6.440 Essential Coding Theory April 17, 2013

Lecture 18

Lecturer: Madhu Sudan *Scribe: Sam Elder*

Today: Polar codes: decoding and speed of polarization (on BEC).
Scribes are needed for all future lectures; e-mail Madhu.
The main point of this class will be to prove the following theorem.

Theorem 1 (Arikan '07, Guruswami-Xia '13)**.** *There exists a polynomial f such that for all $\epsilon > 0$ and $0 < p < 1/2$, there are encoding and decoding schemes for $BSC(p)$ of rate $1 - H(p) - \epsilon$ such that encoding and decoding take time $f(1/\epsilon)$ (and therefore, the length is $\mathrm{poly}(1/\epsilon)$), and the decoding error is probability $\leq \exp(-(1/\epsilon)^{.49})$.*

If we had only achieved a constant probability of error, that would have been new to us, but we can do better, like this. The hard part here is getting an algorithm in $1/\epsilon$.

As we talked about last time, these are related to polar codes, which are related to the polarization phenomenon we saw last time: If we start with two bits x_1, x_2 and get out bits $y_1 = x_1 \oplus x_2$ and $y_2 = x_2$ with $H(x_1) = H(x_2) = H(p)$, then $H(y_1) = H(2p - 2p^2) > H(p)$, so $H(y_2|y_1) < H(p)$. That simply follows from the fact that these must sum to $2H(p)$. Notice that we're not talking about $H(y_2)$ itself, but conditioning on y_1.

We want to start with bits $u_1, \ldots, u_N$ and polarize them to $x_1, \ldots, x_N$, then send the x_i through a binary symmetric channel, getting out $y_1, \ldots, y_N$. So what we want to know is $H(u_i|u_1, \ldots, u_{i-1}, y_1, \ldots, y_n)$. We'd like this to be small, so in other words, u_i is very likely 0 or very likely 1.

So we're going to pair our bits up and xor them like this. We then take the first bits from result, those with higher entropy, and recursively apply the encoding E_{n-1} on those, and do the same with the second bits. We keep going until we've distinguished all of the bits.

Now we won't be describing how to find it, but there will be some cutoff where the entropy will sharply go from 0 to 1, i.e.

$$\Pr_{i \leftarrow [n]}[H(u_i|u_1, \ldots, u_{i-1}, y_1, \ldots, y_N) \in (\delta, 1 - \delta)] \to 0.$$

That is, almost all of the bits are either completely determined or completely undetermined from the previous ones. On the other hand, we also know that

$$\sum_{i=1}^{N} H(u_i|u_1, \ldots, u_{i-1}, y_1, \ldots, y_N) = H(u_1, \ldots, u_N|y_1, \ldots, y_N)$$
$$= H(x_1, \ldots, x_N|y_1, \ldots, y_N) = nH(p).$$

So this tells us that the fraction of i for which $H(u_i|u_1, \ldots, u_{i-1}, y_1, \ldots, y_N) > 1 - \delta$ is $\sim H(p)$. These are the bits that you can't figure out from the rest. To make this a good error-correcting code, we'll take exactly those i for which this is greater than δ, which will also be a $\sim H(p)$ fraction of the total. Call this set of bits F, for *frozen*. We don't look too carefully at what subset of bits these are.

Our encoding procedure is to take the message and map it to $[N] - F$. Set the remaining U_i (in F) to 0, and apply the transformation to get the encoding $X_1, \ldots, X_N$.

The decoding procedure will be the following: For $i \leftarrow [N]$, we need to determine the value of u_i. If $i \in F$, then we know $u_i \leftarrow 0$. On the other hand, if $i \notin F$, we compute the probability that $u_i = 0$ conditioned on $y_1, \ldots, y_N$ and $u_1, \ldots, u_{i-1}$. If this probability is greater than $1/2$, we set it to 0 and continue; otherwise, set it to 1 and continue.

If we make one mistake during these steps of decoding, that could be a big problem for the rest of the bits. But we only care about when we get it exactly right, and we've already deviated if we make that one mistake. What we want to show is that the probability that this decodes correctly is as high as we want. We'll find that it's very unlikely that we make a mistake at any stage, and then we'll use the union bound on all of these possible errors.

Now we need to estimate the probability of equalling 0 and so on. Given an arbitrary linear transform, we wouldn't easily be able to compute these probabilities, but this transform has a nice structure that we'll be able to take advantage of. Let's define $P_{u_1,\ldots,u_i,y_1,\ldots,y_N}$ to be the probability of $U_1 = u_1, \ldots, U_i = u_i, Y_1 = y_1, \ldots, Y_N = y_N$, where the capital variables are a way of denoting the random variables that those bits are assigned to. By Bayes' Rule, we get

$$\Pr[U_i = u_i | u_1, \ldots, u_{i-1}, y_1, \ldots, y_N] = \frac{P_{u_1,\ldots,u_i,y_1,\ldots,y_N}}{P_{u_1,\ldots,u_{i-1},y_1,\ldots,y_N}}.$$

Let's imagine one step of our algorithm. Let our bits be $u_1, \ldots, u_N$, going to $v_1 = u_1 \oplus u_2$, $v_2 = u_2$, and so on, and then the odd-numbered bits go to one encoding and the even-numbered bits go to another smaller encoding, eventually yielding (respectively) $y_1, \ldots, y_{N/2}$ and $y_{N/2+1}, \ldots, y_N$.

