1 Hello

- Hi! My name is Ido, and we are going to talk about polar codes
- As you may know, polar codes were invented in 2009 by Erdal Arıkan.
- When presenting them, I'm going to try and strike a good balance between a simple explanation that you can follow, on the one hand, and a presentation that is general enough so as that you will be in a good position to carry out your own research.
- So, when presenting, I'm going to be mixing in results and outlooks from several papers. Whoever is interested, please come and talk to me after the talk, and I'll tell you from where I took what.
- I do have a very important request. We're going to be spending 3 hours together. So, this talk is going to very very boring if you lose me. So, if you do, please don't be shy, and ask me to explain again. Really, don't be shy: if you missed something, chances are you're not the only person. So, ask! OK?

2 Introduction

- Polar codes started life as a family of error correcting codes. This is the way we're going to be thinking about them in this talk, but just so you know: they are much more general than this.
- Now, you might expect that since I'm gong to talk about an error correcting code, I'll start by defining it, say by a generator matrix, or a parity check matrix.
- But if you think about it, what I've implicitly talked about just now is a linear code. Linear codes are fine for a symmetric channel, like a BSC or a BEC. But what if our channel is not symmetric for some reason?
- For example, what if our channel is a Z channel: $0 \to^{1-p} 0$, $0 \to^{p} 1$, $1 \to^{1} 1$. Take, say, $p = 0.1$, just to be concrete. Then, the capacity achieving input distribution is not symmetric:

$$C = \max_{P_X} I(X;Y) \ .$$

Since 1 is not as error prone, we'll have $P_X(0) < 1/2 < P_X(1)$. Not something a linear error correcting code handles well.

- So, we'll have to use something fancier than a linear code.
- Also, you'll need a bit of patience: we'll get an error correcting scheme eventually, but we'll start by defining concepts that will seem a bit abstract at first. You'll have to trust me that everything will be useful in the end.
- Let's begin by defining the *pair* of random variables X and Y: X is a random variable having the input distribution we now fix, and Y is the random variable with a distribution corresponding to the output. So, think of X and Y as one input to the channel, and one corresponding output, respectively.
- So, if $X \sim \text{Ber}(\tau)$ then,

  $P_{X,Y}(X = x, Y = y) = P_X(x) \cdot W(y|x)$, where W is the channel law and

  $$P_X(1) = 1 - P_X(0) = \tau$$

  is our input distribution.

  I've written this in a different color since it is key concept that you should keep in mind, and I want to keep it on the board.

- The previous step is important, so I should emphasize it: we are going to build our polar code for a specific channel *and a specific input distribution to the channel.* You'd usually pick the capacity achieving input distribution to the channel, but you don't have to.
- The rate of our code is going to approach $I(X;Y)$ and the probability of error is going to approach 0.
- Now, let's define the pair of random vectors (X^N, Y^N) as N independent realizations of X and Y. That is, $X^N = (X_1, X_2, \ldots, X_N)$ is input to the channel, and $Y^N = (Y_1, Y_2, \ldots, Y_N)$ is the corresponding output.
- OK, so (X^N, Y^N) is the first — for now abstract — concept that you need to remember. Let's write it here, in a different color.

- (X^N, Y^N)

- Eventually, X^N is going to be the input to the channel — the codeword, and Y^N is going to be the channel output. However, we're not there yet: for now, these are just mathematical definitions.

- For simplicity, I'm going to assume a channel with binary input. So, X is going to be a binary random variable and X^N is going to be a binary vector of length N. (write on board).

- The second concept I need to tell you about is the Arıkan transform. It takes a vector x^N of length N and transforms it into the vector

$$u^N = \mathcal{A}(x^N) \; .$$

- The Arıkan transform is simple to define. However, I don't want to burden you with the details just yet. Here is what I want you to know:

  - The Arıkan transform is one-to-one and onto.
  - Thus, there is also an inverse transform $x^N = \mathcal{A}^{-1}(u^N)$.

- Remember our pair of vectors, X^N and Y^N? Let's define

$U^N = \mathcal{A}(X^N)$.

