CS 229r Essential Coding Theory, Lecture 15 March 21, 2017

Lecture 15: Polar Codes

Instructor: Madhu Sudan *Scribes: Christina Ilvento*

Administrivia

Office hours for the rest of the semester

- Tuesdays: 2:30-3:30
- Thursdays: 5-6 (but cancelled this week)

Today: Polar Codes

Until now, we've been covering primarily standard material in coding theory, and for the rest of the class we're going to cover more recent stuff (within the last 10 years).

Today: Polar Codes

- Motivation
- Idea
- Some initial proofs

Next time:

- Complete proofs

Motivation

We want to deal well with random errors: efficient encoding and decoding achieving capacity. Polar Codes, we'll see, are the ultimate answer to how well can we deal with random errors.

The Challenge

Consider the binary symmetric channel with probability p, $BSC(p)$,

$$BSC_p(x) = \begin{cases} x & \text{with } \Pr = p \\ \bar{x} & \text{with } \Pr = 1-p \end{cases}$$

Given an input bit $x \in \{0,1\}$, the output bit $y = x$ with probability p, $\bar{x}$ with probability $1-p$. We only deal with $p < 0.5$, as with $p = 0.5$ we learn nothing.

We know from Shannon, we can achieve Rate $R < 1 - H(p)$, the catch is that if we look at *constructive* results and we want to achieve rate $R = 1 - H(p) - \epsilon$, so far all of our results have required decoding time $\geq 2^{1/\epsilon^2}$. The challenge: Can we do better? Yes, with Polar Codes.[1]

[1] There may be other codes, but Polar Codes seem to be the cleanest.

Codes Thus Far

Let's revisit the codes we've seen so far in the course:

1. Existential Results: less interesting, because they don't have efficient algorithms
2. Algebraic codes: algorithms are poly-time, but they don't work great over small alphabets
3. Graph-theoretic codes: also good for worst case errors, have linear time algorithms, but haven't yet gotten us to capacity
4. Information Theoretic codes (today): depart a bit from the above and rely more heavily on information-theoretic concepts like conditional entropy

Polar Codes: Basic Construction Ideas

The first step in understanding the construction of polar codes is to understand that if we can get a good compression algorithm, we can get a good coding algorithm. That is, linear-time compression implies linear time decoding.

Compression to Decoding

First, recall the definition of a Bernoulli Random Variable

$$Bern(p) \triangleq x = \begin{cases} 0 \text{ with } \Pr = p \\ 1 \text{ with } \Pr = 1 - p \end{cases}$$

Take an x which is an n bit Bernoulli random variable $x \in Bern(p)^n$ and a function $f : \{0,1\}^n \to \{0,1\}^m$ for $m < n$ and $f^{-1} : \{0,1\}^m \to \{0,1\}^n$ where $m \approx H(p) * n$.

Claim: If f is linear, and $\Pr_{x \sim Bern(p)^n}[f^{-1}(f(x) \neq x] \to 0$, then we have an efficient encoding and decoding algorithm. Why?

- f is linear implies $f(x) = xH$ for some matrix H. We'll take H will be the parity check matrix of the code we're building
- Take c such that $cH = 0$ to be our codewords. The receiver gets $z = c + x$ where $x \sim Bern(p)^n$ and $f^{-1}((c + x)H) = f^{-1}(xH) = f^{-1}(f(x))$ which with high probability $= x$ by our assumption. So by applying f^{-1}, a linear function, we compute our error vector which we simply subtract from the received word z to recover c.
- So the decoding algorithm based on this is to just compute $z - f^{-1}(f(z))$.

As a side note, we could also pick H at random, but we won't know how to compute f^{-1} quickly; a key assumption in the above was that we had f and f^{-1} which are linear.

Constructing H

The new idea in Arikan's work is to build H very carefully so that it meets these requirements. To construct H we'll construct a larger matrix P which will act to concentrate entropy in certain parts of our output.

$$\begin{bmatrix} & x & \end{bmatrix} \begin{bmatrix} & & \\ & P & \\ & & \end{bmatrix} = \begin{bmatrix} xP_L & | & xP_R \end{bmatrix}$$

Where x is our n bit input, P is a square $n \times n$ matrix, and xP_L is the left-hand m bits of the result.

We'll pick P to be invertible, and we'll insist that $H((xP)_R|(xP)_L)$, the entropy of the righthand side of the result conditioned on the lefthand side of the result, is small or about $= o(1)$. The amount of uncertainty about the right hand side, once you know about the left is very very small, so xP_R will usually be determined by xP_L. If we can construct such a P, then we just take the m left columns of of P to be H.

Information Theory Background

Before we get to the proofs, we'll need to cover some background on information theory.

