CS 59000 CTT Current Topics in Theoretical CS	Oct 18, 2012
Lecture 18	
Lecturer: Elena Grigorescu	*Scribe: Gowtham Kaki*

Today we'll introduce locally testable codes(LTCs), see connections with other objects studied in complexity theory (PCPs), and introduce the Fourier analytic technique that we will use in the next lecture to show that the Hadamard code is a locally testable code.

1 Intro to Locally Testable Codes

LTCs are error-correcting codes that can admit fast membership testers. To motivate these codes, first recall that given an (n, k, d)-error correcting code and a received word r one can uniquely decode it when it has at most $\frac{d}{2}$ errors. If more errors occur then a classical algorithm would still run the decoding procedure but have no guarantees about the output. In a previous lecture we saw a decoding algorithm that was efficient (it ran in linear time), but this would still be huge waste of time if by the end we realize that no codeword is output. We would be able to avoid this wasteful computation if we could quickly test whether the received word is close to some codeword. This is exactly what a locally testable code is: one for which we could test in sub-linear time if a received word is close to a codeword or not. We remark that though not all codes are locally testable, most of the codes commonly used in communication and storage are.

Definition 1 (Locally Testable Code) *Let $C = \{C_n\}_{n\to\infty}$ where C_n is an (n, k_n, d_n) error correcting code. Then, C is $(k, \epsilon_1, \epsilon_2, \delta)$-locally testable if there exists a property testing algorithm A such that, when given oracle access to $r \in \{0,1\}^n$, makes k queries and satisfies*

(Completeness) if $r \in C$, then accepts it with probability atleast $1 - \epsilon_1$

(Soundness) if r is δ-far from C, then accepts it with probability atmost ϵ_2.

So an LTC is just a testable property consisting of codewords. Given the above definition, the following questions are interesting

- What codes are locally testable?
- What parameters can be achieved for locally testable codes?

As already mentioned, in the next lecture we will see an example of a LTC. LTCs are combinatorial objects related to Probabilistically Checkable Proofs (PCPs). While no formal reductions between these two objects is known, currently the settings of parameters that can be achieved for these objects (number of queries vs length) are very similar. We will next give just a quick flavor of what PCPs are, to give some intuition about the connection with LTCs.

Definition 2 (Deterministic Polynomial Time Verifier) *A deterministic polynomial time verifier V for a language L is a an algorithm running in polynomial time that takes as input a string x and a proof π and accepts/rejects.*

The language verified (not decided) by V is $L = \{x | \exists \pi, V(x, \pi) = ACCEPT\}$. So, V satisfies the following properties

(Completeness) $\forall x \in L$, $\exists \pi$ such that $V(x, \pi)$ returns ACCEPT
(Soundness) $\forall x \notin L$, $\forall \pi, V(x, \pi)$ returns REJECT
So NP = class of all languages $L(V)$ for some deterministic verifier V.

Definition 3 (Randomized Verifier) *An (r, k)- verifier V for a language L is a randomized algorithm that*

1. *Takes as input a string x and a proof π to which it has oracle access,*
2. *Makes r random choices,*
3. *Makes k queries to π, and*
4. *Accepts/rejects.*

The language defined by V is L s.t.
(Completeness) $\forall x \in L$, $\exists \pi$ such that $V^{\pi}(x)$ always returns ACCEPT;
(Soundness) $\forall x \notin L$, $\forall \pi, V^{\pi}(x)$ returns REJECT with probability $> \frac{1}{2}$
The proof π is a probabilistically checkable proof. $PCP[r, k]$ = class of languages that have an (r, k)-verifier.

In particular, the class of languages decidable with no coin flips and no queries to a proof form P. Also the languages decidable with no randomness and a $poly(n)$ queries are exactly those in NP. (I.e. $PCP(0, poly(n)) = NP$.) The PCP theorem states one of the most surprising results in computational complexity: any language in NP has a verifier that makes only a constant number of queries to a proof and tosses very few coins.

Theorem 4 (PCP Theorem) *[1, 2] $NP = PCP(O(\log n), O(1))$*

We will next proceed with introducing the Fourier analytic method, which is a technique with numerous applications in complexity theory, including applications to the proof of the PCP theorem.

2 Fourier Analysis Of Boolean Functions

The Fourier analytic method has been very useful in the study of Boolean functions. The idea is to express a function in a particularly nice basis, the Fourier basis.

