CSE 522: Sublinear (and Streaming) Algorithms **Spring 2014**

Lecture 11: J-L Lower Bound, Communication Complexity

May 5, 2014

Lecturer: Paul Beame *Scribe: Paul Beame*

We finish the proof of the lower bound on the dimension required for the Johnson-Lindenstrauss Lemma to hold:

Theorem 0.1 (Alon). *There is a set of $n+1$ points in $\mathbb{R}^n$ such that for $1/2 \geq \varepsilon > 1/\sqrt{n}$, any mapping $f : \mathbb{R}^n \to \mathbb{R}^k$ that preserves the squares of ℓ_2^2 distance within $(1 \pm \varepsilon)$ factor requires that $k = \Omega(\frac{\log n}{\varepsilon^2 \log(1/\varepsilon)})$.*

we note that this theorem implies that the distributional Johnson-Lindenstrauss Lemma has a dimension lower bound of $\Omega(\frac{\log(1/\delta)}{\varepsilon^2 \log(1/\varepsilon)})$.

Last time we showed that by considering the images under f of the $n+1$ points $0 = (0, \ldots, 0)$ and $e_1 = (1, 0, \ldots, 0)$ through $e_n = (0, \ldots, 0, 1)$, we could find an $n \times n$ symmetric real matrix B of rank at most $2k$ that is nearly the identity matrix in that its diagonal entries are all 1 and its off-diagonal entries are all at most 4ε in absolute value. We also proved the following Lemma.

Lemma 0.2. *Let $B = (b_{ij})_{ij}$, be an $n \times n$ symmetric real matrix with $b_{ii} = 1$ for all i and $|b_{ij}| \leq \varepsilon$ for all $i \neq j$. Then the rank r of B is at least*

$$\frac{n}{1 + (n-1)\varepsilon^2}.$$

This lemma does not yield a strong enough lower bound when applied to the matrix B derived from the images under f because the lower bound is only $\Omega(1/\varepsilon^2)$ and does not depend on n.

However, if the off-diagonal entries were bounded by $1/\sqrt{n}$ in absolute value then the lower bound on the rank r would be at least $n/2$. The obvious way we can reduce the magnitude of the off-diagonal entries while preserving the diagonal entries is raise each entry to some fixed power ℓ. This may increase the rank of B but Alon showed that this operation does not increase the rank by too much.

Definition 0.3. *For a matrix $B = (b_{ij})_{ij}$, write $B^{(\ell)}$ to denote the matrix whose ij-th entry is b_{ij}^{ℓ}. (This is powering with respect to the* Hadamard matrix product *where the product of matrices $A = (a_{ij})_{ij}$ and $B = (b_{i}j)_{ij}$ is the matrix $C = A \circ B$ whose ij-th entry is $a_{ij}b_{ij}$.)*

Lemma 0.4. *If $rank(B) \leq k$ then $rank(B^{(\ell)}) \leq \binom{k+\ell-1}{\ell} \leq \binom{k+\ell}{\ell}$.*

Proof. Let $u_1, \ldots, u_k$ be a basis for the row space of B. Therefore we can write the i-th row of B as

$$a_{i1}u_1 + \cdots a_{ik}u_k$$

for some real values $a_{i1}, \ldots, a_{ik}$ and in particular

$$b_{ij} = a_{i1}u_{1j} + \cdots a_{ik}u_{kj}.$$

Then

$$\begin{aligned} b_{ij}^{\ell} &= (a_{i1}u_{1j} + \cdots a_{ik}u_{kj})^{\ell} \\ &= \sum_{\ell_1+\cdots+\ell_k=\ell} c_{i\ell_1\ldots\ell_k} u_{1j}^{\ell_1} u_{2j}^{\ell_2} \cdots u_{kj}^{\ell_k} \end{aligned}$$

Where $c_{i\ell_1\ldots\ell_k}$ is constant depending on the values of $a_{i1}, \ldots, a_{ik}, \ell_1, \ldots, \ell_k$. (In particular it does not depend on j.) This means that the i-th row of $B^{(\ell)}$ is a linear combination of the vectors

$$(u_{11}^{\ell_1}u_{21}^{\ell_2} \cdots u_{k1}^{\ell_k}, u_{12}^{\ell_1}u_{22}^{\ell_2} \cdots u_{k2}^{\ell_k}, \ldots u_{1n}^{\ell_1}u_{2n}^{\ell_2} \cdots u_{kn}^{\ell_k}),$$

each with coefficient $c_{i\ell_1\ldots\ell_k}$. This means that $rank(B^{(\ell)})$ is at most the number of choices of $\ell_1, \ldots, \ell_k$ that sum to ℓ. We can identify each such choice with a binary vector of length $k + \ell - 1$ having ℓ 1s and $k - 1$ 0s, where the 0s denote the separators between the k runs of 1s (possibly empty) whose length is the associated ℓ_i. There are precisely $\binom{k+\ell-1}{\ell}$ such choices. $\square$

Proof of Theorem 0.1. In order to reduce the off-diagonal entries of our matrix B to at most $1/\sqrt{n}$, it suffices to have $(4\varepsilon)^{\ell} \leq 1/\sqrt{n}$, that is $\sqrt{n} \leq \left(\frac{1}{4\varepsilon}\right)^{\ell}$, or $\frac{\log n}{2} \leq \ell \log_2(\frac{1}{4\varepsilon})$. Therefore we choose $\ell = \lceil \frac{\log n}{2\log_2(1/(4\varepsilon))} \rceil$.