The entropy of a single variable: Take $z = (p_1, ..., p_n)$ where $p_i \triangleq Pr[z = i]$ and $H(z) \triangleq \sum p_i \log_2(\frac{1}{p_i})$. If entropy of z is going to 0, then z is almost completely determined. If $H(z) = k$, then z has roughly 2^k possible values.

Joint Distributions We'll often be interested in joint distributions, that is the pair of variables (x, y) with $\Pr[x = i, y = j] = p_{ij}$, which can be expressed in the same way as the above.

Conditional entropy The entropy of X conditioned on Y is written as:

$$H(X|Y) = \sum_j \Pr[Y = j] H(X|Y = j)$$

There are several key rules and expressions we'll come back to:

- **Chain rule:** $H(X, Y) = H(Y) + H(X|Y)$
- Conditional entropy is always less than or equal to unconditional entropy in expectation: $H(X|Y) \leq H(X)$ - for particular quantities it may increase, but in expectation it does not. To see this, consider the joint distribution for X and Y,

$$X = \begin{cases} 1 & \text{if } Y = 0 \\ \text{uniform}(n) & \text{if } Y = 1 \end{cases}$$

- The "entropy triangle inequality" $H(X, Y) \leq H(X) + H(Y)$
- $H(X) = \alpha$ for some $\alpha < 1$, then $\Pr[X = mode(X)] \geq 1 - \alpha$ Follows from the fact that $H(p) \geq p$ for $p < 1/2$.
- $H(X) = n \rightarrow supp(X) \geq 2^n$; the only way you get high entropy is if the support is large.
- If we just relabel the domain, the uncertainty/entropy doesn't change. Consider $x \sim \Delta([n])$, (x drawn from a distribution on $1, ..., n$). If we take a permutation π and apply it to x to generate y, we clearly have that $H(x) = H(y)$. **Simple Exercise:** Verify this is the case for any bijection.

Entropy is a way of managing uncertainty, and we are interested in uncertainty because we want to start with a high entropy value x, do some linear operations (apply P), shove the uncertainty towards the left and get the right hand side to be completely determined given the left hand side.

So our whole goal is to move the uncertainty from the right-hand side of the output to the left.

Constructing P

Before we get too far, recall that P is invertible, which means that the entropy of xP is the same as the entropy of x. We also have that $H(x) = nH(p)$ when $x \sim Bern(p)^n$. So we must have $H(p)n$ bits in the L side of our result, so $m \approx H(p) * n$. Now we'll show how to construct P to meet these requirements.

We'll use a simple iterative process to build up P, by building smaller sub-matrices first.

Each sub-matrix will polarize the entropy of its outputs. To see how this works, start by looking at x_1, x_2, independent Bernoulli bits and convert to a new two-bit sequence where the entropy is non-uniform. That is, we want:

- $H(x_1, x_2) = H(y_1, y_2)$
- $H(y_2|y_1) < H(x_2)$ which implies
- $H(y_1) > H(x_1)$

How to do this? Take $y_1 = x_1 \oplus x_2$ and $y_2 = x_2$. So given $x_1 \oplus x_2$, x_2 has low entropy.[2]

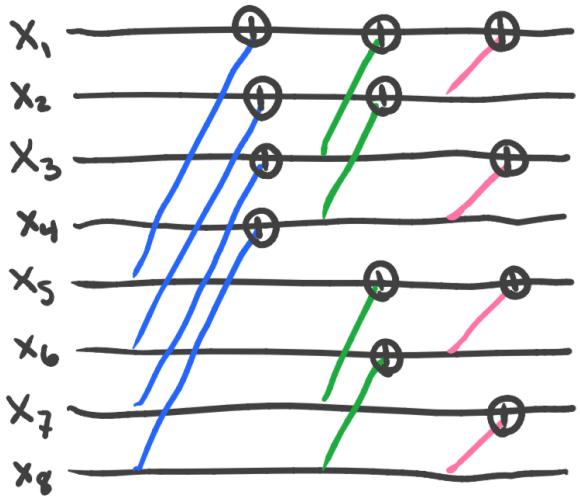
For example, let's look at the first sub-matrix P_1 on two bits:

$$P_0 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} x_1 \oplus x_2 & x_2 \end{bmatrix} = \begin{bmatrix} y_1 & y_2 \end{bmatrix}$$

We'll then do this process recursively, building up larger and larger transformations until our final matrix of P_{2^ℓ}.At each step, $x * P_n = (u * P_{n/2}, v * P_{n/2})$, where $u = (x_1 \oplus x_{n/2+1}, ..., x_{n/2} \oplus x_n)$ and $v = (x_{n/2+1}...x_n)$.

