COMP 441 — Large Scale Machine Learning.	Jan 24, 2016
Assignment 1	
Due By: 8th Feb 2016	*Due By: 8th Feb 2016*

This assignment is worth total 25 points. Your assignment is due by 12:00pm, 8th Feb, 2016 either by email or in class.

Some Logistics

1. You are encouraged to discuss but you should write your own solutions in detail.
2. Solve either the theory part (Section 1) or the programming part (Section 2).
3. If you do both theory and programming, the maximum of the two will be considered as your final score. You will get 10 bonus points if you solve both the parts perfectly.
4. You will get 15 bonus points if you implement the programming part using GPUs and show significant speedups.
5. You can only claim one bonus from 3 and 4.

1 Theory Question (25 Point)

1.1 Problem 1 (8 Points): Random Sampling and Chernoff Bound

We want to infer the fraction of population f that uses iphone. We can interview a sample $\bar{S} = \{a_1,\ a_2, ...,\ a_k\}$ of size k, chosen randomly from the population $S = \{1,\ 2, ...,\ n\}$. We compute the fraction $\bar{f}$ of the people, that uses iphone, from the k sample. We will show that this is a very good estimate, if k is big enough.
What is the size of $\bar{S}$ (or the value to k) to ensure that

$$Pr(|f - \bar{f}| \geq \epsilon) \leq \delta$$

Hint: Show that $k \geq \frac{2+\epsilon}{\epsilon^2} \log \frac{2}{\delta}$ (independent of n!)
Tool: 2-sided Chernoff bound If $X_i \sim Bernoulli(p)$, and $\bar{X} = \sum_{i=1}^{n} X_i$ where X_i's are i.i.d., then

$$Pr(|\bar{X} - pn| \geq \epsilon pn) \leq 2e^{-\frac{\epsilon^2 pn}{2+\epsilon}}$$

Get yourself familiar with the implications. What is more costly? Getting high probability or getting sharp confidence interval $[f - \epsilon, f + \epsilon]$ for $\bar{f}$?

1.2 Problem 2 (10 Points): Analyze the Count-Min Sketch Algorithm

Let $S = \{s_1,\ s_2, ..., s_T\}$ be the stream with every element $s_i \in \{I_1,\ I_2, ..., I_N$ coming from universe of items. Let f_i denotes the frequency of item I_i observed in the stream S, i.e., $f_i = |\{j | s_j = I_i\}|$.
Show that the estimate $\widehat{f}_i$ returned by count-min sketch satisfies the following for $w = \lceil \frac{e}{\epsilon} \rceil$ and $d = \log \frac{1}{\delta}$

$$f_i \leq \widehat{f}_i \leq f_i + \epsilon \sum_{i=1}^{T} s_i,$$

with probability $1 - \delta$.

CMS Algorithm The *Count-Min Sketch* CMS is a data structure with a two-dimensional array of counter cells M of width w and depth d, shown in Figure 1. It is accessed via d pairwise-independent hash functions $h_1, h_2, ..., h_d : \{1, 2, ..., N\} \mapsto \{1, 2, ..., w\}$. Each counter is

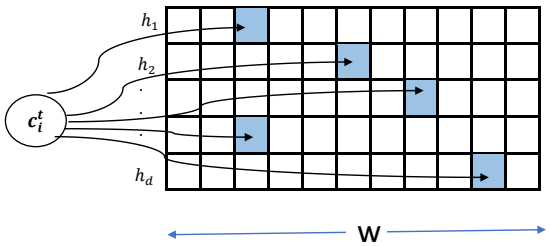

Figure 1: Count-Min Sketch Data Structure

initialized with zero, and every update s_i is added for all $d = \log \frac{1}{\delta}$ rows to counters $M(j, h_j(s_i))$, where $j = \{1, 2, ..., d\}$. A query for the count of item I_i reports the minimum of the corresponding d counters i.e., $\min_{j \in \{1,2,...,d\}} M(j, h_j(I_i))$.

Algorithm 1 *Count-Min Sketch*

Input:
$w \leftarrow \lceil \frac{e}{\epsilon} \rceil$
$d \leftarrow \log \frac{1}{\delta}$
$M \leftarrow d \times w$ array initialized to 0

Update s_i
for $j = 1$ **to** d **do**
    $M(j, h_j(s_j)) \leftarrow M(j, h_j(s_j)) + s_i$
end for

Query for counts of item I_i
return $\widehat{f}_i \leftarrow \min_{j \in \{1,2,...,d\}} M(j, h_j(I_i))$

1.3 Problem 3 (7 Points)

We want to generate a random permutation of $[1,\ 2, ...,\ n]$, where n is potentially huge. We are given a very long array A[] = new int[n] containing numbers from 1 to n. We have to return a shuffled array. We can only scan the array once and can use only few temporary variables. Here is an algorithm that works

- Scan the array one-by-one starting from A[0].
- If we are at element A[i], generate a random number j between 0 and i, and switch elements A[i] and A[j]. (In case (i = j) nothing happens).

Prove that this scheme generates a random permutation.

2 Programming Question (25 Point)

Implement Count-Min Sketch to compute the frequency (counts) of words (or n-grams) over time. This will give a nice visualization of how the usage of words evolve over time. Your aim is to build something like Google N-gram Viewer. (no need for user interface, just the algorithm) `https://books.google.com/ngrams/`
(Use your favorite programming language.)

1. Download the 12GB (compressed) Wikipedia Dump (`https://en.wikipedia.org/wiki/Wikipedia:Database_download`).
2. From the metadata, extract text and the time when the page was created.
3. Build a count min sketch (CMS) (with width w and depth d, suggested values $w = 2^{22}$ and $d = 8$. So you are using approx 1GB memory or RAM) which hashes both the time and the word. You can use any reasonable hash function to hash (time, word) pair to an integer, and later use a 2-universal hashing to map the generated integer to the range w (For Count Min Sketch Algorithm, refer the web or the lecture scribes or any book).
4. Read every wiki document (word by word) and insert it in the CMS sketch.
5. During the query phase, the user inputs the word w. Plot the evolution of word w over time, i.e. query the sketch with (w,t), where t is the time you are interested in and plot the counts over time.
6. You can justify working in a group of two if you do a parallel (multi-core or multi-node) implementation of count-min sketch and show performance improvements (speedup in computing the sketch) compared to a naive non-parallel implementation.
7. Bonus 15 points for GPU implementation.

Things to Ponder On: Can you use this evolution of words over time to detect timings of some interesting events ? Anything more ? maybe popularity of something ? Instead of words, you can also count frequency of phrases (n-grams), for example the query can be "Rice University" (bi-grams)