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

Lecture 16: Property Testing of Functions, Junta Testing

May 28, 2014

Lecturer: Paul Beame *Scribe: Paul Beame*

1 Property Testing of Functions

For problems of testing functions, we consider the input to be a function $f : X \to Z$ on some finite domain X given by a table of values from Z. Some examples:

- An input $f : [n] \to [M]$ is equivalent to our usual input $x_1, \ldots, x_n$ where $x_i = f(i)$.
- A Boolean function $f : \{0,1\}^n \to \{0,1\}$. Note that the size of the input table is huge as a function of n. We can think of f as being given by a black box that allows us to evaluate f or in a form that we have to go out and collect.
- More generally, a function $f : \Omega^n \to Z$, where Ω is some finite set.
- $f : G \to \mathbb{R}$, where G is some group.

For the purposes of property testing we need a distance measure between functions and associated notions of ε-close and hence ε-far. The distance measure we use is the fractional Hamming distance between the truth tables, which we generalize according to some probability distribution μ on X: We say that f and g are ε-close under μ iff

$$\mathrm{P}_{x \sim \mu}[f(x) \neq g(x)] \leq \varepsilon$$

and they are ε-far otherwise.

Junta Testing Junta testing applies to input functions of the form $f : \Omega^n \to Z$. Before defining juntas in general, we motivate the terminology by beginning with the simplest case.

Definition 1.1. *A function f is a* dictatorship *iff there is some $i \in [n]$ and function $g : \Omega \to Z$ such that for all $x \in \Omega^n$, $f(x) = g(x_i)$. That is, i is a dictator whose input determines the value of f.*

For Boolean functions, the only possibile dictatorships are $f(x) = x_i, \overline{x_i}, 0, 1$.

Definition 1.2. *A function f is a k-*junta *iff there is a subset $J \subseteq [n]$ with $|J| \leq k$ and function $g : \Omega^J \to Z$ such that for all $x \in \Omega^n$, $f(x) = g(x_J)$ where $x_J = (x_{j_1}, \ldots, x_{j_k})$ for $J = \{j_1, \ldots, j_k\}$.*

Obviously a dictatorship is a 1-junta. The property testing problem we consider is to determine whether

- f is a k-junta, or
- f is ε-far from any k-junta

and to answer correctly with probability $\geq 2/3$.

Dictatorship testing The first major application of junta testing was the use of dictatorship testing in strong versions of the PCP Theorem which show how to take an arbitrary CNF formula and convert it to a form so that any satisying assignment can be checked with good probability using only a constant number (3) of queries to the assignment. For satisfying assignments for the new formula, parts of the satisfying assignment of the original are encoded as a dictatorship as follows: Encode some $i \in [n]$ by given the truth table of the function $f : \{0,1\}^n \to \{0,1\}$ given by $f(x) = x_i$. This takes the $\log n$ bits to specify i to 2^n bits of the table for f and is naturally called the "long code". In the PCP theorem proof, n is very small, but the tests require only 3 queries to f. It necessarily has 2-sided error. The motivation and proofs for the tests are too long to consider here.

k-junta testing We now consider the general case of testing k-juntas. This is very natural, especially in machine learning when one has a very large number of features of the input data and one wants to know if there is a small number of these features that determine the value of some other property of the data. In this case, the features are the coordinates.

Definition 1.3. *We say that f* depends on *$i \in [n]$ or i is* relevant for *f iff there exist $x, x' \in \Omega^n$ such that $f(x) \neq f(x')$ but for all $j \neq i$, $x_j = x'_j$.*

The general idea for determining whether f is a k-junta will be to isolate coordinates that are relevant for f, one after another. But how do we find a single such coordinate?

