Lower bound for streaming algorithms

Barna Saha

1 Communication Complexity

Suppose there are two parties, Alice and Bob. Alice has an input $x \in \{0,1\}^a$, Bob an input $y \in \{0,1\}^b$. Neither one has any idea what the other's input is. Alice and Bob want to cooperate to compute a Boolean function: $f : \{0,1\}^a \times \{0,1\}^b \to \{0,1\}$, defined on their joint input.

We here only consider one-way communication complexity. Alice sends Bob a message z which is only a function of her input x. Bob computes the function $f(x,y)$ based on z and y, and declare it.

The *one-way communication complexity* of a Boolean function f is the minimum worst-case number of bits used by any 1-way protocol that correctly decides the function (or say decide it correctly with probability $> \frac{1}{2}$).

Note that, one-way communication complexity of a function f is always at most a, since Alice can send her entire input to Bob who can then compute the function exactly.

2 Connection to Streaming Algorithm

Small-space streaming algorithms imply low one-way communication complexity. Consider a problem that can be solved using a streaming algorithm S that uses space s only. The idea is for Alice and Bob to treat their input (x,y) as a stream, with all of x arriving before any of y arrives. Alice can then feed x to S, which returns a summary of size s bits. Alice sends these s bits of information to Bob. Bob can then simply restart the streaming algorithm S seeded with s, and feed y to it. The algorithm them computes some function of (x,y) with one way communication of s bits.

The communication cost of the induced protocol is exactly the same as the space used by the streaming algorithm.

To prove lower bound on space usage of a streaming algorithm, we need to come up with a Boolean function that (i) can be reduced to a streaming problem that we want to study, and (ii) does not admit a low one-way communication complexity.

3 The Disjointness Problem

Alice and Bob both hold n-bit vectors x and y. We interpret these as characteristic vectors of two subsets of the universe $\{1,2,....,n\}$ with the subsets corresponding to the 1 coordinates. We then define the Boolean function $DISJ$ as $DISJ(x,y) = 0$ if there is an index $i \in \{1,2,...,n\}$ with $x_i = y_i = 1$, and $DISJ(x,y) = 1$ otherwise.

Here are some well-knows results.

Theorem 1. *Every deterministic one-way communication protocol that computes the function $DISJ$ uses at least n bits of communication in the worst case.*

Easy to prove using Pigeonhole Principle. Consider any 1-way communication protocol where Alice always sends at most $n-1$ bits. This means, ranging over 2^n possible inputs x that Alice might have, the possible number of different messages is at most 2^{n-1}. Therefore, by the

Pigeonhole Principle, there exist at least two inputs x_1 and x_2 for which Alice sends the same message z. Suppose x_1 and x_2 differ in the ith coordinate, and y is simple the ith basis vector (all 0s except the ith position). Since Bob only receives z and has y, for both x_1 and x_2, Bob computes the same answer–thus the protocol is incorrect. □

However, a stronger result for randomized protocols can also be proven.

Theorem 2. *Every randomized protocol that, for every input (x, y), correctly decides the function $DISJ$ with probability at least $\frac{2}{3}$, uses $\Omega(n)$ communication in the worst case.*

The probability in the above theorem is over the coin flips performed by the protocol–there is no randomness in the input, which is "worst-case". There is nothing special about the constant $\frac{2}{3}$ which can be replaced by any constant $> \frac{1}{2}$.

4 Space Lower Bound for F_∞

We have see space-efficient streaming algorithms for computing F_0 (distinct items), F_2 (F_1 is easy), and heavy-hitters. Here we show that if we have to compute the maximum frequency within some constant factor of approximation in a single pass, then the space requirement is $\Omega(n)$, even using a randomized algorithm.

Theorem 3. *Every randomized streaming algorithm that, for every data stream of length m, computes F_∞ to within $(1 \pm .2)$ factor with probability at least $2/3$ uses space $\Omega(\min\{m, n\})$.*

We will reduce $DISJ$ to streaming computation of F_∞.

Suppose S is a streaming algorithm using space s that computes F_∞. Given an input (x, y) of $DISJ$, Alice sends to S, $(i, 1)$ for every $x_i = 1$. S then computes a summary of size s and sends it to Bob. Bob, seeds S with s, and feeds $(i, 1)$ for every $y_i = 1$. At the end if S returns an answer ≥ 1.6, then Bob declares $DISJ = 0$, else it declares $DISJ = 1$.

Note that $F_\infty = 2$ iff the two sets x and y are not disjoint, that is $DISJ = 0$. In that case S returns an estimate of F_∞ at least $0.8 * 2 = 1.6$.

When $DISJ = 1$, $F_\infty = 1$ (or 0 if x and y are empty), in that case S can return an estimate at most 1.2.

Therefore, from the hardness of communication complexity of $DISJ$, we conclude that computing F_∞ in the streaming setting in a single pass requires $\Omega(\min\{m, n\})$ space, even using a randomized protocol.