Algorithms for Big Data (VIII)

Chihao Zhang

Shanghai Jiao Tong University

Nov. 8, 2019

Review

Review

Last week, we learnt a few graph streaming algorithms.

Review

Last week, we learnt a few graph streaming algorithms.

Recall that we have the following simple algorithm for counting triangles.

Review

Last week, we learnt a few graph streaming algorithms.

Recall that we have the following simple algorithm for counting triangles.

Let $\mathbf{f} = (f_T)_{T \in \binom{[n]}{3}}$ be the vector where for $T = \{x, y, z\}$, $f_T = |\{\{x, y\}, \{x, z\}, \{y, z\}\} \cap E|$.

Review

Last week, we learnt a few graph streaming algorithms.

Recall that we have the following simple algorithm for counting triangles.

Let $\mathbf{f} = (f_T)_{T \in \binom{[n]}{3}}$ be the vector where for $T = \{x, y, z\}$, $f_T = |\{\{x, y\}, \{x, z\}, \{y, z\}\} \cap E|$.

The algorithm simply returns $F_0 - 1.5F_1 + 0.5F_2$, where $F_i = \|\mathbf{f}\|_i^i$.

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5f_T^2 - 1.5f_T + \mathbf{1}[f_T \neq 0].$$

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5f_T^2 - 1.5f_T + \mathbf{1}[f_T \neq 0].$$

The "polynomial" $f(x) = 0.5x^2 - 1.5x + \mathbf{1}[x \neq 0]$ satisfies

- $f(0) = f(1) = f(2) = 0$;
- $f(3) = 1$.

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5f_T^2 - 1.5f_T + \mathbf{1}[f_T \neq 0].$$

The "polynomial" $f(x) = 0.5x^2 - 1.5x + \mathbf{1}[x \neq 0]$ satisfies

- $f(0) = f(1) = f(2) = 0$;
- $f(3) = 1$.

The multiplicative error of the algorithm is unbounded!

We can expand $F_0 - 1.5F_1 + 0.5F_2$ as

$$\sum_{T \in \binom{[n]}{3}} 0.5 f_T^2 - 1.5 f_T + \mathbf{1}[f_T \neq 0].$$

The "polynomial" $f(x) = 0.5x^2 - 1.5x + \mathbf{1}[x \neq 0]$ satisfies

- $f(0) = f(1) = f(2) = 0$;
- $f(3) = 1$.

The multiplicative error of the algorithm is unbounded!

I leave the analysis of the algorithm as an exercise.

Communication complexity

Communication complexity

Suppose we want to compute some function $f(x, y)$ where $x \in \{0, 1\}^a$ and $y \in \{0, 1\}^b$.

Communication complexity

Suppose we want to compute some function $f(x, y)$ where $x \in \{0, 1\}^a$ and $y \in \{0, 1\}^b$.

Alice has x and Bob has y, they collaborate to compute f.

Communication complexity

Suppose we want to compute some function $f(x, y)$ where $x \in \{0, 1\}^a$ and $y \in \{0, 1\}^b$.

Alice has x and Bob has y, they collaborate to compute f.

The compleixty is measured by bits communicated between the two.

Communication complexity

Suppose we want to compute some function $f(x, y)$ where $x \in \{0, 1\}^a$ and $y \in \{0, 1\}^b$.

Alice has x and Bob has y, they collaborate to compute f.

The compleixty is measured by bits communicated between the two.

We consider one-way communication model, with possible public random coins.

Example: Equality

Example: Equality

Consider the function $f(x,y) = \mathrm{EQ}(x,y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise,} \end{cases}$ where $x, y \in \{0,1\}^n$.

The one-way complexity of EQ is n.

Example: Equality

Consider the function $f(x,y) = \mathrm{EQ}(x,y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise,} \end{cases}$ where $x, y \in \{0,1\}^n$.

The one-way complexity of EQ is n.

This can be shown by a simple counting argument:

Example: Equality

Consider the function $f(x,y) = \mathrm{EQ}(x,y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise,} \end{cases}$ where $x, y \in \{0,1\}^n$.

The one-way complexity of EQ is n.

This can be shown by a simple counting argument:

If the number of bits sent by Alice is less than n, then she can send at most $2^1 + 2^2 + \cdots + 2^{n-1} = 2^n - 2$ distinct messages.

Example: Equality

Consider the function $f(x, y) = \mathrm{EQ}(x, y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise,} \end{cases}$ where $x, y \in \{0, 1\}^n$.