Suppose that we have $x \neq y$ such that $f(x) \neq f(y)$. It is not hard to see that at least one of the coordinates i such that $x_i \neq y_i$ must be relevant for f. Let D be the set of such coordinates. D might be a very large fraction of all coordinates so this observation does not immediately seem to help. However, we can consider the hybrid strings between x and y that look like x on part of D and y on the rest. One way to to consider such hybrid string is in terms of the Hamming graph G on Ω^n where we include an edge (u, v) iff the Hamming distance $\Delta(u, v) = 1$. Then any shortest path between x and y will be of length $\Delta(x, y)$ and will consist of hybrid strings. The endpoints of the path will have different values of f. The key observation is that we can form binary search on the path to find a pair of adjacent vertices of the path z, z' with different f values. The coordinate on which z and z' differ is relevant for f. The cost is $\log_2 \Delta(x, y)$ queries to f.

In general $\log_2 \Delta(x, y)$ can be as large as $\log_2 n$ and we would like the algorithm to have a number of queries that is independent of n. To achieve this, we won't isolate coordinates precisely. Suppose that we have broken up the input coordinates into blocks $I_1, \ldots, I_s$ then using the same hybrid technique but selecting the hybrid elements by only changing entire blocks at a time between x and y we can find a block that contains a relevant coordinate using binary search in only $\log_2 s$ queries. Of course, this will only really be useful for testing k-juntas if each block contains at most one relevant coordinate of f.

Of course, isolating just one relevant coordinate won't be enough. To make sure that we don't repeat finding the same relevant coordinates, the algorithm maintains a set S on which the strings x and y are allowed to differ. This leads to the following algorithm, due to Eric Blais.

```
**JuntaTest**$(f, k, \varepsilon)$
 1: $s \leftarrow s(k, \varepsilon)$
 2: $t \leftarrow 12(k+1)/\varepsilon$
 3: Randomly partition $[n]$ in $s$ sets $I_1, \ldots, I_s$; i.e., choose a random $h : [n] \to [s]$ and let $I_j = \{i \mid h(i) = j\}$.
 4: $S \leftarrow [n]$
 5: $\ell \leftarrow 0$
 6: for $r \leftarrow 1$ to $t$ do
 7:     Choose $x, y$ independently from $\Omega^n$
 8:     $y_{\overline{S}} \leftarrow x_{\overline{S}}$
 9:     if $f(x) \neq f(y)$ then
10:         Use binary search on $[s]$ to find a set $I_j$ such that $I_j$ contains a relevant variable
11:         Remove $I_j$ from $S$
12:         $\ell \leftarrow \ell + 1$
13:         if $\ell > k$ then
14:             Output "NOT A $k$-JUNTA" and halt
15:         end if
16:     end if
17: end for
18: Output "$k$-JUNTA"
```

We have not yet specified a value for $s(k, \varepsilon)$. One of the things we wanted in our plan is that if, for example, f is a K-junta for $K = k + 1$, say, then no two of its relevant coordinates would land in the same block of the partition. For this we do not need $s(k, \varepsilon)$ to be particularly large. In particular, our observation much earlier in the course that even if h is only pairwise independent, the probability that a set of size K has a collision under h is at most $\binom{K}{2}/s$ and hence $s = \omega(K^2)$ suffices.

However, we require somewhat more than this for success, namely that two random strings that do not differ outside of S will have a good probability of having different values under f.

Theorem 1.4 ((Blais)). *The JuntaTest algorithm succeeds with probability $\geq 2/3$ when $s(k, \varepsilon)$ is $(k/\varepsilon)^{O(1)}$.*

It is not hard to see that the total number of queries of JuntaTest(f, k, ε) is $O(t + (k+1)\log_2 s)$. For s of the form $(k/\varepsilon)^{O(1)}$ this is

$$O(k/\varepsilon + k\log(k/\varepsilon)) = O(k/\varepsilon + k\log k + k\log(1/\varepsilon)) = O(k/\varepsilon + k\log k).$$

The algorithm chooses two random strings that agree outside of S and compares the value of f on those strings. The probability that f has a different value on these strings is known as the *influence* of S on f,

$$Inf_f(S) = \mathbb{P}_{x,y:x_{\bar{S}}=y_{\bar{S}}}[f(x) \neq f(y)].$$

Influence is both monotonic and sub-additive; i.e., for every S, T,

$$Inf_f(S) \leq Inf_f(S \cup T) \leq Inf_f(S) + Inf_f(T).$$

Using a generalization of Fourier analysis for Boolean functions one can show the following:

Proposition 1.5. *If $f : \Omega^n \to Z$ is ε-far from every k-junta then for all $J \subseteq [n]$ with $|J| \leq k$, $Inf_f([n] - J) > \varepsilon$.*

Blais used an extension of these ideas to prove that the influence of even smaller sets will be large:

Lemma 1.6. *Let $I_1, \dots, I_s$ for $s = 10^20k^9/\varepsilon^5$ be a random partition of $[n]$. Then with probability $\geq 5/6$ any function $f : \Omega^n \to Z$ that is ε-far from every k-junta satisfies $Inf_f([n] - \mathcal{J}) > \varepsilon/2$ for every $\mathcal{J} = I_{j_1} \cup \cdots \cup I_{j_k}$.*

That is, for every k-junta J, even if we remove all the elements in the blocks containing J then we still get large influence. The large constants in the definition of s are likely artifacts of the proof and it would be nice to reduce them.

The proof of the correctness of JuntaTest now follows immediately:

It is immediate that the algorithm always is correct when f is a k-junta.

Suppose that f is ε-far from every k-junta. Observe that whenever the algorithm chooses x and y, the set S in the algorithm satisfies $S \supseteq [n] - \mathcal{J}$ for a set $\mathcal{J}$ of the form given in the lemma. Therefore $Inf_f(S) \geq Inf_f([n] - \mathcal{J}) > \varepsilon/2$.

Therefore in line 9, $f(x) \neq f(y)$ with probability $> \varepsilon/2$ and hence ℓ is incremented with this probability. Therefore the expected number of iterations before ℓ is incremented is $< 2/\varepsilon$. It follows that the expected number of iterations before ℓ is incremented $k+1$ times is $< 2(k+1)/\varepsilon$. By Markov's inequality the probability that this number of iterations is more than 6 times its expected value (i.e., more than $t = 12(k+1)/\varepsilon$) is at most $1/6$.

The total failure probability is then at most this 1/6 plus the 1/6 failure probability of Lemma 1.6 or 1/3.

2 Lower bounds for Property Testing

We now show that the junta testing algorothm is nearly optimal using the following special case of a general lower bound method of Blais, Brody, and Matulef.

Given a property P of functions with Boolean output, we define a 2-player communication complexity problem $P_\oplus$ as follows:

- Alice gets as input a function f
- Bob gets as input a function g
- The output $P_\oplus(f, g)$ is 1 iff the function h given by $h(x) = f(x) \oplus g(x)$ has property P

The following is almost immediate:

Proposition 2.1. *If $q(P)$ is the number of queries to test property P with success $\geq 2/3$. then*

$$R_{1/3}(P_\oplus) \leq 2q(P).$$

Further, if $q^{na}(P)$ is the number of non-adaptive queries to test P, then $R_{1/3}^{A\to B}(P_\oplus) \leq q^{na}(P)$.

Proof. Alice and Bob simply simulate the property testing algorithm for P using their shared random bits. Whenever the algorithm for P requires an evaluation of h on x, Alice and Bob exchange bits $f(x)$ and $g(x)$ to compute $h(x)$. Once Alice sends Bob $f(x)$, Bob has enough information to continue but Alice needs $g(x)$ to continue the next step of the property testing algorithm. If the property testing algorithm is non-adaptive, she does not need this information. □

We previously discussed the unique disjointness problem $UDISJ_n = UDISJ_n^2$ where

$$UDISJ_n(A, B) = \begin{cases} 0 & A \cap B = \emptyset \\ 1 & |A \cap B| = 1 \\ \text{don't care} & \text{otherwise.} \end{cases}$$

and noted that $R_{1/3}(UDISJ_n) = \Omega(n)$.

In fact this lower bound holds even when we are promised that $|A| = |B| = n/4$. More precisely, if we define the problem where

$$UDISJ_{k,n}(A,B) = \begin{cases} 0 & A \cap B = \emptyset,\ |A| = |B| = k \\ 1 & |A \cap B| = 1,\ |A| = |B| = k \\ \text{don't care} & \text{otherwise,} \end{cases}$$

then the above says that $R_{1/3}(UDISJ_{n/4,n}) = \Omega(n)$.