- Our first result on polar codes is called slow polarization. Here it is

Theorem 1: For every $\epsilon > 0$, we have

$$\lim_{N\to\infty} \frac{\left|\left\{i : H(U_i|Y^N, U^{i-1}) < \epsilon\right\}\right|}{N} = 1 - H(X|Y)$$

and

$$\lim_{N\to\infty} \frac{\left|\left\{i : H(U_i|Y^N, U^{i-1}) > \epsilon\right\}\right|}{N} = H(X|Y) \; .$$

- That's quite a mouth-full. Let's try and understand what I've just written.
  - Imagine that we are on the decoder side. So, we get to see Y^N. So, having, Y^N on the conditioning side makes sense.
  - You usually think of the decoder as trying to figure out the codeword, this is going to be X^N, from the received word, which is going to be Y^N.
  - However, since the polar transform is invertible, we might just as well try to figure out U^N for Y^N. That is, we will guess $\hat{U}^N$, and from this guess that the codeword was $\hat{X}^N = \mathcal{A}^{-1}(\hat{U}^N)$
  - Suppose that our decoder is trying to figure out U_i, and, for some reason that I will justify later, someone tells us the what U^{i-1} was.
  - Then, for N large enough, for a fraction of $1-H(X|Y)$ indices, this is going to be "easy". That is, if ϵ is small and $H(U_i|U^{i-1}, Y^N) < \epsilon$, then the entropy of U_i given the above is very small: the decoder has a very good chance of guessing it correctly.
  - Conversely, if ϵ is very small and $H(U_i|U^{i-1}, Y^N) > 1-\epsilon$, then the decoder has an almost fifty-fifty chance of guessing U_i correctly, given that it has seen Y^N and has been told U^{i-1}. So, in this case, we don't want to risk the decoder making the wrong guess, and we will "help" it. How, we'll see...
- In order to show that the probability of misdecoding goes down to 0 with N, we will need a stronger theorem.

Theorem 2: For every $0 < \beta < 1/2$, we have

$$\lim_{N\to\infty} \frac{\left|\left\{i : H(U_i|Y^N, U^{i-1}) < 2^{-N^\beta}\right\}\right|}{N} = 1 - H(X|Y)$$

and

$$\lim_{N\to\infty} \frac{\left|\left\{i : H(U_i|Y^N, U^{i-1}) > 1 - 2^{-N^\beta}\right\}\right|}{N} = H(X|Y) \, .$$

- The above already gives us a capacity achieving coding scheme for any symmetric channel. Namely, for any channel for which the capacity achieving input distribution is $P_X(0) = P_X(1) = 1/2$.
- Assume that W is such a channel, and take $P_X(0) = P_X(1) = 1/2$.
  - So, all realizations $x^N \in \{0,1\}^N$ of X^N are equally likely.
  - That is, for all $x^N \in \{0,1\}^N$, we have $P(X^N = x^N) = 1/2^N$.
  - That means that for all $u^N \in \{0,1\}^N$, we have $P(U^N = u^N) = 1/2^N$.
  - Fix $\beta < 1/2$. Say, $\beta = 0.4$. Take

$$\begin{aligned} \mathcal{F} &= \left\{ i : \mathrm{P_{err}}(U_i | Y^N, U^{i-1}) \geq 2^{-N^\beta} \right\} \;, \\ \mathcal{F}^c &= \left\{ i : \mathrm{P_{err}}(U_i | Y^N, U^{i-1}) < 2^{-N^\beta} \right\} \;, \quad |\mathcal{F}^c| = k \end{aligned}$$

  - Just to be clear, for a binary random variable A,

$$\begin{aligned} \mathrm{P_{err}}(A|B) = \sum_{(a,b)} & P(A = a, B = b) \\ & \cdot \big([P(A = a | B = b) < P(A = 1 - a | B = b)] \\ & + \frac{1}{2} [P(A = a | B = b) = P(A = 1 - a | B = b)] \big) \end{aligned}$$

  - That is, the probability of misdecoding A by an optimal (ML) decoder seeing B.
  - Theorem 2 continues to hold if $H(\ldots)$ is replaced by $2\mathrm{P_{err}}(\ldots)$.
  - The first set is called "Frozen", since it will be frozen to some known value before transmission. That analogy won't be great when we move to a non-systematic setting, so don't get too attached to it.
  - We are going to transmit k information bits.
  - The rate $R = k/N$ will approach $1 - H(X|Y)$, by Theorem 1 (with $2\mathrm{P_{err}}$ in place of H, and fiddling with β).

