Lecture 13 — February 23

Lecturer: David Tse *Scribe: David L, Tong M, Vivek B*

13.1 Outline

- Polar Codes

13.1.1 Reading

- CT: 8.1, 8.3 – 8.6, 9.1, 9.2

13.2 Recap - Polar Coding Introduction

Last time, we modified the repetition coding scheme to obtain a capacity-achieving coding scheme as shown in Figure 13.1. The modified coding scheme takes in a two bit message U, V, and transmits $X_1 = U$ and $X_2 = U \oplus V$ in two uses the channel P.

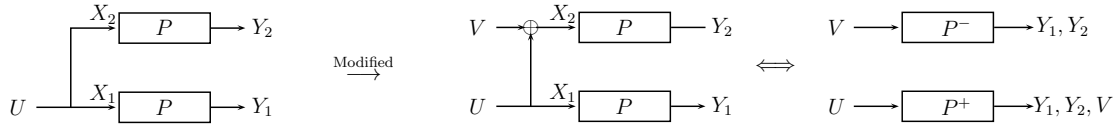


Figure 13.1: The coding scheme in the left is a plain-vanilla repetition code for two uses of the channel. It is modified (right) by adding an extra bit V to the second use of the channel.

Equivalence: As shown in Figure 13.1, the modified coding scheme is **equivalent** to transmitting U via channel P^+, and transmitting V via channel P^-, where P^+ has higher capacity than P, and P^- is lower capacity than P.

Even though the underlying physical channels do not transform/split, for ease of exposition, we shall loosely use phrases like - *Channel P* ***splits*** *into channels P^+ and P^-, $P, P \Leftrightarrow P^+, P^-$* to refer to the above equivalence, and we shall also refer P^+, P^- as the *bit* channels.

13.2.1 Example: BEC(p)

In the previous lecture, we explicitly characterize the bit channels P^+ and P^- for $P =$ BEC(p) (Refer to Figure 13.2). In particular, we showed that $C(P^+) + C(P^-) = 2C(P)$.

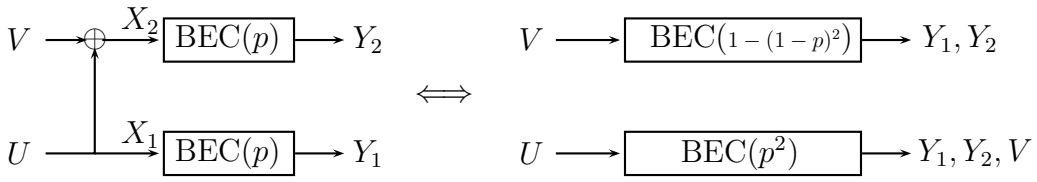


Figure 13.2: $P^+ = \text{BEC}(p^2)$ and $P^- = \text{BEC}(1-(1-p)^2)$.

In the next section, we will give the idea behind polar codes and show that they achieve capacity.

13.3 Idea

Note: some of the figures in this section are reproduced from Erdal Arikan's slides[1]

Without loss of generality, we will restrict ourselves to symmetric channels P whose capacity is $0 \leq C(P) \leq 1$. We know that channel coding is trivial for two types of channels:

1. *Noiseless* channel - Channel with no noise i.e, $C(P) = 1$.
2. *Useless* channel - Channel with *zero* capacity i.e, $C(P) = 0$.

The **main idea** behind polar codes is to transform the message bits in such a way that $C(P)$ fraction of these bits are 'effectively' passed through a *noiseless channel*, whereas the the remaining $(1 - C(P))$ fraction are 'effectively' passed through a *useless channel* (Figure 13.3). Polar codes achieve this by successively applying the transformation $P, P \Leftrightarrow P^+, P^-$ until 'most' of the bit channels are either *noiseless* or *useless*.