Since the matrix B has rank at most $2k$, by the above two lemmas, we have $\binom{2k+\ell}{2k} \geq rank(B^{(\ell)}) \geq n/2$. To analyze this we use one of the standard upper bounds on binomial coefficients: It is immediate that $\binom{n}{m} \leq \frac{n^m}{m!}$ and one can easily show by induction and the definition of e that $m! \geq (m/e)^m$ and hence that $\binom{n}{m} \leq (ne/m)^m$.

Apply this we have that $\binom{2k+\ell}{\ell} \leq [(1 + 2k/\ell)e]^{\ell}$. Therefore we have

$$[(1 + 2k/\ell)e]^{\ell} \geq n/2.$$

Taking log base 2 and dividing by ℓ we have

$$\begin{aligned} \log_2((1 + 2k/\ell)e) &\geq \frac{\log_2 n - 1}{\ell} \\ &= 2\log_2(1/(4\varepsilon)) - o(1) \end{aligned}$$

by the definition of ℓ. This implies that $\log_2(1+2k/\ell) \geq 2\log_2(1/\varepsilon) - C$ for some constant $C < 2$. Since the right-hand quantity grows large, it must be the case that $2k/\ell$ is larger than 1 and hence we obtain that $\log_2(2k/\ell) \geq 2\log(1/\varepsilon) - C'$ for some constant C'. Exponentiating implies that $2k/\ell$ is $\Omega(1/\varepsilon^2)$. Therefore k is $\Omega(\ell/\varepsilon^2)$ which matches the theorem statement when we plug in the value of ℓ. $\square$

Continuing a focus on lower bounds we introduce communication complexity which provides the main vehicle for proving lower bounds for streaming algorithms (among its many applications).

1 Communication Complexity

Communication complexity studies the communication required by some number of computationally unbounded players, each of whom receives part of the input, and who cooperate to compute some function of the entire input.

In the cases we consider, all communication is done by writing on a shared blackboard. (In the case of two players this is equivalent to sending messages to each other.) The complexity is measured in terms of the number of bits sent.

In the 2-party case we have two players, typically denoted Alice and Bob. Alice receives as input $x \in X = \{0,1\}^n$, and Bob receives $y \in Y = \{0,1\}^n$ and they must exchange messages to compute some function $f(x,y)$. We require that $f(x,y)$ is the last message sent/written on the blackboard. Only one play can write on the blackboard at a given time.

A communication protocol is given by a binary tree in which each node corresponds to a different prefix of what is written on the blackboard. Each internal node is labeled by the player whose turn it is to write after that prior communication has been sent. Whether the protocol next sends a 0 or a 1 depends only on the input of the player who speaks. We can think of this dependence as a function mapping the set of inputs of the player who speaks to $\{0,1\}$.

In addition to general 2-party protocols we also will consider 1-way protocols in which Alice sends one message to Bob and Bob must then immediately produce $f(x,y)$.

In deterministic protocols, there is a fixed function at each node and this precisely determines what bit to communicate next. Randomized protocols will be important for us; there are two natural versions depending on how the random bits are shared. In the case of *private* protocols, each player has its own source of random bits and so each function in the protocol tree is simply a randomized function of the input that is determined in part by the private random bits. In *public* protocols, there is a single source of randomness that can be ready by all players. Since each fixed choice of the shared random bits yields a separate deterministic function, we can think of the randomized protocols with public randomness as distributions over deterministic protocols.

For a given protocol P, we can define C_P to be the maximum number over all inputs to f of the number of bits sent by players using P. Define

- $D(f) = \min_{P \text{ computes } f, P \text{ deterministic}} C_P$, the minimum worst-case number of bits communicated among all deterministic 2-party protocols that output $f(x,y)$ for all inputs $(x,y) \in X \times Y$. Write $D^{A \to B}(f)$ for the analogous one way version.

- $R_\delta(f)$ is the minimum number of bits sent in any 2-party randomized protocol with public randomness that for every $x \in X, y \in Y$ computes $f(x, y)$ correctly with probability at least $1 - \delta$. Write $R_\delta^{A\to B}(f)$ to be the analogous definition for 1-way randomized protocols.
- Let $R_\delta^{priv}(f)$ and $R_\delta^{priv,A\to B}(f)$ be the corresponding definitions for 2-party protocols with private randomness.

We can also consider protocols for t players in which each player receives its own private input $x_i \in X_i$ and the function to be computed is $f(x_1, \ldots, x_t)$. At each step in these protocols, the player next to speak is determined by the communication prior to that step.