- In our case $1 - H(X|Y) = I(X;Y)$, the capacity.
- Let's "build" U^N using our information bits, and then make sure that our U^N indeed has the right distribution.
- We will put k information bits into the U_i for which

$$\mathrm{P}_{\mathrm{err}}(U_i|Y^N, U^{i-1}) < 2^{-N^{\beta}} .$$

- Assumption: the information bits are i.i.d. Ber(1/2). This is a fair assumption: otherwise, we have not fully compressed the source.
- Anyway, we can always make this assumption valid by XORing our input bits with an i.i.d. Ber(1/2) vector, and then undoing this operation in the end.
- In the remaining $N - k$ entries, U_i will contain i.i.d. Ber(1/2) bits, chosen in advance and known to both the encoder and the decoder.
- The resulting vector U^N has the correct probability distribution: all values are equally likely.
- Since we've built U^N, we've also built $X^N = \mathcal{A}^{-1}(U^N)$. That is what the encoder transmits over the channel.
- Now, let's talk about decoding.
- Let u^N, x^N, y^N denote the realizations of U^N, X^N, Y^N.
- The decoder sees y^N, and has to guess u^N. Denote that guess by $\hat{u}^N$.
- We will decode sequentially, first $\hat{u}_1$, then guess $\hat{u}_2, \ldots$, and finally $\hat{u}_N$.
- At stage i, when decoding $\hat{u}_i$, there are two possibilities:
  - $\hat{u}_i$ does not contain one of the k information bits.
  - $\hat{u}_i$ contains one of the k information bits.
- The first case is easy: everybody, including the decoder, knows the value of u_i. So, simply set $\hat{u}_i = u_i$.
- In the second case, we set

$$\hat{u}_i = D_i(y^N, \hat{u}^{i-1}) = \begin{cases} 0 & P(U_i = 0|Y^N = y^n, U^{i-1} = \hat{u}^{i-1}) \\ & \quad \geq P(U_i = 1|Y^N = y^n, U^{i-1} = \hat{u}^{i-1}) \\ 1 & \text{otherwise} \end{cases}$$

- Note that when decoding, we are calculating assuming that we have decoded correctly all previous $\hat{u}^{i-1}$. This might not be true, but that is the calculation we carry out nevertheless.
- This decoder is not optimal. That is, it is not ML. However, it can run in $O(N \cdot \log N)$ time. It is also "good enough".
- Let $\hat{U}^N$ be the corresponding random variable.
- Claim: the probability of misdecoding our data is at most
$$P(\hat{U}^N \neq U^N) \leq k \cdot 2^{-N^\beta} \leq N \cdot 2^{-N^\beta} .$$
- Proof, the second inequality is obvious, since $k \leq N$.
- Denote by $\hat{U}^N$ the vector we have decoded. The probability of error is
$$\begin{aligned} P(\hat{U}^N \neq U^N) &= \sum_{i \in \mathcal{F}^c} P(\hat{U}_i \neq U_i, \hat{U}^{i-1} = U^{i-1}) \\ &= \sum_{i \in \mathcal{F}^c} P(D_i(Y^N, \hat{U}^{i-1}) \neq U_i, \hat{U}^{i-1} = U^{i-1}) \\ &= \sum_{i \in \mathcal{F}^c} P(D_i(Y^N, U^{i-1}) \neq U_i, \hat{U}^{i-1} = U^{i-1}) \\ &\leq \sum_{i \in \mathcal{F}^c} P(D_i(Y^N, U^{i-1}) \neq U_i) \\ &\leq \sum_{i \in \mathcal{F}^c} 2^{-N^\beta} \\ &= k \cdot 2^{-N^\beta} \end{aligned}$$

- Summary: the rate of our scheme approaches $1 - H(X|Y) = C(W)$ and the probability of error is bounded by $N \cdot 2^{-N^\beta}$, which goes down to 0 as N increases.

- What do we do when P_X is not symmetric?
- Now X^N is not uniformly distributed. So, U^N is not uniformly distributed as well.
- Let's do a mental exercise.
  - The encoder will produce u^N successively.
  - It won't actually bother encoding information just yet, this is why it's a mental exercise.
  - It will set $u_1 = 0$ with probability $P(U_1 = 0)$ by flipping a biased coin.
  - Then, it will set $u_2 = 0$ with probability

    $$P(U_2 = 0|U_1 = u_1) .$$

  - Then, it will set $u_3 = 0$ with probability

    $$P(U_3 = 0|U_1 = u_1, U_2 = u_2) .$$

  - Generally, at stage i it will set $u_i = 0$ with probability

    $$P(U_i = 0|U^{i-1} = u^{i-1}) , \quad i = 1, 2, \ldots, N .$$

- Now, suppose that on the decoder side, we are told the value of u_i for all $i \in \mathcal{F}$, and our aim, as before, is to decode correctly the u_i for which $i \in \mathcal{F}$. Exactly the same proof gives us that our probability of misdecoding is at most $k \cdot 2^{-N^\beta}$.
- Of course, this would also hold if $\mathcal{F}^c$ were a subset of what it was defined as above — we would only be making the life of the decoder easier by having it make fewer guesses.
- Now, how do we make the encoder actually encode data?
- To do this, we first specialize Theorem 2 to the case of a stupid channel. Namely, a channel whose output Y is always 0. We use the same input distribution as before. We get something strange, but valid. Specializing Theorem 2 to this case gives.

