CS 880: Advanced Complexity Theory 1/30/2008

Lecture 4: Linearity Testing

Instructor: Dieter van Melkebeek Scribe: Dalibor Zelený

Last time we discussed the basics of Fourier transforms over finite groups and the Boolean cube. Today we focus on linear functions that map points on the Boolean cube to bits. We develop a test for linearity, which will be our starting point for a later test for dictatorship.

1 Testers

Before we start discussing testers for various properties of functions, we formally define what testers are, and prove some facts about them.

Definition 1. *A* tester *for property $\mathcal{P} \subseteq \{f : \{0,1\}^n \to \{0,1\}\}$ is an efficient randomized oracle Turing machine T such that for all $\epsilon > 0$,*

- *If $f \in \mathcal{P}$, $T^f(\epsilon)$ accepts with high probability.*
- *If $D(f, \mathcal{P}) \geq \epsilon$, $T^f(\epsilon)$ accepts with small probability.*

We consider T efficient if its running time is a polynomial in n and $1/\epsilon$.

Definition 2. *A* local tester *for property $\mathcal{P} \subseteq \{f : \{0,1\}^n \to \{0,1\}\}$ is an efficient randomized oracle Turing machine T that makes $O(1)$ oracle queries, such that the following hold:*

- *For all $f \in \mathcal{P}$, $\Pr[T^f \text{ accepts}] = 1$.*
- *For all f, $\Pr[T^f \text{ rejects}] \geq \Omega(D(f, \mathcal{P}))$.*

We consider T efficient if its running time is a polynomial in n.

The most important difference between testers and local testers is the fact that local testers allow only a costant number of queries, whereas in the case of a tester the number of oracle queries can be arbitrary as long as the running time is polynomial.

Testers and local testers are related. In particular, if we have a local tester for some property, we can use it to create a tester for that property. The following claim states the relationship.

Claim 1. *Given a local tester T for property $\mathcal{P}$, we can construct a tester T' for $\mathcal{P}$ that makes $O(1/\epsilon)$ queries.*

Proof. The tester T' runs the local tester T $O(1/\epsilon)$ times. If T rejects on at least one of these runs, T' rejects as well. Otherwise, T' accepts. This yields a tester for $\mathcal{P}$.

To see that, notice that if $f \in \mathcal{P}$, T always accepts, so T' always accepts. Hence, if $f \in \mathcal{P}$, $T^f(\epsilon)$ accepts with probability 1 for any ϵ. Now suppose $D(f, \mathcal{P}) \geq \epsilon$. Then T^f rejects with probability $\Omega(\epsilon)$, so it accepts with probability no more than $1 - c\epsilon$ for some constant $c > 0$. To reduce the error probability to a small constant, say $1/e$, and thus to satisfy the second property of a tester, $1/(c\epsilon)$ independent runs of T^f are sufficient. The probability that T^f accepts on all of those runs is no more than $(1 - c\epsilon)^{1/(c\epsilon)} = (1 - 1/m)^m \leq 1/e$, where $m = 1/(c\epsilon)$. □

2 A Local Tester For Linearity

Given our convention that $f : \{-1,1\}^n \to \{-1,1\}$, we say that a function is linear if for all x, y we have $f(x \cdot y) = f(x)f(y)$, where the multiplication on the left-hand side is componentwise, and the multiplication on the right-hand side is the usual multiplication. Notice that this definition of linearity coincides with the definition of a character.

In the remainder of this section, we describe one linearity test and prove that it satisfies the definition of a local tester. We use the following test:

1. Pick x, y uniformly at random from $\{-1,1\}^n$.

2. If $f(x \cdot y) = f(x)f(y)$, accept. Otherwise reject.

Theorem 1. *The algorithm we just described is a local tester for linearity.*

Proof. If f is linear, the test succeeds with probability 1 because all pairs of x and y satisfy the acceptance criterion. Now assume that f is not linear. Our aim is to show that in that case, our test rejects with probability at least $D(f, LIN)$. For that, we need the following lemma.

Lemma 1. $\Pr[T^f$ *rejects*$] = \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]}\left(\widehat{f}(S)\right)^3$.

We prove this lemma later. First let's see how it helps us complete the proof of Theorem 1. Notice that

$$\sum_{S\subseteq[n]}\left(\widehat{f}(S)\right)^3 \leq \left(\max_{S\subseteq[n]}\widehat{f}(S)\right)\sum_{S\subseteq[n]}\left(\widehat{f}(S)\right)^2.$$

By Parseval's equality, the summation of the squared Fourier coefficients evaluates to 1, so

$$\sum_{S\subseteq[n]}\left(\widehat{f}(S)\right)^3 \leq \max_{S\subseteq[n]}\widehat{f}(S).$$

This implies that T^f rejects with probability at least $\frac{1}{2} - \frac{1}{2}\max_{S\subseteq[n]}\widehat{f}(S) = D(f, LIN)$. □

Now we prove Lemma 1.