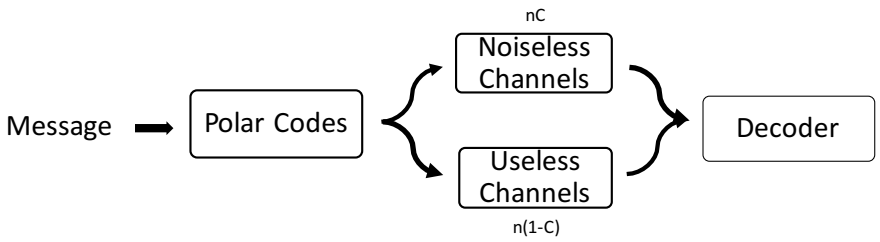
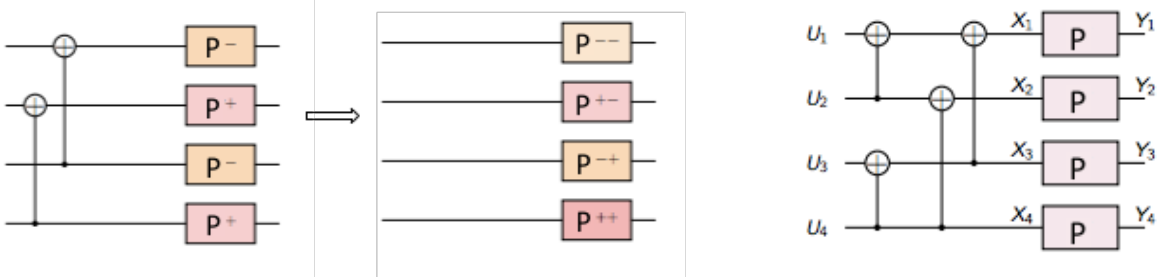


Figure 13.3: Idea behind polar codes.

13.3.1 Successive splitting

Up until now, we've applied transformations which effectively *split* a pair of P channels into P^+ and P^- channels. We now apply the same transformation on pairs of P^+, P^- channels, and *split* them further into P^{++}, P^{+-} and P^{+-}, P^{--} channels respectively. As illustrated in Figure 13.4, this is equivalent of *splitting* four P channels into four channels - P^{++}, P^{+-}, P^{+-} and P^{--}.

[1] *https://simons.berkeley.edu/sites/default/files/docs/2691/slidesarikan.pdf*



(a) (b)

Figure 13.4: (a) *Splitting* pairs of channels P^+, P^- into P^{++} P^{+-} and P^{-+}, P^{--} respectively. (b) Overall transformation from 4 P channels.

Applying the same transformation to the pairs of above four channels will further split them into 8 different channels whose transformation is shown in Figure 13.5

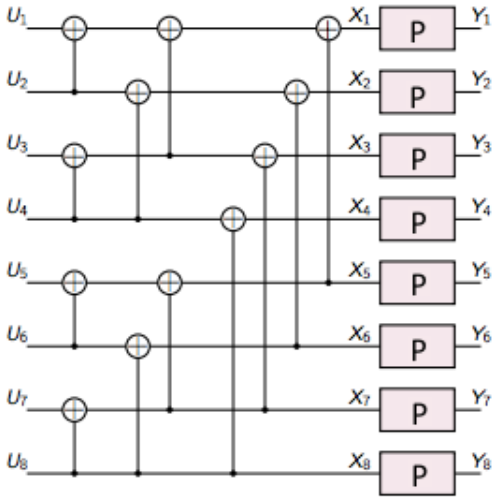


Figure 13.5: Third stage of the transformation with 8 P channels, transformed bits U, codeword bits X, and output Y.

As we keep on splitting further, the bit channels denoted by $P_i^{(n)}$ converge to either a *noiseless* or a *useless* channel. This statement is made precise in the following theorem:

Theorem 1. As the number of channels grow, capacity of the 'bit' channels converge to either 0 or 1, and the fraction is equal to

$$\begin{aligned} \frac{\left|\{i:\ C(P_i^{(n)}) > 1-\delta\}\right|}{n} &\xrightarrow{n\to\infty} C(P) \quad \forall \delta > 0 \\ \frac{\left|\{i:\ C(P_i^{(n)}) < \delta\}\right|}{n} &\xrightarrow{n\to\infty} 1 - C(P) \quad \forall \delta > 0, \end{aligned} \tag{13.1}$$

where n is the total number of 'original' P channels.