Theorem 2': For every $0 < \beta < 1/2$, we have

$$\lim_{N\to\infty} \frac{\left|\left\{i : H(U_i|U^{i-1}) < 2^{-N^\beta}\right\}\right|}{N} = 1 - H(X)$$

and

$$\lim_{N\to\infty} \frac{\left|\left\{i : H(U_i|U^{i-1}) > 1 - 2^{-N^\beta}\right\}\right|}{N} = H(X) \ .$$

- So, in that mental exercise, for about $N \cdot H(X)$ indices, the encoder was flipping an "almost fair" coin!
- Let's redefine $\mathcal{F}^c$ to include indices for which the coin flip is "almost fair", and the decoder has an excellent chance of decoding.

$$\mathcal{F}^c = \left\{ i : \mathrm{P}_{\mathrm{err}}(U_i|Y^N, U^{i-1}) < 2^{-N^\beta}, \mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}) > \frac{1}{2} - 2^{-N^\beta} \right\} \ ,$$
$$|\mathcal{F}^c| = k \ .$$

- Since the encoder we will soon define is going to be "cheating" a bit, let's denote the u vector it produces by $\tilde{u}^N$, and the corresponding codeword as $\tilde{x}^N$.
- The encoder will do as before, if $i \in \mathcal{F}$. That is, set $\tilde{u}_i = 0$ with probability $P(U_i = 0|U^{i-1} = \tilde{u}^{i-1})$. Otherwise, for $i \in F^c$, set $\tilde{u}_i$ to an information bit (this is the cheating).
- Denote the corresponding random variable as $\tilde{U}^N$, and the resulting codeword as $\tilde{X}^N$.
- The distribution of $\tilde{U}^N$ is "almost" that of U^N, and the same goes for $\tilde{X}^N$ versus X^N.

- Theorem 3: The total variational distance between U^N and $\tilde{U}^N$ is at most $2 \cdot k \cdot 2^{-N^\beta}$. That is,

  $$\sum_{u^N} |P(U^N = u^N) - P(\tilde{U}^N = u^N)| < 2 \cdot k \cdot 2^{-N^\beta} \; .$$

  The same goes for X^N and $\tilde{X}^N$. That is,

  $$\sum_{x^N} |P(X^N = x^N) - P(\tilde{X}^N = x^N)| < 2 \cdot k \cdot 2^{-N^\beta} \; .$$

- I won't prove the first part (in the appendix). However, if you believe the first part, then the second follows since $\mathcal{A}^{-1}$ is one-to-one.

- We assume that the above coin flips are the result of a pseudo random number generator common to both encoder and decoder. That is, both the encoder and the decoder have access to a common random vector α^N, where each $\alpha_i \in [0, 1]$ was chosen uniformly and randomly.

- At stage i, for $i \in \mathcal{F}$, the encoder sets

  $$\tilde{u}_i = 0 \quad \text{if} \quad \alpha_i \leq P(U_i = 0 | U^{i-1} = \hat{u}^{i-1}) \; .$$

  Otherwise, it sets $\tilde{u}_i = 1$.

- So, if $\hat{u}^{i-1} = \tilde{u}^{i-1}$, the decoder can emulate this. It sets

  $$\hat{u}_i = 0 \quad \text{if} \quad \alpha_i \leq P(U_i = 0 | U^{i-1} = \hat{u}^{i-1})$$

  and $\hat{u}_i = 1$ otherwise.

- That is, we have essentially, "told the decoder" the values of the $\tilde{u}_i$ for $i \in \mathcal{F}$, assuming it has not made a mistake thus far.