It follows immediately that $R_{1/3}(UDISJ_{k,n}) = \Omega(k)$ for all $k \leq n/4$ by observing that the function $UDISJ_{k,4k}$ is a special case $UDISJ_{k,n}$ in which the sets A and B are contained in only the first $4k$ elements.

Theorem 2.2. *For any $\varepsilon < 1/2$, property testing for $2k$-juntas requires $\Omega(k)$ queries.*

Proof. We give a communication complexity lower bound for the property $P_\oplus$ where P is the property of being a $2k$-junta. In particular, we prove that such an algorithm solves $UDISJ_{k+1,n}$ and hence $R_{1/3}(P_\oplus) \geq R_{1/3}(UDISJ_{k+1,n}) = \Omega(k)$.

Suppose that Alice gets input a set A of size $k+1$ and Bob gets a set B of size $k+1$. Alice use as her input to the protocol for $P_\oplus$ the function $f(x) = \bigoplus_{i \in A} x_i$ and Bob uses $g(x) = \bigoplus_{i \in B} x_i$. Observe that

$$h(x) = \bigoplus_{i \in A} x_i \oplus \bigoplus_{i \in B} x_i = \bigoplus_{i \in A \oplus B} x_i.$$

If $|A \cap B| = 1$ then $|A \oplus B| = 2k$ and hence h is a $2k$-junta.

If $A \cap B = 0$ then $|A \oplus B| = 2k+2$. In this case, we claim that h is at distance precisely 1/2 from every $2k$-junta: To see this, consider any $2k$-junta h' and any assignment of values to the $2k$ coordinates on which h' depends. This fixes the value of h'. However, this does not fix the values of all the coordinates in $A \oplus B$ and hence the values of h consistent with this fixing are equally balanced between 0 and 1, so it agrees with h' on precisely 1/2 of the inputs. It follows that for any $\varepsilon < 1/2$, h is ε-far from h'.

This $P_\oplus$ can compute the output of $UDISJ_{k+1,n}$ as required. □

Open Question Is the complexity of testing k-juntas $O(k)$ or $\Omega(k \log k)$?

In the non-adaptive case we get the stronger lower bound based on the following 1-way communication complexity lower bound.

Lemma 2.3. $R^{A \to B}(UDISJ_{k,n}) = \Omega(k \log k)$

Proof. Assume without loss of generality that k is a power of 2, $k = 2^\ell$. We give a reducton from the $INDEX_{k(\ell+1)}$ problem and recall that $R^{A\to B}_{1/3}(INDEX_m)$ is $\Omega(m)$. Choose $n = 2k^2$. We view n as consisting of k blocks of $2k = 2^{\ell+1}$ bits each and index each block by binary strings of length $\ell + 1$.

Alice's input of length $k(\ell + 1)$ to the INDEX problem can be viewed as k indices for the 1's, one per block which defines a set $A \subseteq [n]$ of size k as that set of indices.

Bob's input in $[k(\ell + 1)]$ determines an element of $[k] \times [\ell + 1]$ which gives the name i of a block together with an element j of $[\ell + 1]$ This yields a set B consisting of all those positions in the i-th block that are indexed by bit strings with a 1 in the j-th position. There are $2k$ coordinates in each block, half of which have a 1 in the j-th position so $|B| = k$.

Finally observe that by construction A and B can intersect in at most one spot. They do so only if the index selected by Bob's input was a 1 in Alice's original input to the INDEX problem. $\square$

However, as shown by Hastad and Wigderson, $R_{1/3}(UDISJ_{k,n}) = O(k)$ so this lower bound does not generalize.

One way to try to attack this would be to consider a more general class of hard inputs for the associated communication complexity problem. Define the communication complexity problem k-HAM where Alice and Bob get sets $A, B \subseteq [n]$ respectively and they need to determine whether or not $|A \oplus B| = k$. The above reduction shows that k-HAM reduces to the property testing problem for $2k$-juntas.

Open Question Is $R_{1/3}(k\text{-HAM}) = O(k)$ or $\Omega(k \log k)$.