Notes for Lecture 10

Today we continue the study of average-case complexity. We improve the polynomial reconstruction procedure we saw last time so that it corrects up to almost a fraction 1/2 of errors, and we interpret it as a sub-linear time decoder for an error-correcting code.

1 Improved Polynomial Reconstruction

We will use the following property of pairwise independent random variables.

Lemma 1 *Let $X_1, \ldots, X_m$ be pairwise independent 0/1 random variables. Then for every $\delta > 0$ we have*

$$\mathbf{Pr}\left[\left|\sum_i X_i - \mathbf{E}\left[\sum_i X_i\right]\right| \le \sqrt{\frac{m}{\delta}}\right] \ge 1 - \delta$$

Proof: For each i, $\mathbf{Var}[X_i] \le \mathbf{E}[(X_i)^2] \le 1$, so $\mathbf{Var}[\sum_i X_i] \le m$. The rest follows from Chebyschev's inequality stating that for a random variable X

$$\mathbf{Pr}[|X - \mathbf{E}[X]| > k] < \mathbf{Var}[X]/k^2$$

□

Theorem 2 *Suppose we have oracle access to a function $f : \mathcal{F}^n \to \mathcal{F}$ and f agreees with some degree d polynomial $p : \mathcal{F}^n \to \mathcal{F}$ on a fraction $\frac{1}{2} + \epsilon$ of its inputs where $|\mathcal{F}| \ge 1 + \max\{64/\epsilon^2, d/\epsilon\}$. Then we can compute $f(x)$ with high probability for all $x \in \mathcal{F}^n$ in poly(n, d) time, assuming field operations can be performed in poly(n, d) time.*

Proof: The main idea in the proof is the apply the algorithm we saw last time *recursively*. As before, on input x we pick a random line $l(t) = x + ty$, and we try to compute the univariate polynomial $q()$ such that $q(t) = p(l(t))$. From Berlekamp-Welch we know we can find $q()$ if only we could know the value of $p(x + ty)$ for more than $(|\mathcal{F}| + d)/2$ values of t. In our previous algorithm, we just used $f(x+ty)$ as an "estimate" of $p(x+ty)$; this time we recursively use our previous algorithm for this approximation. That is, we pick at random another point z, and for every $t \neq 0$ we consider the line $l_t(s) = x + ty + sz$. We then try to compute the polynomial q_t such that $q_t(s) = p(x + ty + sz)$ for every s, and finally use $q_t(0)$ as our estimate for $p(x + ty)$. What did we gain? Now the points $(x + ty + sz)$ are *pairwise independent*, so the accuracy of $f()$ in predicting $p()$ along each line $l_t(s)$ is concentrated around its expected accuracy $1/2 + \epsilon$, and so on most of these lines the accuracy is good enough to reconstruct $q_t()$ and so $p(x + ty)$. Once we have most of the values $p(x + ty)$ we can then reconstruct $p(x)$. Details follow.

To simplify notation, let us call $m = |\mathcal{F}| - 1$ the number of non-zero field elements. This is our reconstruction algorithm.

```
input: x,
  pick y, z ∈ F^n
  for each t ∈ F − {0}
    find univariate polynomial q_s() such that
      |{s ∈ F − {0} : q_t(s) ≠ f(x + ty + sz)}| < (m − d)/2
    halt and fail if no such polynomial exists
    a_t ← q_t(0)
  find univariate polynomial q() such that
    |{t ∈ F − {0} : q(t) = a_t}| < (m − d)/2
  halt and fail if no such polynomial exists
  return (q(0))
```

Let us analyse the success probability of the algorithm.

First of all, it easy to check that for every $t \neq 0$ and for every $s \neq s'$, the random variables $x + ty + sz$ and $x + ty + s'z$ are uniform and independent over the choices of y and z. Then, we use pairwise independence to prove the following claim.