We will study Boolean functions defined as

$$f : Z_2^n \to Z_2.$$

It is sometimes more convenient to make the notation switch to a real valued function with ± 1 values: i.e replace bit b by $(-1)^b$ (so $0 \mapsto 1$ and $1 \mapsto 0$) and now the function is

$$f : Z_2^n \to \{-1, 1\}$$

The Fourier basis For all $a \in Z_2^n$ Let us define a boolean function χ_a as follows

$$\chi_a(x) = (-1)^{a \cdot x}$$

where $a \cdot x$ denotes inner product of vectors a and $x \mod 2$. We will later show that the vectors $\{\chi_a\}_{a \in \mathbb{Z}_2^n}$ form a basis of R^{2^n} (the real vector space of dimension 2^n.)

We will proceed to show useful properties of χ_a.

Proposition 5 $\mathbb{E}_{x \in Z_2^n} \chi_a(x) = \begin{cases} 1, & \text{if } a = 0 \\ 0, & \text{ow} \end{cases}$

Proof When $a = \bar{0}$, then RHS reduces to $\frac{1}{2^n} \sum 1 = 1$.

If $a \neq 0$, let i be s.t. $a_i \neq 0$, then we can pair up any x with $x + e_i$ since $a \cdot x \neq a \cdot (x + e_i)$, and so $(-1)^{a \cdot x} + (-1)^{a \cdot (x + e_i)} = 0$. ■

Proposition 6 $\forall a, x, y \in z_2^n,\ \chi_a(x).\chi_a(y) = \chi_a(x + y)$

Proof Trivially follows from the the additive property of exponents. ■

Proposition 7 $\forall a, b, x \in z_2^n,\ \chi_a(x).\chi_b(x) = \chi_{a+b}(x)$

Proof Trivially follows from the the additive property of exponents. ■

Definition 8 (Inner Product) *The inner product on pairs of functions $f, g : \mathbb{Z}_2^n \to \pm 1$ is*

$$\langle f, g \rangle = \frac{1}{2^n} \sum_{x \in Z_2^n} f(x) \cdot g(x) = \mathbb{E}_{x \in Z_2^n}(f(x) \cdot g(x))$$

It follows that the $\ell_2 - norm$ of f is

$$\|f\| = \sqrt{\mathbb{E}[f^2]} = \sqrt{\langle f, f \rangle}$$

Proposition 9 $\{\chi_a\}_{a \in Z_2^n}$ *forms an orthonormal basis of* $\mathbb{R}^{2^n}$

Proof

$$\|\chi_a\|^2 = \langle \chi_a, \chi_a \rangle = \frac{1}{2^n} \sum_{x \in Z_2^n} \chi_a(x)^2 = 1.$$

$\langle \chi_a, \chi_b \rangle = \frac{1}{2^n} \sum_{x \in Z_2^n} \chi_{a+b}(x) = 0$, whenever $a \neq b$ (follows from proposition 5).

So, they are orthogonal. Since these are 2^n orthonormal vectors in R^{2^n} they must form a basis. ■

Definition 10 (Fourier Expansion of f) *Every function $f : \{-1, 1\}^n \to \mathbb{R}$ can be uniquely expanded as*

$$f = \sum_{a \in Z_2^n} \hat{f}_a \chi_a$$

This expansion is called Fourier expansion of f and $\hat{f}_a$ is called Fourier coeffecient of f.

Proposition 11 *For a boolean function f and its fourier coeffecient $\hat{f}_a$,*

$$\hat{f}_a \quad = \quad \langle f, \chi_a \rangle$$

Proof

$$\begin{aligned}
\langle f, \chi_a \rangle &= \mathbb{E}_x f(x)\chi_a(x) = \mathbb{E}_x(\sum_b \hat{f}_b\chi_b(x))\chi_a(x) \\
&= \mathbb{E}_x \sum_b \hat{f}_b\chi_b(x)\chi_a(x) = \mathbb{E}_x \sum_b \hat{f}(b)\chi_{a+b}(x) \\
&= \sum_b \hat{f}_b\mathbb{E}_x\chi_{a+b}(x) = \hat{f}_a,
\end{aligned}$$

where we used that when $a \neq b$, $\chi_{a+b}(x) = 0$, and so, the only term remaining above is when $a = b$. ■

Theorem 12 (Parseval's Theorem) *For any boolean function f,*

$$\langle f, f \rangle = \mathbb{E}_{x \in Z_2^n} f^2 = \textstyle\sum_{a \in Z_2^n} \hat{f}_a^2 = 1$$

Proof

$$\langle f, f \rangle = \left\langle \textstyle\sum_{a \in Z_2^n} \hat{f}_a\chi_a, \sum_{a \in Z_2^n} \hat{f}_a\chi_a \right\rangle = \textstyle\sum_{a,b \in Z_2^n} \left\langle \hat{f}_a\chi_a, \hat{f}_b\chi_b \right\rangle = \sum_{a \in Z_2^n} \hat{f}_a^2$$

■

References

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[2] Irit Dinur. The pcp theorem by gap amplification. *J. ACM*, 54(3):12, 2007.