If i is even, then after the $u_1, \ldots, u_i$ combine in pairs into $v_1, \ldots, v_i$, half of the results go to the odd encoder and the other half to the even encoder. Therefore, if we denote the probabilities of the n-level encoder with $P^{(n)}$, we get

$$P^{(n)}_{u_1,\ldots,u_i,y_1,\ldots,y_N} = P^{(n-1)}_{u_1 \oplus u_2, u_3 \oplus u_4, \ldots, u_{i-1} \oplus u_i, y_1, \ldots, y_{N/2}} \times P^{(n-1)}_{u_2,u_4,\ldots,u_i,y_{N/2+1},\ldots,y_N}.$$

When i is odd, we can just reduce to the even case:

$$P^{(n)}_{u_1,\ldots,u_i,y_1,\ldots,y_N} = \frac{1}{2}\left(P^{(n)}_{u_1,\ldots,u_i,0,y_1,\ldots,y_N} + P^{(n)}_{u_1,\ldots,u_i,1,y_1,\ldots,y_N}\right).$$

Of course, you can always do this sort of expansion, but this is nice because we don't have to do too much of it: Just two terms to sum for the odd values. Then for the base case, the binary symmetric channel gives us the 0th-level expression:

$$P^{(0)}_{x_1,\ldots,x_i,y_1,\ldots,y_N} = (1-p)^{i-\Delta(x_1,\ldots,x_i,y_1,\ldots,y_i)} p^{\Delta(x_1,\ldots,x_i,y_1,\ldots,y_i)}.$$

We can build this up inductively from here. This is a nice recursion; it's rare for general encoding schemes that we can write it down as easily as this.

Now we want to set δ, the bounds on our entropies, to much less than $1/N$, so that by the union bound, it's still very unlikely to make a mistake. Notice that what you get isn't necessarily the most likely vector given what you know; it's more like the "greediest." But we can easily find it and prove it's good enough.

Now we want to show that the probability that our entropies are within the range $(\delta, 1-\delta)$ is very small. What we'll do is take up the analysis in the special case of the Binary Erasure Channel, since the calculations are a bit nicer there. Again, decoding an erasure channel isn't a very surprising result, since you're just solving a system, but proving polarization in it is profound.

Let's see what happens in a single step of our erasure channel. Starting with $u_1, u_2 \in \{0, 1\}$, we get out $v_1, v_2 \in \{0, 1, ?\}$ from $v_1 = u_1 \oplus u_2$ and $v_2 = u_2$, with a probability p of erasure. We want to know $H(u_1|v_1, v_2)$ and $H(u_2|u_1, v_1, v_2)$. With just one input in $BEC(p)$ given by $x \to y$, if y is 0 or 1 then x is determined, so $H(x|y) = pH(1/2) = p$.

Suppose $H(v_1) = H(v_2) = \alpha$, and look at the entropy of u_2 conditioned on the rest: $H(u_2|u_1, v_1, v_2) = \Pr[u_2 = ?|v_1, v_2, u_1] = \Pr[v_1 = ?] \Pr[v_2 = ?] = p^2$ because if we know either v_1 or v_2, assuming you know u_1, you have enough to recover u_2. Therefore, $H(u_1|v_1, v_2) = 2\alpha - \alpha^2$.

So in our procedure, we replace two α's with α^2 and $2\alpha - \alpha^2$. We want to claim that by doing this repeatedly, we get either very close to 0 or very close to 1. How can we analyze this?

Here are the ideas of the analysis:

(1) At intermediate stages, show that the rough polarization so far is polynomial in the block lengths.

(2) Roughly polarized high entropy bits become highly polarized in the remaining steps.

To analyze the rough polarization, let $Z_i^{(l)} = \Pr[x_i^{(l)} = ?]$, where the l denotes the stage in our polarization this is in. This is the quantity we want to be close to either 0 or 1. Instead, we'll analyze $Y_i^{(l)} = \sqrt{Z_i^{(l)}(1 - Z_i^{(l)})}$.

What we then show is that $\|Y^{(l)}\|_1 < \frac{\sqrt{3}}{2}\|Y^{(l-1)}\|_1$. This will be good enough to prove rough polarization, so let's see why it's true.

In the L_1 norm, α, α give us $2\sqrt{\alpha(1-\alpha)}$, and $\alpha^2, 2\alpha - \alpha^2$ give us

$$\|Y^{(l)}\|_1 = \alpha\sqrt{1-\alpha^2} + \sqrt{(2\alpha-\alpha^2)(1-2\alpha+\alpha^2)} = \sqrt{\alpha(1-\alpha)}(\sqrt{\alpha(1+\alpha)} + \sqrt{(2-\alpha)(1-\alpha)}.$$

Now the quantity in parentheses is the dot product of vectors $x = (\sqrt{\alpha}, \sqrt{1-\alpha})$ and $y = (\sqrt{1+\alpha}, \sqrt{2-\alpha})$, which is at most $\sqrt{\|x\|\|y\|} = \sqrt{1 \cdot 3} = \sqrt{3}$. Therefore, $\|Y^{(l)}\|_1 \leq \frac{\sqrt{3}}{2}\|Y^{(l-1)}\|_1$.

So you can actually trace how quickly the polarization is happening. Then you find that the aveage number of polarized bits is exponential or close to exponential. It's not good enough for the full polarization, but it gets a rough polarization that you can then use to get a full polarization, but we won't see that.

For those who care, in the general BSC setting, you can still talk about the entropy of these bits, but the analysis tends to focus on the Bhattachorya parameter, which plays the role of the Z_i here. The argument is similar but more complicated. You still look at $Y = Z(1-Z)$. You can't prove something as simple as the L_2 norm of these vectors shrinking.

Finding polarization was a very information-theoretic approach. In the next lectures, we'll talk about other ways of using graphs in error-correcting codes.