Claim 3 *For each $t \neq 0$, we have*

$$\mathbf{Pr}_{y,z}[|\{s \in \mathcal{F} - \{0\} : q_t(s) \neq f(x + ty + sz)\}| < (m - d)/2] \geq 15/16$$

PROOF: The cardinality $|\{s \in \mathcal{F} - \{0\} : q_t(s) \neq f(x+ty+sz)\}|$ can be seen as the sum of m pairwise independent random variables, each having expectation at most $1/2 - \epsilon$. Therefore we have

$$\mathbf{Pr}_{y,z}\left[|\{s \in \mathcal{F} - \{0\} : q_t(s) \neq f(x + ty + sz)\}| \leq \frac{m}{2} - \epsilon m + 4\sqrt{m}\right] \geq 15/16$$

and the assumption of the Theorem guarantee that $m/2 - \epsilon m + 16\sqrt{m} < m/2 - d/2$, because $\epsilon m - 4\sqrt{m} \geq \epsilon m/2$ and $\epsilon m \geq d$. □

It follows that the average number of t such that $a_t \neq p(x + ty)$ is at most $m/16$. Then we can use Markov to deduce that with probability at least $3/4$ over the choices of y and z there at most $m/4$ values of t such that $a_t \neq p(x + ty)$. Whenever this happens, since $m/4 < (m - d)/2$, the algorithm correctly computes $q()$ and correctly outputs $p(x)$. □

2 Error-Correcting Codes

The motivation for error-correcting codes is to reliably send information over channels that introduce errors. An error-correcting code is a way of an encoding a message so that even if the encoding is corrupted in a bounded number of places it is still possible to reconstruct the original message.

Let Σ be an arbitrary finite set. For most of the following discussion it is useful to think of Σ as either $\{0, 1\}$ or some other finite field.

Let $z, z' \in \Sigma^m$ be two strings: we say that the *Hamming distance* between z and z' is the number of entries where z and z' disagree. The *normalized* Hamming distance, that we denote by $d(z, z')$ is the *fraction of entries where they disagree*, that is:

$$d(z,z') = \mathbf{Pr}_{i\in\{1,\ldots,m\}}[z_i \neq z'_i]$$

A mapping $C : \Sigma^n \to \Sigma^m$ is an *error-correcting code* with *minimum relative distance* δ if for every two inputs $x, x' \in \Sigma^n$ we have $d(C(x), C(x')) \geq \delta$.

Suppose then that we want to send a message x over a channel, and that the channel allows us to send m elements of Σ and it is guaranteed that with high probability the number of transmission errors is less than $\delta m/2$. Then we compute and send $C(x)$. The receiver receives a string z such that $d(C(x), z) < \delta/2$. Now, z uniquely defines x, because the receiver could compile a list of all the x' such that $d(C(x'), z) < \delta/2$: this list will contain x, and it will not contain any other element $x' \neq x$, because otherwise from $d(C(x'), z) < \delta/2$ and $d(C(x), z) < \delta/2$ it would follow $d(C(x), C(x')) < \delta$ that contradicts the assumption that C has minimum relative distance δ.

The goal is then to find codes $C()$ where m is not too much larger than n, where δ is as large as possible, $C()$ is efficiently compute (in polynomial or even linear time in n), and given a string z it is possible to efficiently (in polynomial or preferably linear time in n) find the unique x (if any) such that $d(C(x), z) < \delta/2$.

We have already observed (and used) the fact that two distinct polynomials of degree d over a field F can agree at most on a fraction $d/|F|$ of the inputs and, in particular, that two distinct univariate polynomials of degree d can agree on at most d inputs. This observation, along with the Berlekamp-Welch algorithm, gives a good family of error-correcting codes: the Reed-Solomon codes.

In the Reed-Solomon codes, the messages are associated to polynomials of degree at most d over a finite field $\mathcal{F}$, and the encoding of the message is the evaluation of the polynomial at m points.

Let then $\mathcal{F}$ be a field, $d < |\mathcal{F}|$ be an integer, and $a_1, \ldots, a_m$, $m > d$ be distinct elements of $\mathcal{F}$. Then the Reed-Solomon code is a mapping

$$C : \mathcal{F}^{d+1} \to \mathcal{F}^m$$

where $C(c_0, \ldots, c_d)[i] = \sum_{j=0}^{d} c_j a_i^j$. That is, the message $(c_0, \ldots, c_d)$ is seen as the description of a univariate polynomial $p(x) = c_0 + c_1 x + \cdots + c_d x^d$, and the encoding is $(p(a_1), \cdots, p(a_m))$, the evalutation of the polynomial at points $a_1, \ldots, a_m$.

Since two different messages are encoded as the evaluations of two different degree-d polynomials, it follows that two different encodings agree in at most d entries, and so the code has minimum distance $m - d$ and minimum relative distance $(m-d)/m$.