The one-way complexity of EQ is n.

This can be shown by a simple counting argument:

If the number of bits sent by Alice is less than n, then she can send at most $2^1 + 2^2 + \cdots + 2^{n-1} = 2^n - 2$ distinct messages.

By the pigeonhole principle, two different strings x and x' share the same message.

Example: Equality

Consider the function $f(x, y) = \mathrm{EQ}(x, y) = \begin{cases} 1 & \text{if } x = y; \\ 0 & \text{otherwise,} \end{cases}$ where $x, y \in \{0, 1\}^n$.

The one-way complexity of EQ is n.

This can be shown by a simple counting argument:

If the number of bits sent by Alice is less than n, then she can send at most $2^1 + 2^2 + \cdots + 2^{n-1} = 2^n - 2$ distinct messages.

By the pigeonhole principle, two different strings x and x' share the same message.

Bob can then use $y = x$ to fool the algorithm, a contradiction.

Randomness in communication

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

We treat x and y as two integers in $\{0, 2^n - 1\}$.

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

We treat x and y as two integers in $\{0, 2^n - 1\}$.

- Alice picks a random prime $p \in [n^2, 2n^2]$.
- She sends $(p, x \mod p)$ to Bob.
- Bob outputs 1 if $y \mod p = x \mod p$, and outputs 0 otherwise.

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

We treat x and y as two integers in $\{0, 2^n - 1\}$.

- Alice picks a random prime $p \in [n^2, 2n^2]$.
- She sends $(p, x \mod p)$ to Bob.
- Bob outputs 1 if $y \mod p = x \mod p$, and outputs 0 otherwise.

If $x = y$, the algorithm is always correct.

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

We treat x and y as two integers in $\{0, 2^n - 1\}$.

- Alice picks a random prime $p \in [n^2, 2n^2]$.
- She sends $(p, x \mod p)$ to Bob.
- Bob outputs 1 if $y \mod p = x \mod p$, and outputs 0 otherwise.

If $x = y$, the algorithm is always correct.

If $x \neq y$, the algorithm is wrong only if $x = y \mod p$.

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

We treat x and y as two integers in $\{0, 2^n - 1\}$.

- Alice picks a random prime $p \in [n^2, 2n^2]$.
- She sends $(p, x \mod p)$ to Bob.
- Bob outputs 1 if $y \mod p = x \mod p$, and outputs 0 otherwise.

If $x = y$, the algorithm is always correct.

If $x \neq y$, the algorithm is wrong only if $x = y \mod p$.

The number of primes between n^2 and $2n^2$ is $\Theta\left(\frac{n^2}{\log n}\right)$ (prime number theorem).

Randomness in communication

We can design a more efficient protocol for EQ by tossing coins.

We treat x and y as two integers in $\{0, 2^n - 1\}$.

- Alice picks a random prime $p \in [n^2, 2n^2]$.
- She sends $(p, x \mod p)$ to Bob.
- Bob outputs 1 if $y \mod p = x \mod p$, and outputs 0 otherwise.

If $x = y$, the algorithm is always correct.

If $x \neq y$, the algorithm is wrong only if $x = y \mod p$.

The number of primes between n^2 and $2n^2$ is $\Theta\left(\frac{n^2}{\log n}\right)$ (prime number theorem).

At most n primes q satisfy $x - y \mod q$ since $x, y < 2^n$.

Disjointness

Disjointness

The function $\text{DISJ}(x, y)$ tests whether two sets represented by x and y respectively share common elements.

Disjointness

The function DISJ(x, y) tests whether two sets represented by x and y respectively share common elements.

Formally, $\mathrm{DISJ}(x, y) = \begin{cases} 1 & \text{if } \langle x, y\rangle > 0 \\ 0 & \text{otherwise.} \end{cases}$, where $x, y \in \{0, 1\}^n$.

Disjointness

The function $\text{DISJ}(x, y)$ tests whether two sets represented by x and y respectively share common elements.

Formally, $\text{DISJ}(x, y) = \begin{cases} 1 & \text{if } \langle x, y \rangle > 0 \\ 0 & \text{otherwise.} \end{cases}$, where $x, y \in \{0, 1\}^n$.

Same argument as EQ shows that computing DISJ deterministically requires n bits of one-way communication.

Disjointness

The function $\mathrm{DISJ}(x, y)$ tests whether two sets represented by x and y respectively share common elements.