- For $i \in \mathcal{F}^c$, the decoder sets

  $$\hat{u}_i = D_i(y^N, \hat{u}^{i-1}) = \begin{cases} 0 & P(U_i = 0 | Y^N = y^n, U^{i-1} = \hat{u}^{i-1}) \\ & \quad \geq P(U_i = 1 | Y^N = y^n, U^{i-1} = \hat{u}^{i-1}) \\ 1 & \text{otherwise} \end{cases}$$

- That is, the exact same rule as before. The encoder assumes that it has decoded correctly up to this point, and calculates according to the "none cheating" distribution of U^N.
- Since $\tilde{U}^N$ and U^N are close, the probability of error is not much affected, asymptotically. That is, we now have a factor of 3 added in.
- Claim: the probability of misdecoding our data is at most

  $$\boxed{P(\hat{U}^N \neq \tilde{U}^N) \leq 3 \cdot k \cdot 2^{-N^\beta} \leq 3 \cdot N \cdot 2^{-N^\beta} .}$$

  (proof is in the appendix).
- How about the rate, $R = k/N$?
  - We have to exclude indices i for which

    $$\mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}) \leq \frac{1}{2} - 2^{-N^\beta}$$

    The fraction of these indices tends to $1 - H(X)$.
  - Of the remaining indices, we further exclude those i for which

    $$\mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}) > \frac{1}{2} - 2^{-N^\beta}$$

    and

    $$\mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}, Y^N) > \frac{1}{2} - 2^{-N^\beta}$$

    The second condition implies the first. The fraction of these tends to $H(X|Y)$.
  - We're not done. We also need to exclude those for which

    $$\mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}) > \frac{1}{2} - 2^{-N^\beta}$$

    and

    $$\mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}, Y^N) \leq \frac{1}{2} - 2^{-N^\beta}$$

    and

    $$\mathrm{P}_{\mathrm{err}}(U_i|U^{i-1}, Y^N) \geq 2^{-N^\beta} .$$

    The fraction of these tends to 0.

  - Thus, we have excluded a fraction of $1 - H(X) - H(X|Y)$, and thus the rate tends to

$$1 - (1 - H(X)) - H(X|Y) - 0 = H(X) - H(X|Y) = I(X;Y) \ .$$

- A few comments.
  - We have talked about a memoryless channel, and defined X^N as N i.i.d. copies of X.
  - However, this memorylessness of the channel and the source didn't seem to play too prominent a part.
  - It is important, as we will shortly see, by going "under the hood".
  - However, polar codes can be generalized to settings in which the source, the channel, or both have memory.
  - Then for a fixed input distribution, we can achieve the information rate,

$$\mathcal{I} = \lim_{N \to \infty} \frac{I(X^N;Y^N)}{N} \ .$$

  - So, we can code for ISI channels, Gilbert-Elliot channels, (d,k)-RLL constrained channel, (d,k)-RLL constrained channels with noise, and the list goes on.
  - In fact, we can even use polar codes to code for the deletion channel (shameless plug for my talk at ISIT).

3 The polar transform

- We now define the polar transform $\mathcal{A} = \mathcal{A}_N$.
- First of all, N will have to be a power of 2, that is $N = 2^n$.
- Here is a polar transform of a vector (X_1, X_2) to a vector (U_1, U_2).
- Draw length 4.
- Draw length 8.

- Generally, if $U_1^N = \mathcal{A}(X_1^N)$ and $V_1^N = \mathcal{A}(X_{N+1}^{2N})$, then $F_1^{2N} = \mathcal{A}(X_1^{2N})$ is defined as follows:

$$\begin{aligned} F_1 &= U_1 \oplus V_1 \quad , \qquad F_2 = V_1 \ , \\ F_3 &= U_2 \oplus V_2 \quad , \qquad F_4 = V_2 \ , \\ &\dots \\ F_{2i-1} &= U_i \oplus V_i \quad , \qquad F_{2i} = V_i \ , \\ &\dots \\ F_{2N-1} &= U_N \oplus V_N \ , \quad F_{2N} = V_N \ , \end{aligned}$$

- $$F_{2i-1} = U_i \oplus V_i \ , \quad F_{2i} = V_i$$

- This recursive structure is what makes all the calculations so efficient.
- Define $R_i = (U^{i-1}, Y^N)$.
- We want to first prove Theorem 1.
- The way we will do this is very interesting. Instead of counting indices having a certain property, we will give each index a probability, $1/2^N$ for each $1 \leq i \leq N$, and then ask what is the probability that an index has a property.
- That is, let $B_1, B_2 \dots, B_n$ be i.i.d. and Ber(1/2).
- Define $i = i(B_1, B_2, \dots, B_n)$ as

$$i(B_1, B_2, \dots, B_n) = 1 + \sum_{i=1}^{n} B_i 2^{n-i} \ .$$

Theorem 1: For every $\epsilon > 0$, we have

$$\lim_{N\to\infty} P(H(U_i|Y^N, U^{i-1}) < \epsilon) = 1 - H(X|Y)$$

and

$$\lim_{N\to\infty} P(H(U_i|Y^N, U^{i-1}) > \epsilon) = H(X|Y) \ .$$

- Proof sketch: Define $H_n = H(U_i|Y^N, U^{i-1})$, for the index $i(B_1, B_2, \ldots, B_n)$.
- Then, H_n is a martingale with respect to $B_1, B_2, \ldots, B_n$. That is,

