COL866: Foundations of Data Science

Ragesh Jaiswal, IITD

Algorithms for Massive Data Problems: Streaming algorithm

Streaming Algorithms

- Most of the algorithms that we have seen work in the batch setting in the sense that we can assume that the data is small enough to fit in the memory.
- Question: What if this assumption is not valid for a computational task?
- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).

Streaming Algorithms

- Most of the algorithms that we have seen work in the batch setting in the sense that we can assume that the data is small enough to fit in the memory.
- Question: What if this assumption is not valid for a computational task?
- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).
- Question: How much memory is reasonable for processing data in the streaming setting?

Streaming Algorithms

- Most of the algorithms that we have seen work in the batch setting in the sense that we can assume that the data is small enough to fit in the memory.
- Question: What if this assumption is not valid for a computational task?
- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).
- Question: How much memory is reasonable for processing data in the streaming setting?
  - Suppose the data items can be represented by numbers in the set $\{1, ..., m\}$ and let $b = \lceil \log m \rceil$.
  - The space should be polynomial in b and $\log n$.

Streaming Algorithms

- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).
- Question: How much memory is reasonable for processing data in the streaming setting?
  - Suppose the data items can be represented by numbers in the set $\{1, ..., m\}$ and let $b = \lceil \log m \rceil$.
  - The space should be polynomial in b and $\log n$.
- Example: Design a streaming algorithm for computing the sum of elements of the stream. That is, $\sum_{i=1}^{n} a_i$.
  - How much memory did your algorithm require?

Streaming Algorithms

- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).
- Question: How much memory is reasonable for processing data in the streaming setting?
  - Suppose the data items can be represented by numbers in the set $\{1, ..., m\}$ and let $b = \lceil \log m \rceil$.
  - The space should be polynomial in b and $\log n$.
- Example: Design a streaming algorithm for computing the sum of elements of the stream. That is, $\sum_{i=1}^{n} a_i$.
- Uniform sampling: Design a (randomized) streaming algorithm that returns a_i with probability $\frac{a_i}{\sum_{i=1}^{n} a_i}$.

- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).
- Question: How much memory is reasonable for processing data in the streaming setting?
  - Suppose the data items can be represented by numbers in the set $\{1, ..., m\}$ and let $b = \lceil \log m \rceil$.
  - The space should be polynomial in b and $\log n$.
- Example: Design a streaming algorithm for computing the sum of elements of the stream. That is, $\sum_{i=1}^{n} a_i$.
- Uniform sampling: Design a (randomized) streaming algorithm that returns a_i with probability $\frac{a_i}{\sum_{i=1}^{n} a_i}$.
  - Start with a bucket containing a_1. When a_i arrives $(i = 2, ..., n)$, replace the element of the bucket with a_i with probability $\frac{a_i}{\sum_{j=1}^{i} a_i}$ and with remaining probability leave the bucket alone.

Streaming Algorithms

- One model to address such a computational task is called the streaming model where
  - the n data items $a_1, ..., a_n$ arrive one at a time, and
  - the goal is to process the data (*calculate statistics or summarize data*) using memory much less than n (otherwise one could store and process as in batch setting).
- Question: How much memory is reasonable for processing data in the streaming setting?
  - Suppose the data items can be represented by numbers in the set $\{1, ..., m\}$ and let $b = \lceil \log m \rceil$.
  - The space should be polynomial in b and $\log n$.
- Example: Design a streaming algorithm for computing the sum of elements of the stream. That is, $\sum_{i=1}^{n} a_i$.
- Uniform sampling: Design a (randomized) streaming algorithm that returns a_i with probability $\frac{a_i}{\sum_{i=1}^{n} a_i}$.
  - Start with a bucket containing a_1. When a_i arrives $(i = 2, ..., n)$, replace the element of the bucket with a_i with probability $\frac{a_i}{\sum_{j=1}^{i} a_i}$ and with remaining probability leave the bucket alone.
  - This is called reservoir sampling.

- Consider the streaming setting where the data items $a_1, ..., a_n$ are members of the set $\{1, ..., m\}$.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Consider the streaming setting where the data items $a_1, ..., a_n$ are members of the set $\{1, ..., m\}$.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Design an algorithm that uses $O(m)$ space.