Formally, $\mathrm{DISJ}(x, y) = \begin{cases} 1 & \text{if } \langle x, y \rangle > 0 \\ 0 & \text{otherwise.} \end{cases}$, where $x, y \in \{0, 1\}^n$.

Same argument as EQ shows that computing DISJ deterministically requires n bits of one-way communication.

How about randomized protocols?

Disjointness

The function $\mathrm{DISJ}(x, y)$ tests whether two sets represented by x and y respectively share common elements.

Formally, $\mathrm{DISJ}(x, y) = \begin{cases} 1 & \text{if } \langle x, y \rangle > 0 \\ 0 & \text{otherwise.} \end{cases}$, where $x, y \in \{0, 1\}^n$.

Same argument as EQ shows that computing DISJ deterministically requires n bits of one-way communication.

How about randomized protocols?

Unlike EQ, the power of randomness does not help much here...

Lower bound for DISJ

Lower bound for DISJ

Theorem

Randomized protocol for DISJ with correct probability at least $2/3$ needs $\Omega(n)$ bits of one-way communication.

Lower bound for DISJ

Theorem

Randomized protocol for DISJ with correct probability at least $2/3$ needs $\Omega(n)$ bits of one-way communication.

We prove this for a special case of DISJ, the problem of INDEX.

Lower bound for DISJ

Theorem

Randomized protocol for DISJ with correct probability at least $2/3$ needs $\Omega(n)$ bits of one-way communication.

We prove this for a special case of DISJ, the problem of INDEX. So the lower bound is stronger.

Lower bound for DISJ

Theorem

Randomized protocol for DISJ with correct probability at least $2/3$ needs $\Omega(n)$ bits of one-way communication.

We prove this for a special case of DISJ, the problem of INDEX. So the lower bound is stronger.

INDEX: Alice holds a string $x \in \{0,1\}^n$, Bob holds an index $i \in [n]$. $\mathrm{INDEX}(x,i) = x_i$.

Yao's principle

Yao's principle

The main tool we will use to derive the lower bound is Yao's principle.

Yao's principle

The main tool we will use to derive the lower bound is Yao's principle.

Lemma

If there exists some distribution $\mathcal{D}$ over $\{0,1\}^a \times \{0,1\}^b$ such that any deterministic one-way communication protocol P with

$$\mathbf{Pr}_{(x,y)\sim\mathcal{D}}\left[P \text{ is wrong on } (x,y)\right] \le \varepsilon$$

costs at least k bits, then any randomized one-way protocol with error at most ε on any input also costs at least k bits one-way communication.

Lower bound for INDEX

Lower bound for INDEX

By Yao's principle, we only need to construct a distribution $\mathcal{D}$ over $\{0,1\}^n \times [n]$ so that for any protocol with costs $o(n)$, it outputs the correct answer with probability less than $7/8$.

Lower bound for INDEX

By Yao's principle, we only need to construct a distribution $\mathcal{D}$ over $\{0,1\}^n \times [n]$ so that for any protocol with costs $o(n)$, it outputs the correct answer with probability less than $7/8$.

We let $\mathcal{D}$ be the uniform distribution over $\{0,1\}^n \times [n]$.

Lower bound for INDEX

By Yao's principle, we only need to construct a distribution $\mathcal{D}$ over $\{0,1\}^n \times [n]$ so that for any protocol with costs $o(n)$, it outputs the correct answer with probability less than $7/8$.

We let $\mathcal{D}$ be the uniform distribution over $\{0,1\}^n \times [n]$.

Assume there exists a protocol P that uses at most $0.1n$ bits of one-way communication.

Namely, Alice holds a function $f : \{0,1\}^n \to \{0,1\}^{0.1n}$. On input x, she sends $f(x)$ to Bob.

Lower bound for INDEX

By Yao's principle, we only need to construct a distribution $\mathcal{D}$ over $\{0,1\}^n \times [n]$ so that for any protocol with costs $o(n)$, it outputs the correct answer with probability less than $7/8$.

We let $\mathcal{D}$ be the uniform distribution over $\{0,1\}^n \times [n]$.

Assume there exists a protocol P that uses at most $0.1n$ bits of one-way communication.

Namely, Alice holds a function $f : \{0,1\}^n \to \{0,1\}^{0.1n}$. On input x, she sends $f(x)$ to Bob.

Upon receiving $f(x)$, Bob outputs some number $y(f(x))_i$. We collect the outputs (for all possible $i \in [n]$ as a vector $y(f(x)) \in \{0,1\}^n$.