$$E(H_{n+1}|B_1, B_2, \ldots, B_n) = H_n \ .$$

Indeed, for $i = i(B_1, B_2, \ldots, B_n)$,

$$\begin{aligned}
&E(H_{n+1}|B_1, B_2, \ldots, B_n) \\
&= \frac{1}{2}H(F_{2i-1}|F^{2i-2}, Y_1^{2N}) \qquad (B_{n+1} = 0) \\
&\quad + \frac{1}{2}H(F_{2i}|F^{2i-1}, Y_1^{2N}) \qquad (B_{n+1} = 1) \\
&= \frac{1}{2}H(U_i \oplus V_i|U_1 \oplus V_1, V_1, U_2 \oplus V_2, V_2, \ldots, U_{i-1} \oplus V_{i-1}, V_{i-1}, Y_1^{2N}) \\
&\quad + \frac{1}{2}H(V_i|U_1 \oplus V_1, V_1, U_2 \oplus V_2, V_2, \ldots, U_{i-1} \oplus V_{i-1}, V_{i-1}, U_i \oplus V_i, Y_1^{2N}) \\
&= \frac{1}{2}H(U_i \oplus V_i|U^{i-1}, V^{i-1}, Y_1^{2N}) \\
&\quad + \frac{1}{2}H(V_i|U^{i-1}, V^{i-1}, U_i \oplus V_i, Y_1^{2N}) \\
&= \frac{1}{2}H(U_i \oplus V_i, V_i|U^{i-1}, V^{i-1}, Y_1^{2N}) \\
&= \frac{1}{2}H(U_i, V_i|U^{i-1}, V^{i-1}, Y_1^{2N}) \\
&= \frac{1}{2}\left(H(U_i|U^{i-1}, V^{i-1}, Y_1^{2N}) + H(V_i|U^{i-1}, V^{i-1}, U_i, Y_1^{2N})\right) \\
&= \frac{1}{2}\left(H(U_i|U^{i-1}Y_1^N) + H(V_i|V^{i-1}, Y_{N+1}^N)\right) \\
&= \frac{1}{2}\left(H_n + H_n\right) \\
&= H_n
\end{aligned}$$

- Since the Martingale is bounded, it converges almost surely and in L_1. Namely, there exists a random variable H_∞ such that

$$P(\lim_{n\to\infty} H_n = H_\infty) = 1$$

and

$$\lim_{n\to\infty} E(|H_n - H_\infty|) = 0\ .$$

We need to show that $H_\infty \in \{0,1\}$ with probability 1.

- Assume to the contrary that this is not the case.
- The heart of the argument is that to show that if $\epsilon \le H_n \le 1-\epsilon$, then $|H_n - H_{n+1}| > \delta(\epsilon) > 0$. Thus, we cannot have convergence in L_1, since

  $$E(|H_n - H_{n+1}|) = E(|H_n - H_\infty + H_\infty - H_{n+1}|) \le E(|H_n - H_\infty|) + E(|H_{n+1} - H_\infty|)$$

- The LHS is assumed to be bounded away from 0, even in the limit, while the RHS converges to 0.
- The proof of that $\epsilon \le H_n \le 1-\epsilon$ implies $|H_n - H_{n+1}| > \delta(\epsilon) > 0$ follows by Mrs. Gerber's lemma.
- A funny name for a useful result. Essentially, the smallest difference $|H_n - H_{n+1}|$ occures when we are in fact dealing with a BSC.

Theorem 2: For every $0 < \beta < 1/2$, we have

$$\lim_{N\to\infty} P(H(U_i|Y^N, U^{i-1}) < 2^{-N^\beta}) = 1 - H(X|Y)$$

and

$$\lim_{N\to\infty} P(H(U_i|Y^N, U^{i-1}) > 1 - 2^{-N^\beta}) = H(X|Y)\ .$$

- Here, we take an indirect approach. Instead of tracking H_n, we track two new random variables, Z_n and K_n.
- For two random variables, A and B, where A is binary we define

