'$

Using the Union Bound, we know the total $Pr[ERROR]$ cannot exceed the sum of the $Pr[ERROR]$ of the lower bound and the $Pr[ERROR]$ of the upper bound.

$$\tfrac{\delta'}{2} + \tfrac{\delta'}{2} \leq \delta'$$

Lemma 5. *For all* t *such that* $|DE - t| > \epsilon' t$ *then* $Pr[ERROR] \leq \delta$

Theorem 6. *Algorithm 1 returns an estimate of DE within* $(1 \pm \epsilon)$ *with probability* $\geq (1-\delta)$.

Theorem 6 shows that this algorithm to count distinct items has achieved our goal of finding an algorithm that computes DE' under the following accuracy constraint: $(1-\epsilon)DE \leq DE' \leq DE(1+\epsilon)$ for $\epsilon > 0$ and does so with probability $\geq (1-\delta)$.

6 Space and Time Complexity of Count Distinct Items

Space Complexity: $O(\frac{1}{\epsilon^3} \log n(\log\frac{1}{\delta} + \log \log n + \log\frac{1}{\epsilon}))$
Time Complexity: $O(\frac{1}{\epsilon^3} \log n(\log\frac{1}{\delta} + \log \log n + \log\frac{1}{\epsilon}))$

Ignoring constants, there are $\frac{1}{\epsilon}\log n$ copies that need to be stored and each requires 1 bit.
The space complexity of $ESTIMATE$ is $\frac{1}{\epsilon^2} \log \frac{logn}{\epsilon\delta}$
Expanding this space complexity yields: $\frac{1}{\epsilon^2}(loglogn + log\frac{1}{\epsilon} + log\frac{1}{\delta})$
Combining the space complexity and number of copies yields the total space complexity:

$$O(\frac{1}{\epsilon^3}\log n(\log\frac{1}{\delta} + \log\ \log n + \log\frac{1}{\epsilon})) \tag{5}$$

The time complexity can be computed in the same way as the space complexity.

In practice, the space and time dependency on ϵ^3 is generally problematic. The optimal lower bound on space complexity for counting distinct items in a stream was shown to be $\Omega(\frac{1}{\epsilon^2} + \log\ n)$.

References

[1] Daniel M. Kane, Jelani Nelson and David P. Woodruff. An Optimal Algorithm for the Distinct Elements Problem. PODS 2010: 41-52.