Proof. (Lemma 1) Note that if T^f accepts, then $f(x)f(y)f(xy) = 1$, whereas if T^f rejects, $f(x)f(y)f(xy) = -1$. Thus

$$E_{x,y}[f(x)f(y)f(x\cdot y)] = \Pr[T^f \text{ accepts}] - \Pr[T^f \text{ rejects}] = 1 - 2\Pr[T^f \text{ rejects}]. \tag{1}$$

We also have that

$$\begin{aligned} E_{x,y}[f(x)f(y)f(xy)] &= E_{x,y}\left[\left(\sum_{S\subseteq[n]}\widehat{f}(S)\chi_S(x)\right)\left(\sum_{U\subseteq[n]}\widehat{f}(U)\chi_U(y)\right)\left(\sum_{V\subseteq[n]}\widehat{f}(V)\chi_V(x\cdot y)\right)\right] \\ &= \sum_{S,U,V\subseteq[n]}\widehat{f}(S)\widehat{f}(U)\widehat{f}(V)E_{x,y}\left[\chi_S(x)\chi_U(y)\chi_V(x)\chi_V(y)\right] \\ &= \sum_{S,U,V\subseteq[n]}\widehat{f}(S)\widehat{f}(U)\widehat{f}(V)E_x\left[\chi_S(x)\chi_V(x)\right]E_y\left[\chi_U(y)\chi_V(y)\right] \\ &= \sum_{S\subseteq[n]}\left(\widehat{f}(S)\right)^3 \end{aligned} \tag{2}$$

We obtained the first equality by taking the Fourier expansion of f. The second equality comes from linearity of expectation and the defining property of characters. The third equality holds because x and y are independent. Finally, since the characters χ are orthonormal, we get a nonzero contribution to the summation if and only if $S = U = V$ (otherwise at least one of the expectations turns out to be zero because it can be viewed as an inner product), which yields the last line.

When we combine (1) and (2), we get the proof of Lemma 1. □

We point out that a similar strategy can be used to analyze the acceptance probability of many tests, as long as they make their queries in a nonadaptive way. We can view the query answers $f(x)$, $f(y)$, and $f(x \cdot y)$ on the left-hand side of (2) as variables, and express the acceptance criterion of the test as a multilinear function in those variables. We then write each of the query answers using the Fourier expansion of f, and work out the expectation of the resulting expression.

3 A Tester For Small Linear Functions

Our final goal is to find a tester that distinguishes dictators from functions that are far from dictators. As a first step, we show how we can tweak the local tester for linearity T from the previous section to create a tester T' for small linear functions.

We know from the previous section that the probability that our local tester T accepts is

$$\frac{1}{2} + \frac{1}{2}\sum_S \left(\widehat{f}(S)\right)^3 \leq \frac{1}{2} + \frac{1}{2}\max_S \widehat{f}(S). \tag{3}$$

Intuitively, if the probability of acceptance is high, there is an S for which the Fourier coefficient $\widehat{f}(S)$ is high, which means that f correlates well with the corresponding function χ_S. To ensure that χ_S is a small linear function, we would like the set S that maximizes $\widehat{f}(S)$ to be small ($|S| = 1$ in case of a dictatorship). However, this is not guaranteed by (3). We need to change the quantity we are maximizing in (3) to enforce that a small S maximizes it. One way to do this is to replace $\widehat{f}(S)$ on the right-hand side of (3) by $\alpha^{|S|}\widehat{f}(S)$ for some constant $0 < \alpha < 1$. Since $\widehat{f}(S) \leq 1$, this makes $\alpha^{|S|}\widehat{f}(S)$ for large S so small that it cannot be the maximum.

Our goal now is to reverse-engineer this idea. That is, we start with the goal of replacing $\widehat{f}(S)$ on the right-hand side of (3) by $\alpha^{|S|}\widehat{f}(S)$, and trace our way back through the analysis to see how we need to modify the local test for linearity to get that replacement. It suffices to replace one of the factors $\widehat{f}(S)$ on the left-hand side of (3) by $\alpha^{|S|}\widehat{f}(S)$. Tracing back through (2), this means we need to replace one of the applications of f on the left-hand side of (2) by g, where g is a function such that $\widehat{g}(S) = \alpha^{|S|}\widehat{f}(S)$. Note that g won't be a Boolean function unless f is constant. This is because $\sum_S \left(\widehat{g}(S)\right)^2 \leq \sum_S \left(\widehat{f}(S)\right)^2 = 1$, where the inequality is strict unless f is constant. If we let h be a function such that $\widehat{h}(S) = \alpha^{|S|}$, we have that $\widehat{g}(S) = \widehat{h}(S)\widehat{f}(S)$, so $g = (h * f)$. In order to compute g, we first compute h and then work out the convolution.