Proof. Out of the scope of this class. Interested reader can refer the original paper [Arikan]. □

Corollary 1. The above equations (13.1) directly implies that capacities of all the $P_i^{(n)}$ channels gravitate to either 0 or 1 i.e,

$$\frac{\left|\left\{i:\ \delta < C(P_i^{(n)}) < 1-\delta\right\}\right|}{n} \xrightarrow{n\to\infty} 0.$$

This phenomenon of *polarization* of capacities is illustrated in Figures 13.6 and 13.7.

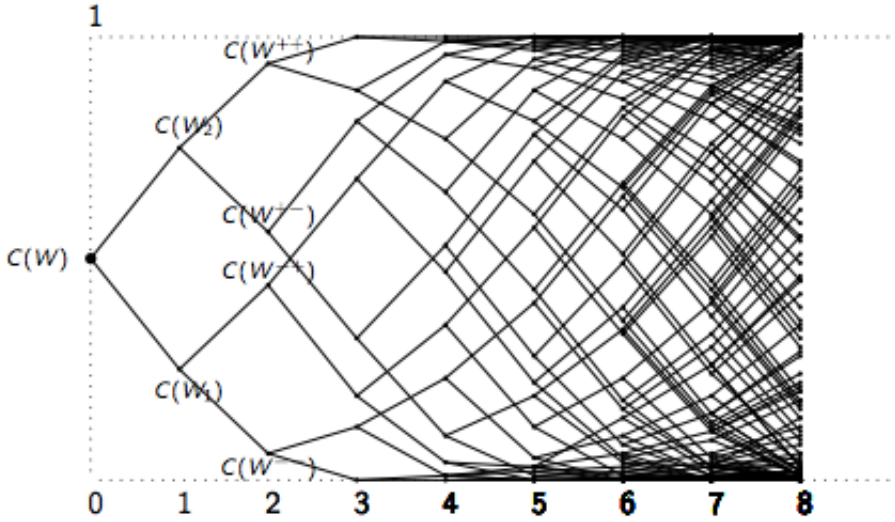


Figure 13.6: Plotting capacity of the bit channels as we apply the polar code transformation in successive stages. Note that there are $n = 2^k$ bit channels at the k^{th} stage. In this figure the channels are denoted by W instead of P.