We will apply communication complexity by considering the input steam as being partitioned among the players so that the function to be computed or approximated by the streaming algorithm takes as input the stream $x_1 x_2$ or $x_1 \ldots x_t$. If the streaming algorithm uses space S, the players can simulate the streaming algorithm in which players $1, \ldots, t$ speak in order, player i simulates the algorithm during the reading of x_i, and player i writes the content of the memory on the blackboard at the end of his segment. In particular, the total communication will be at most $(t-1)S$. Hence a lower bound on communication protocols can yield a lower bound on the space used by the protocol.

We add a subscript t to the above complexity definitions to represent t-party protocols; i.e., $D_t(f)$ and $R_{t,\delta}(f)$.

Randomized protocols will be need to simulate randomized streaming algorithms. Although the model with private randomness at first seems somewhat more natural than the model with public randomness, the following theorem of Newman shows that the two are not significantly different.

Theorem 1.1. *Let $f : X \times Y \to \{0, 1\}$ where $X, Y \subseteq \{0, 1\}^n$. Then*

$$R_{2\delta}^{priv}(f) \le R_\delta(f) + O(\log n + \log(1/\delta)).$$

Proof Sketch. We show how to simulate public randomness with private randomness at the expense of a small amount of additional error: The original algorithm with public randomness, running the algorithm on a random string is like an experiment with a biased coin that has a probability of at most δ of landing tails. By a Chernoff bound one can show that after making a polynomial (in n) number of independent samples of random strings, the probability of observing larger than error 2δ is exponentially small, say less than 2^{-2n}. This shows that there exists a polynomial-size set R of random strings such that the error in choosing a random one of these string is at most 2δ for every $(x, y) \in X \times Y$. Alice now simply uses her random bits to choose a random sample element in this R and then sends this as part of her first message. $\square$

Definition 1.2. *A* combinatorial rectangle *in $X \times Y$ is a set of the form $A \times B$ for some subsets $A \subseteq X$ and $B \subseteq Y$.*

Lemma 1.3. *For any deterministic 2-party protocol, the set of the inputs $(x, y) \in X \times Y$ that reach each node in the protocol tree is a combinatorial rectangle.*

Proof. This is easily proved by induction starting at the root of the protocol tree. This set is $X \times Y$ at the root. At each node, if we already know that the associated set for a given node is combinatorial rectangle, then the same property holds for its children since the condition for deciding which child an input leads to depends only on the input of the player whose turn it is to speak. $\square$

In particular, each leaf of the protocol tree, which has the property that the protocol produces the same output for all of the inputs in the rectangle associated with that leaf. This lets us prove the following easy property.

Definition 1.4. *Define the Equality predicate $EQ : \{0,1\}^{2n} \to \{0,1\}$ by $EQ(x, y) = 1$ if and only if $x = y$.*

It is immediate that $D(EQ) \leq n + 1$ via the protocol in which Alice sends x and Bob responds with the answer $f(x, y)$.

Proposition 1.5. $D(EQ) = n + 1$

Proof. Consider the $2^n \times 2^n$ matrix M_{EQ} that has rows indexed by elements of X and columns indexed by Y, and whose (x, y) entry has value $EQ(x, y)$. It is immediate that M_{EQ} is the $2^n \times 2^n$ identity matrix.

Observe that no two distinct points on the diagonal (x, x) and (x', x') on which EQ evaluates to 1 can be in a combinatorial rectangle on which the output is constant since, otherwise, (x, x') would be in the rectangle and $EQ(x, x') = 0$. There are 2^n diagonal elements and each much reach a separate leaf of the protocol. Moreover, there must be at least 1 leaf reached by off-diagonal elements. Therefore there are at least $2^n + 1$ leaves and so the height of the protocol tree must be at least $n + 1$. $\square$

We immediately obtain the following corollary for streaming algorithms, which shows that we need to move beyond exact deterministic algorithms to find small space algorithms:

Corollary 1.6. *Any deterministic streaming algorithm that exactly computes F_0 requires space $\Omega(n)$.*

Proof. On inputs x and y, respectively, Alice and Bob can simulate the streaming algorithm for F_0 on input stream $\overline{xy}$ where $\overline{x}$ is an encoding of x that represents x_i by the number $2i - x_i$. (Alternatively one can consider this as a mapping of the bit x_i as two bits where 0 is mapped to

01 and 1 is mapped to 10 and viewing the resulting input as the characteristic vector of a subset of $[2n]$.) Alice will send Bob the contents of the storage after $\overline{x}$ has been read.

Observe that if $x_i = y_i$ then their corresponding elements together contribute just one element to $F_0(\overline{xy})$, but if $x_i \neq y_i$ then their corresponding elements yield a count of two for $F_0(\overline{xy})$. Therefore

$$F_0(\overline{xy}) = n + \Delta(x, y)$$

where $\Delta(x, y)$, the *Hamming distance* between x and y, is the number of coordinates on which x and y differ. Observe that $x = y$ iff $\Delta(x, y) = 0$. Now, if the protocol Bob can compute F_0 exactly, then Bob can compute $\Delta(x, y) = F_0(\overline{xy}) - n$ and hence determine $EQ(x, y)$. The storage must be at least $n + 1$ which is $\Omega(n)$. $\square$