View a string $v \in \{1, 1\}^n$ both as a string and as the subset of $[n]$ with ± 1-characteristic sequence v, i.e., $i \in [n]$ is in the set v if $v_i = -1$. Then $|v|$ denotes the size of the set, i.e., the number of positions in the string v that are set to -1. To get h, we compute

$$\begin{aligned}
h(v) &= \sum_{S\subseteq[n]} \widehat{h}(S)\chi_S(v) \\
&= \sum_{S\subseteq[n]} \alpha^{|S|}(-1)^{|v\cap S|} \\
&= \left(\sum_{S_1\subseteq v} \alpha^{|S_1|}(-1)^{|S_1|}\right)\left(\sum_{S_2\subseteq[n]-v} \alpha^{|S_2|}\right) \\
&= \left(\sum_{k=0}^{|v|} \binom{|v|}{k}(-\alpha)^k\right)\left(\sum_{l=0}^{n-|v|} \binom{n-|v|}{l}\alpha^l\right) \\
&= (1-\alpha)^{|v|}(1+\alpha)^{n-|v|}
\end{aligned}$$

The first line is the Fourier expansion of h. On the second line, we use the fact that $\chi_S(v) = \prod_{j\in S} v_i$. We rewrite the sum in a different way and get the third line. This works because we can partition S into $S_1 = S\cap v$ and $S_2 = S \cap \overline{v}$. To get the fourth line, we rewrite the two sums by regrouping terms based on the cardinality of the sets S_1 and S_2. Finally, we obtain the last line by applying the binomial theorem to each of the two terms in the product on the fourth line.

Notice that all values of $h(v)$ are nonnegative, provided $|\alpha| \leq 1$. We know that $E_v[h(v)] = \widehat{h}(\emptyset) = \alpha^{|\emptyset|} = 1$, so $h(v)/2^n$ denotes a probability distribution over all strings $v \in \{-1, 1\}$. We claim that $h(v)/2^n$ equals the probability of obtaining v using the following experiment. We set each bit of a string of length n independently to -1 with probability $\frac{1-\alpha}{2}$, and set it to 1 otherwise (i.e., with probability $\frac{1+\alpha}{2}$). Let V be the random variable representing the string obtained by this process. Then $\Pr[V = v] = \frac{1}{2^n}(1-\alpha)^{|v|}(1+\alpha)^{n-|v|}$. Thus, the last line of the expression for $h(v)$ corresponds to $2^n \Pr[V = v]$, so

$$\Pr[V = v] = \frac{1}{2^n}h(v). \tag{4}$$

Now we can find what g is.

$$\begin{aligned}
g(u) &= (h * f)(u) \\
&= E_v[h(v)f(u\cdot v)] \\
&= \frac{1}{2^n}\sum_v h(v)f(u\cdot v) \\
&= \sum_v \Pr[V = v]f(u\cdot v)
\end{aligned}$$

The first line follows from the definition of g and the convolution property of the Fourier transform. The second line is the expression for a convolution of two functions. On the third line, we express the expectation from the second line as a sum. Finally, we get the fourth line using (4). The final expression is the expected value of the random variable $f(u\cdot V)$ where the distribution of V is given by (4).

Recall that we need to replace one application of f on the left-hand side of (2) by g. Say we replace $f(x\cdot y)$ by $g(u)$, where $u = x\cdot y$.

Now, consider the following experiment. We start with a string u, and flip each bit independently with probability $\frac{1-\alpha}{2}$. This gives us some string z, and we output $f(z)$. The expected value of this process is exactly $g(u)$ because this process is equivalent to picking another string v, setting each of its bits independently to -1 with probability $\frac{1-\alpha}{2}$, and computing $z = u \cdot v$. Thus, we have that

$$E_{x,y}[f(x)f(y)g(x \cdot y)] = E_{\substack{x,y \\ z \sim_{\frac{1-\alpha}{2}} x \cdot y}} [f(x)f(y)f(z)]. \tag{5}$$

The right-hand side of (5) is equal to $\Pr[T_\alpha^f \text{ accepts}] - \Pr[T_\alpha^f \text{ rejects}]$, where T_α operates as follows.

1. Pick $x, y \in \{-1, 1\}^n$ uniformly at random.
2. Compute $x \cdot y$ and then flip each bit in that product with probability $\frac{1-\alpha}{2}$ to get a string z.
3. Accept if $f(x)f(y) = f(z)$. Otherwise reject.

The machine T_α is the tester for small linear functions that we wanted to construct; its probability of acceptance is given by

$$\Pr[T_\alpha^f \text{ accepts}] = \frac{1}{2} + \frac{1}{2} \sum_{S \subseteq [n]} \alpha^{|S|} \big(\widehat{f}(S)\big)^3.$$

4 Closing Remarks

Next lecture, we will further analyze the test T_α and extract a test for dictatorship. The linearity and dictatorship tests play a crucial role in the PCP theorems, which we will also discuss next time.