Suppose now that we receive a string $(b_1, \ldots, b_m)$ and we know that there is a codeword within distance $< (m-d)/2$ from such a string. In order to decode, we need to find the coefficients of the unique degree-d polynomial p such that $|\{i : p(a_i) \neq b_i\}| < (m-d)/2$. This is precisely the problem solved by the Berlekamp-Welch algorithm!

Suppose now that we want to see the message as a multivariate polynomial, and the encoding as the evaluation of the polynomial at all points.

A degree d polynomial over n variables in the field $\mathcal{F}$ is specified by $\binom{n+d}{d}$ coefficients, so every sequence of $\binom{n+d}{d}$ field elements can be seen as a message. The encoding requires the evaluation of the polynomial at $\mathcal{F}^n$ points. Therefore we have a code, called the Reed-Muller code,

$$C : \mathcal{F}^{\binom{n+d}{d}} \to \mathcal{F}^{|F|^n}$$

The reader can check that the Schwartz-Zippel lemma implies that the Reed-Muller code has relative minimum distance $1 - d/|\mathcal{F}|$. Then, in principle, a fraction $1/2 - d/2|F|$ errors can be corrected. Theorem 2 gives a procedure to correct up to a $1/2 - d/2|F| - 4/\sqrt{|\mathcal{F}|})$ fraction of errors, that is almost as good. More notably, assuming that the fraction of errors is less than $1/2 - d/2|F| - 16/\sqrt{|\mathcal{F}|})$, the algorithm of Theorem 2 can reconstruct an entry of the correct codeword in time *polylogarithmic* in the length of the encoding.

From this perspective, the average-case complexity of the permanent is due to the fact that, for a fixed field $\mathcal{F}$ and size n, the values of the permanent of the matrices in $\mathcal{F}^{n\times n}$ form a codeword of the Reed-Muller code. A good-on-average algorithm for the permanent over $\mathcal{F}^{n\times n}$ can be seen as defining a string in $\mathcal{F}^{|\mathcal{F}|^{n^2}}$ that is close to the codeword corresponding to the permanent problem. Then the value of the permanent for a given matrix can be reconstructed by "decoding" the average-case algorithm.

When looking at the average-case complexity of problems in **EXP** and **PSPACE**, we could still use Theorem 2 if we are fortunate enough to find a **PSPACE**-complete or an **EXP**-complete problem such that its truth-table for inputs of a certain length can be seen as a codeword of the Reed-Muller code. Indeed, we will *construct* a **PSPACE**-complete and an **EXP**-complete problems that are just like that.

Since we want to construct a decision problem such that its truth table for inputs a certain length is a codeword of an error-correcting code, we need codes whose encodings are binary strings.

Consider the Reed-Muller code with $\mathcal{F} = \{0,1\}$. Then we must choose $d = 1$ and we get a code

$$C : \{0,1\}^{n+1} \to \{0,1\}^{2^n}$$

The code has a message for each string $c = (c_0, \ldots, c_n) \in \{0,1\}^n$, and the encoding of length 2^n has an entry for each string $a \in \{0,1\}^n$. The content of the a-th entry of the encoding of c is just $c_0 + a_1c_1 + \cdots a_nc_n$ where operations are done modulo 2. Indeed, we will typically not use the bit c_0, in order to simplify notation, so we will have a code

$$C : \{0,1\}^n \to \{0,1\}^{2^n}$$

such that the a-th entry of the message $(c_1, \ldots, c_n)$ is $\sum_i a_ic_i$. We call this code the *Hadamard code*. The relative minimum distance of this code is $1/2$, which follows from the general formula $1 - d/|\mathcal{F}|$ with $d = 1$ and $|\mathcal{F}| = 2$.

Now suppose we have a code, for example a Reed-Muller code, that uses an alphabet Σ of size 2^k.

$$C : \Sigma^n \to \Sigma^m$$

Then we can obtain a binary code by re-encoding each entry of $C(x)$, which we can think of as an element of $\{0,1\}^k$, with the Hadamard code. Then we get a binary encoding

$$C' : \Sigma^n \to \{0,1\}^{2^k \cdot m}$$

It is easy to verify that if $C()$ had minimum relative distance δ, then C' has minimum relative distance $\delta/2$.