  $$Z(A|B) = 2\sum_b \sqrt{P(A=0, B=b)\cdot P(A=1, B=b)}$$

and

$$K(A|B) = \sum_b |P(A=0, B=b) - P(A=1, B=b)|$$

- Define, for $i = i(B_1, B_2, \ldots, B_n)$,

$$Z_n = Z(U_i | U^{i-1}, Y^N)$$

and

$$K_n = K(U_i | U^{i-1}, Y^N) \ .$$

- It turns out that

$$Z_{n+1} \leq \begin{cases} 2Z_n & \text{if } B_{n+1} = 0 \\ Z_n^2 & \text{if } B_{n+1} = 1 \end{cases}$$

and

$$K_{n+1} \leq \begin{cases} K_n^2 & \text{if } B_{n+1} = 0 \\ 2K_n & \text{if } B_{n+1} = 1 \end{cases}$$

- For $Z_0 << 1$, the squaring operation is much more dramatic that multiplying by 2.
- Assume for a moment that the multiplier 2 was changed to 1. Then, Z_0 would be squared roughly $n/2$ times and K_0 would be squared roughly half of $n/2$ times.
- That is, if $Z_n \approx (Z_0)^{2^{n/2}} = (Z_0)^{\sqrt{N}}$, and the same for K_0.
- This is where the $\beta < 1/2$ comes from.
- We prove that a fraction $H(X)$ of the Z_n converge to 0 very fast, and a fraction $1 - H(X)$ converge to 0 very fast.
- These give us Theorem 2.

Proof of Theorem 3

- Set

$$\alpha_i = P(U_i = u_i | U^{i-1} = u^{i-1})$$

and

$$\beta_i = P(\tilde{U}_i = u_i | \tilde{U}^{i-1} = u^{i-1}) \ .$$

- Then, we are looking at

$$\sum_{u^N}\left|\prod_{i=1}^{N}\alpha_i-\prod_{i=1}^{N}\beta_i\right| .$$

- Write this as

$$\sum_{u^N}\left|\sum_{i=1}^{N}(\alpha_1\alpha_2\cdots\alpha_{i-1}\alpha_i\beta_{i+1}\beta_{i+2}\cdots\beta_N)-(\alpha_1\alpha_2\cdots\alpha_{i-1}\beta_i\beta_{i+1}\beta_{i+2}\cdots\beta_N)\right| .$$

- Now, note that

$$\begin{aligned}
&\sum_{u^N}\left|\sum_{i=1}^{N}(\alpha_1\alpha_2\cdots\alpha_{i-1}\alpha_i\beta_{i+1}\beta_{i+2}\cdots\beta_N)-(\alpha_1\alpha_2\cdots\alpha_{i-1}\beta_i\beta_{i+1}\beta_{i+2}\cdots\beta_N)\right|\\
&=\sum_{u^N}\left|\sum_{i=1}^{N}(\alpha_1\alpha_2\cdots\alpha_{i-1})\cdot(\alpha_i-\beta_i)\cdot(\beta_{i+1}\beta_{i+2}\cdots\beta_N)\right|\\
&\le\sum_{u^N}\sum_{i=1}^{N}|(\alpha_1\alpha_2\cdots\alpha_{i-1})\cdot(\alpha_i-\beta_i)\cdot(\beta_{i+1}\beta_{i+2}\cdots\beta_N)|\\
&=\sum_{u^N}\sum_{i=1}^{N}(\alpha_1\alpha_2\cdots\alpha_{i-1})\cdot|(\alpha_i-\beta_i)|\cdot(\beta_{i+1}\beta_{i+2}\cdots\beta_N)\\
&=\sum_{i=1}^{N}\sum_{u^N}(\alpha_1\alpha_2\cdots\alpha_{i-1})\cdot|(\alpha_i-\beta_i)|\cdot(\beta_{i+1}\beta_{i+2}\cdots\beta_N)\\
&=\sum_{i=1}^{N}\sum_{u_1^{i-1}}(\alpha_1\alpha_2\cdots\alpha_{i-1})\cdot\sum_{u_i}|(\alpha_i-\beta_i)|\cdot\sum_{u_{i+1}^N}(\beta_{i+1}\beta_{i+2}\cdots\beta_N)\\
&=\sum_{i=1}^{N}\sum_{u_1^{i-1}}(\alpha_1\alpha_2\cdots\alpha_{i-1})\cdot\sum_{u_i}|(\alpha_i-\beta_i)|
\end{aligned}$$

- Two cases.
  - If $i\in\mathcal{F}$, then $\alpha_i=\beta_i$, for both $u_i=0$ and $u_i=1$.