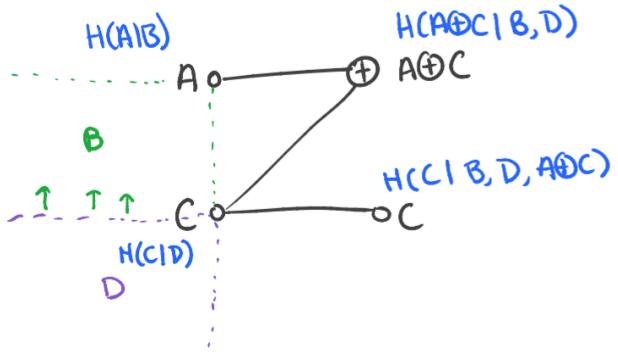
Which will give us the following pictorial representation:[3]



At any given $\oplus$ node, we have the following "Z" relation:

[2]We'll see the full proof of this next time, but recall that our Bernoulli random variables have $p < \frac{1}{2}$ and work out a few small examples to build intuition.

[3]Note, in class, we showed an interleaved version of this picture, which is technically correct, but a bit confusing to reason about. The benefit of the non-interleaved version is that each bit is conditioned on all of the bits above it in the diagram.



Which let's us reason about the conditional probability of each output in relation to only the values above it. As we have that D and B are independent, $H(A|B) = H(A|B, D)$ and likewise $H(C|D) = H(C|D, B)$, so we can relate the sum of the entropy of the left nodes of the Z with the sum of the (conditional) entropies of the right nodes. We do rely on the fact that at every node, we are polarizing on entropically identical bits. We'll see a more complete proof of the entropy polarization in the next lecture, so for now, just state informally that the conditional entropies are getting polarized, being driven to 0 or 1.

Partitioning P into H and C

Now, we'll complete our construction of H by reasoning about the number of entries in P which have very high or very low entropy.

Consider our inputs: $x_1...x_n$ mapped to $y_1...y_n$ where $y_1...y_n = xP$. Take $\eta_i \triangleq H(y_i|y_1....y_{i-1})$

Claim: As $N \to \infty$, where $N = 2^\ell$, $\#\{i|\eta_i \in (\frac{1}{N^2}, 1 - \frac{1}{N^2})\} = o(N)$. That is, as N tends to infinity, the the entropy of each bit is polarized to either side of the interval $[\frac{1}{N^2}, 1 - \frac{1}{N^2}]$.

Now, let's define sets of columns in P based on their entropy:

$$A = \{i|\eta_i \geq 1 - 1/N^2\}$$

$$B = \{i|\eta_i \leq 1/N^2\}$$

$$C = \{i|\eta_i \in (1/N^2, 1 - 1/N^2\}$$

A is the set of bits whose entropy is high - they aren't well specified by the bits above them. B is the set of bits whose entropy is low, they are well specified by the bits above them, and C is the set that falls in the interval (not high or low). We claim that $|C| = o(N)$, (proof in next lecture). We know that $|A| \leq H(p)N$, so $|B| \geq (1 - H(p) - o(1))N$.

Completing the Construction

Now, we have all the ingredients we need for the construction of H. First, we'll permute the columns of P so that A is the left most, C is the next and B is the last.[4] We argue that this increases the entropy for the bits in A, as they are conditioned on fewer events, and decreases the entropy for B as they are conditioned on more events. Namely, $H(xB|xA, xC) \leq \frac{1}{N^2}|B|$. This follows from the chain rule and monotonicity under conditioning. **Exercise**: argue this formally given our assumptions.

[4]Note, we do not show explicitly how to identify A, B and C in sub-exponential time, see Tal and Vardy for this.

So now we want to guess xB given xH (which is only xA) so we can invert the matrix. To guess xB given xH, we first define $q_a \triangleq \Pr[xH = a]$ and $aq_{b|a} \triangleq \Pr[xB = b|xH = a]$. Given that $H(xB|xH) = \delta$, by a simple Markov inequality we obtain $\Pr_{xH=a}[H(xB|xH = a) > \sqrt{\delta}] \leq \sqrt{\delta}$. When the entropy is $< \sqrt{\delta}$ we have that

$$\Pr[xB = mode(xB|xH = a)] \geq 1 - \sqrt{\delta}$$

So there exists a function $f^{-1} : \{0,1\}^m \to \{0,1\}^n$ such that with high probability $1 - d\sqrt{\delta}$, $f^{-1}(f(x)) = x$.

Overview of the decoding algorithm We'll discuss the details of the decoding algorithm again next lecture, but the broad strokes are: We're starting with a collection of bits and they were all 1 with probability p and 0 with probability $1 - p$. We made combinations of the bits and produced new variables. We marked the resulting elements as being part of A or C or B. For each bit we compute the probability that it takes 1 or 0 given all of the bits above. Given the fact that the matrix is polarized, we'll get very determined probabilities, and we'll do a deterministic rounding. In the next lecture, we'll show how we can compute the probability for each bit.

1 Bibliographic Notes

There are two core works to consider for Polar Codes. The initial work of Arikan, which lays out the proposal, and a second work of Guruswami and Xia which shows decoding in time poly(n/ϵ).

Citations?

References