Lower bound for INDEX

By Yao's principle, we only need to construct a distribution $\mathcal{D}$ over $\{0,1\}^n \times [n]$ so that for any protocol with costs $o(n)$, it outputs the correct answer with probability less than $7/8$.

We let $\mathcal{D}$ be the uniform distribution over $\{0,1\}^n \times [n]$.

Assume there exists a protocol P that uses at most $0.1n$ bits of one-way communication.

Namely, Alice holds a function $f : \{0,1\}^n \to \{0,1\}^{0.1n}$. On input x, she sends $f(x)$ to Bob.

Upon receiving $f(x)$, Bob outputs some number $y(f(x))_i$. We collect the outputs (for all possible $i \in [n]$ as a vector $y(f(x)) \in \{0,1\}^n$.

The algorithm is correct if $x_i = y(f(x))_i$.

Therefore, we only need to upper bound

$$\mathbf{Pr}_{(x,i)\sim\mathcal{D}}\left[x_i = y(f(x))_i\right]$$

where both $f : \{0,1\}^n \to \{0,1\}^{0.1n}$ and $y : \{0,1\}^{0.1n} \to \{0,1\}^n$ are fixed!

Therefore, we only need to upper bound

$$\mathbf{Pr}_{(x,i)\sim\mathcal{D}}\left[x_i = y(f(x))_i\right]$$

where both $f:\{0,1\}^n \to \{0,1\}^{0.1n}$ and $y:\{0,1\}^{0.1n} \to \{0,1\}^n$ are fixed!

Since i is uniform in $[n]$, for any two strings $x, y \in \{0,1\}^m$,

$$\mathbf{Pr}_{i\in[n]}\left[x_i \neq y_i\right] = \frac{d_H(x,y)}{n}.$$

Let $S = y(x(\{0,1\}^n)) \subseteq \{0,1\}^n$ be a set of size at most $\{0,1\}^{0.1n}$. Since x is uniform in $\{0,1\}^n$, we only need to show: there are many $x \in \{0,1\}^n$ satisfying $d_H(x,S) \geq n/4$.

Let $S = y(x(\{0,1\}^n)) \subseteq \{0,1\}^n$ be a set of size at most $\{0,1\}^{0.1n}$. Since x is uniform in $\{0,1\}^n$, we only need to show: there are many $x \in \{0,1\}^n$ satisfying $d_H(x, S) \geq n/4$.

This is true since $\mathrm{Ball}\left(S, \frac{n}{4}\right) \leq 2^{0.1n} \cdot \sum_{j=0}^{\frac{n}{4}} \binom{n}{j} \leq n2^{0.95n}$.

Lower bound for F_∞

Lower bound for F_∞

Our motivation for introducing communication model is to prove lower bound for streaming problems.

Lower bound for F_∞

Our motivation for introducing communication model is to prove lower bound for streaming problems.

For example, we can use the lower bound for DISJ to derive lower bound for estimating F_∞.

Lower bound for F_∞

Our motivation for introducing communication model is to prove lower bound for streaming problems.

For example, we can use the lower bound for DISJ to derive lower bound for estimating F_∞.

Theorem

Any randomized algorithm to estimate F_∞ within err $\varepsilon = 0.2$ requires $\Omega(n)$ bits of memory.

Lower bound for F_∞

Our motivation for introducing communication model is to prove lower bound for streaming problems.

For example, we can use the lower bound for DISJ to derive lower bound for estimating F_∞.

Theorem

Any randomized algorithm to estimate F_∞ within err $\varepsilon = 0.2$ requires $\Omega(n)$ bits of memory.

Proof.

Treat $\mathbf{x}$ and $\mathbf{y}$ as streams $\{i \in [n] \mid x_i = 1\}$ and $\{i \in [n] \mid y_i = 1\}$ respectively. □

A general paradiam

A general paradiam

Previous proof provides a general paradiam for proving streaming lower bound based on communication lower bound:

A general paradiam

Previous proof provides a general paradiam for proving streaming lower bound based on communication lower bound:

A streaming algorithm to compute $f(x, y)$ using s bits of memory implies a protocol to compute $f(x, y)$ using at most s bits of one-way communication.

A general paradiam

Previous proof provides a general paradiam for proving streaming lower bound based on communication lower bound:

A streaming algorithm to compute $f(x, y)$ using s bits of memory implies a protocol to compute $f(x, y)$ using at most s bits of one-way communication.

We will see more applications next time!