- Consider the streaming setting where the data items $a_1, ..., a_n$ are members of the set $\{1, ..., m\}$.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Design an algorithm that uses $O(m)$ space.
- Design an algorithm that uses $O(n \log m)$ space.

- Consider the streaming setting where the data items $a_1, ..., a_n$ are members of the set $\{1, ..., m\}$.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Design an algorithm that uses $O(m)$ space.
- Design an algorithm that uses $O(n \log m)$ space.
- Question: Does there exist an deterministic algorithm for this problem that uses $< m$ bits of memory?

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Design an algorithm that uses $O(m)$ space.
- Design an algorithm that uses $O(n \log m)$ space.
- Question: Does there exist an deterministic algorithm for this problem that uses $< m$ bits of memory?No
  - There are $2^m - 1$ possible subsets of $\{1, ..., m\}$ but only 2^{m-1} different states of the memory.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Design an algorithm that uses $O(m)$ space.
- Design an algorithm that uses $O(n \log m)$ space.
- Question: Does there exist an deterministic algorithm for this problem that uses $< m$ bits of memory?No
- We would like to design an algorithm that uses space logarithmic in n and m.
  - Solution: Use a randomized streaming algorithm.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let *min* denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of *min*?

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let *min* denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of *min*?
- Question: Suppose $|S| = 2$, what is the expected value of *min*?

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let *min* denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of *min*?
- Question: Suppose $|S| = 2$, what is the expected value of *min*?
- Claim: The expected value of *min* is approximately $\frac{m}{|S|+1}$.
- So, $\left(\frac{m}{min} - 1\right)$ should give a rough estimate of $|S|$. The nice property of this estimation technique is that *min* can be maintained using $\log m$ space.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let *min* denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of *min*?
- Question: Suppose $|S| = 2$, what is the expected value of *min*?
- Claim: The expected value of *min* is approximately $\frac{m}{|S|+1}$.
- So, $\left(\frac{m}{min} - 1\right)$ should give a rough estimate of $|S|$. The nice property of this estimation technique is that *min* can be maintained using $\log m$ space.
- Issue 1: If the assumption that S is chosen uniformly at random from $\{1, ..., m\}$ is not true, then the estimate can be really bad.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let *min* denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of *min*?
- Question: Suppose $|S| = 2$, what is the expected value of *min*?
- Claim: The expected value of *min* is approximately $\frac{m}{|S|+1}$.
- So, $\left(\frac{m}{min} - 1\right)$ should give a rough estimate of $|S|$. The nice property of this estimation technique is that *min* can be maintained using $\log m$ space.
- Issue 1: If the assumption that S is chosen uniformly at random from $\{1, ..., m\}$ is not true, then the estimate can be really bad.
- Solution: Use a random hash function $h : \{1, ..., m\} \to \{1, ..., M\}$ and then maintain minimum of hash values.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let min denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of min?
- Question: Suppose $|S| = 2$, what is the expected value of min?
- Claim: The expected value of min is approximately $\frac{m}{|S|+1}$.
- So, $\left(\frac{m}{min} - 1\right)$ should give a rough estimate of $|S|$. The nice property of this estimation technique is that min can be maintained using $\log m$ space.
- Issue 1: If the assumption that S is chosen uniformly at random from $\{1, ..., m\}$ is not true, then the estimate can be really bad.
- Solution: Use a random hash function $h : \{1, ..., m\} \rightarrow \{1, ..., M\}$ and then maintain minimum of hash values.
- Issue 2: Random hash function is expensive to store.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

- Let us build intuition using a hypothetical scenario.
- Suppose the set S of distinct elements is chosen uniformly at random from $\{1, ..., m\}$.
- Let *min* denote the minimum element from set S.
- Question: Suppose $|S| = 1$, what is the expected value of *min*?
- Question: Suppose $|S| = 2$, what is the expected value of *min*?
- Claim: The expected value of *min* is approximately $\frac{m}{|S|+1}$.
- So, $\left(\frac{m}{min} - 1\right)$ should give a rough estimate of $|S|$. The nice property of this estimation technique is that *min* can be maintained using $\log m$ space.
- Issue 1: If the assumption that S is chosen uniformly at random from $\{1, ..., m\}$ is not true, then the estimate can be really bad.
- Solution: Use a random hash function $h : \{1, ..., m\} \rightarrow \{1, ..., M\}$ and then maintain minimum of hash values.
- Issue 2: Random hash function is expensive to store.
- Solution: Use much cheaper pairwise independent hash function family.