  - If $i \in \mathcal{F}^c$, then $\beta_i = 1/2$, for both $u_i = 0$ and $u_i = 1$. By the definition of $\mathcal{F}^c$, we also have $|\alpha_i - \beta_i| < 2^{-N^\beta}$, for both $u_i = 0$ and $u_i = 1$. Thus, in either case,
$$\sum_{u_i} |(\alpha_i - \beta_i)| < 2 \cdot 2^{-N^\beta} \; .$$

- We continue the above chain of inequalities to get
$$\begin{aligned}
&\sum_{i=1}^{N} \sum_{u_1^{i-1}} (\alpha_1 \alpha_2 \cdots \alpha_{i-1}) \cdot \sum_{u_i} |(\alpha_i - \beta_i)| \\
&= \sum_{i \in \mathcal{F}^c} \sum_{u_1^{i-1}} (\alpha_1 \alpha_2 \cdots \alpha_{i-1}) \cdot \sum_{u_i} |(\alpha_i - \beta_i)| \\
&< \sum_{i \in \mathcal{F}^c} \sum_{u_1^{i-1}} (\alpha_1 \alpha_2 \cdots \alpha_{i-1}) \cdot 2 \cdot 2^{-N^\beta} \\
&= \sum_{i \in \mathcal{F}^c} 2 \cdot 2^{-N^\beta} \\
&= 2 \cdot k \cdot 2^{-N^\beta}
\end{aligned}$$

- Note that in the above, the inequality is due to the assumption that $k > 0$. If $k = 0$, the claim trivially holds as well (all terms $|\alpha_i - \beta_i|$ equal 0).

Proof of claim on probability of error

- Denote by $\epsilon(u^n)$ the probability of our decoder erring when the input to the channel is $\mathcal{A}^{-1}(u^N)$.
- We have previously established that the probability of decoding error corresponding to the "non-cheating" encoder is
$$\sum_{u^N} P(U^N = u^N) \cdot \epsilon(u^N) < k \cdot 2^{-N^\beta} \; .$$
- The probability of error corresponding to the "cheating" encoder is thus
$$\sum_{u^N} P(\tilde{U}^N = u^N) \cdot \epsilon(u^N) \; .$$

- The difference of the above two probabilities is bounded as

$$
\begin{aligned}
&\left|\sum_{u^N} \Big(P(U^N = u^N) - P(\tilde{U}^N = u^N)\Big) \cdot \epsilon(u^N)\right| \\
&\leq \sum_{u^N} \Big|P(U^N = u^N) - P(\tilde{U}^N = u^N)\Big| \cdot \epsilon(u^N) \\
&\leq \sum_{u^N} \Big|P(U^N = u^N) - P(\tilde{U}^N = u^N)\Big| \\
&< 2 \cdot k \cdot 2^{-N^\beta}\ .
\end{aligned}
$$

The result follows.

References

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[2] E. Arıkan and E. Telatar, "On the rate of channel polarization," in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2009)*, Seoul, South Korea, 2009, pp. 1493–1495.

[3] E. Arıkan, "Source polarization," in *Proc. IEEE Int'l Symp. Inform. Theory (ISIT'2010)*, Austin, Texas, 2010, pp. 899–903.

[4] S. B. Korada and R. Urbanke, "Polar codes are optimal for lossy source coding," *IEEE Trans. Inform. Theory*, vol. 56, no. 4, pp. 1751–1768, April 2010.

[5] J. Honda and H. Yamamoto, "Polar coding without alphabet extension for asymmetric channels," *IEEE Trans. Inform. Theory*, vol. 59, no. 12, pp. 7829–7838, December 2012.

[6] I. Tal, "A simple proof of fast polarization," *IEEE Trans. Inform. Theory*, vol. 63, no. 12, pp. 7617–7619, December 2017.

[7] R. Wang, J. Honda, H. Yamamoto, R. Liu, and Y. Hou, "Construction of polar codes for channels with memory," in *Proc. IEEE Inform. Theory Workshop (ITW'2015)*, Jeju Island, Korea, 2015, pp. 187–191.

[8] E. Şaşoğlu and I. Tal, "Polar coding for processes with memory," *IEEE Trans. Inform. Theory*, vol. 65, no. 4, pp. 1994–2003, April 2019.

[9] B. Shuval and I. Tal, "Fast polarization for processes with memory," *IEEE Trans. Inform. Theory*, vol. 65, no. 4, pp. 2004–2020, April 2019.

[10] I. Tal, H. D. Pfister, A. Fazeli, and A. Vardy, "Polar codes for the deletion channel: weak and strong polarization," *To be presented in ISIT'19.*