Definition (Pairwise independent hash function)

A set of hash functions $H = \{h | h : \{1, ..., m\} \rightarrow \{0, 1, ..., M-1\}\}$ is called a pairwise independent iff for all $x, y \in \{1, ..., m\}$ with $x \neq y$ and all $w, z \in \{0, 1, ..., M-1\}$,

$$\mathbf{Pr}_{h \leftarrow H}[h(x) = w \text{ and } h(y) = z] = \frac{1}{M^2}.$$

- Here is a simple way to design a pairwise independent hash function family.
  - Let $M > m$ be a prime number.
  - For $a, b \in \{0, 1, ..., M-1\}$, let $h_{a,b} = (ax + b)\ (mod M)$.
  - Let $H = \{h_{a,b} | a, b \in \{0, 1, ..., M-1\}\}$.
- Claim: H defined above is a pairwise independent hash function family.

Problem

Design a streaming algorithm for computing the number of distinct a_i's in the sequence $a_1, ..., a_n$.

Algorithm

- Let $M > m$ be a prime number
- Let $H = \{h_{a,b} | a, b \in \{0, 1, ..., M-1\}\}$

`Distinct`$(a_1, ..., a_n)$
- Pick a random h from H
- Initialise $min = h(a_1)$
- For $i > 1$: update min to $h(a_i)$ iff $h(a_i) < min$
- return$(\frac{M}{min})$

Theorem

Let d be the number of distinct elements. With probability at least $(2/3 - d/M)$, we have $\frac{d}{6} \leq \frac{M}{min} \leq 6d$ where M and min are as defined in the algorithm.

Algorithm

- Let $M > m$ be a prime number
- Let $H = \{h_{a,b} | a, b \in \{0, 1, ..., M-1\}\}$

`Distinct`$(a_1, ..., a_n)$

- Pick a random h from H
- Initialise $min = h(a_1)$
- For $i > 1$: update min to $h(a_i)$ iff $h(a_i) < min$
- return($\frac{M}{min}$)

Theorem

Let d be the number of distinct elements. With probability at least $(2/3 - d/M)$, we have $\frac{d}{6} \leq \frac{M}{min} \leq 6d$ where M and min are as defined in the algorithm.

Proof sketch

- Let $b_1, ..., b_d$ be the distinct values that appear in the input.
- Let $S = \{h(b_1), ..., h(b_d)\}$ and $min = \min(S)$.
- Claim 1: $\textbf{Pr}[\frac{M}{min} > 6d] < \frac{1}{6} + \frac{d}{M}$.

Algorithm

- Let $M > m$ be a prime number
- Let $H = \{h_{a,b} | a, b \in \{0, 1, ..., M-1\}\}$

`Distinct`$(a_1, ..., a_n)$

- Pick a random h from H
- Initialise $min = h(a_1)$
- For $i > 1$: update min to $h(a_i)$ iff $h(a_i) < min$
- return$(\frac{M}{min})$

Theorem

Let d be the number of distinct elements. With probability at least $(2/3 - d/M)$, we have $\frac{d}{6} \leq \frac{M}{min} \leq 6d$ where M and min are as defined in the algorithm.

Proof sketch

- Let $b_1, ..., b_d$ be the distinct values that appear in the input.
- Let $S = \{h(b_1), ..., h(b_d)\}$ and $min = \min(S)$.
- Claim 1: $\mathbf{Pr}[\frac{M}{min} > 6d] < \frac{1}{6} + \frac{d}{M}$.
- Claim 2: $\mathbf{Pr}[[\frac{M}{min} < \frac{d}{6}] < \frac{1}{6}$.
- The theorem follows from the above two claims. □

End