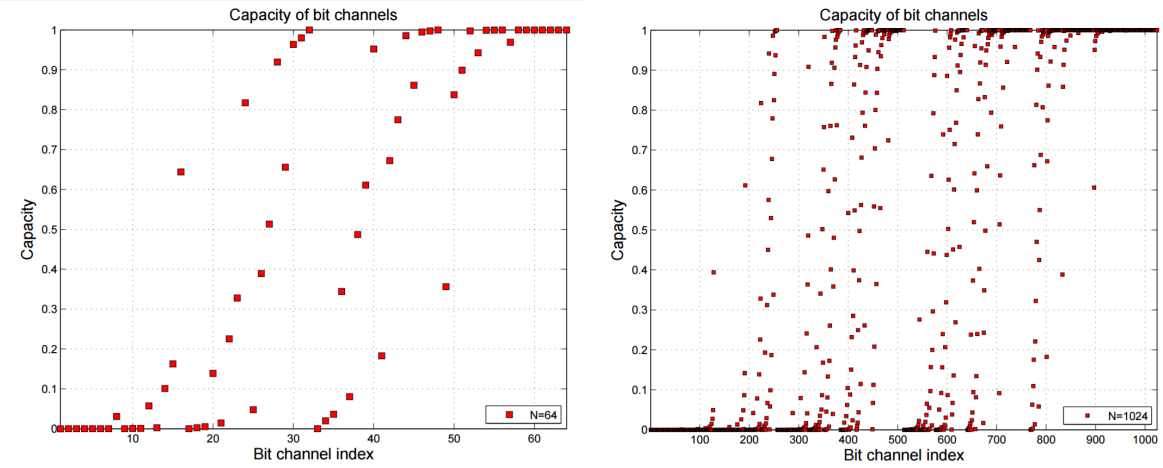


Figure 13.7: Plotting capacity of the bit channels for a BEC(1/2) with $n = 64$ (left) and $n = 1024$ (right). Observe that capacities concentrate near 0 or 1 as n increases.

13.3.2 Why study polar codes?

In the last few lectures, we showed that random codes achieve capacity but their encoding and decoding is inefficient because they do not have a 'nice structure'. Keeping this in mind, we added a structure into the coding scheme by restricting ourselves to random *linear* codes, and this change made the encoding efficient while achieving capacity at the same time. However, this structure is destroyed by the channel noise, and makes decoding inefficient for general linear codes.

Polar codes overcome this problem by *splitting* the channels into *noiseless* and *useless* channels. The *noiseless* channels **preserve** the encoding structure and therefore, for the information passed only over these *noiseless* channels, decoding can be done efficiently.

Therefore, the next logical step is to obtain an encoding scheme to transmit maximum information over the *noiseless* channels. For example, polar codes split 512 BEC(1/2) channels into 256 channels with $C \approx 1$ and 256 channels with $C \approx 0$, and our goal here is to transmit information only using channels with $C \approx 1$.

13.4 Encoding

13.4.1 Linear representation of Polar codes

2^{nd} stage - 4 channels splitting

Let us first consider the case of 4 channels. From Figure 13.8 it is easy to see that codeword bits for message bits U_1, U_2, U_3 and U_4 are

1. $X_1 = U_1 \oplus U_2 \oplus U_3 \oplus U_4$
2. $X_2 = U_2 \oplus U_4$
3. $X_3 = U_3 \oplus U_4$
4. $X_4 = U_4$.

We can express this in a linear form $A_4 X = U$,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}.$$

We see that the top left, top right, and bottom right blocks of A_4 are all the equal to $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, and this is a direct consequence of successive splitting [13.3.1] of the channels.

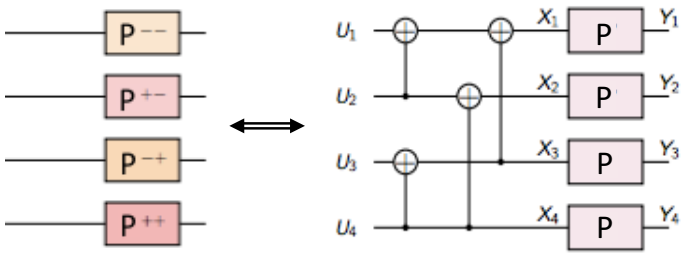


Figure 13.8

3^{rd} stage - 8 channels splitting

Similarly, we can extend the above calculations for eight channels to obtain a linear transformation $A_8U = X$:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_5 \\ U_4 \\ U_6 \\ U_7 \\ U_8 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_5 \\ X_4 \\ X_6 \\ X_7 \\ X_8 \end{bmatrix}$$

which can be verified from Figure 13.9. Here too we have a recursive structure:

$$A_8 = \begin{bmatrix} A_4 & A_4 \\ A_4 & A_4 \end{bmatrix}. \tag{13.2}$$

Up until this point, we have only established equivalence between the *splitting* of the channels and the linear transformations from message bits U's to codeword bits X's. How do we obtain a working coding scheme from this? The answer lies in capacities of the bits channels corresponding to $U_1, U_2 \cdots$

13.4.2 Encoding

For $n = 8$ and $P = \text{BEC}(1/2)$, the effective capacities are listed in Figure 13.9. Here, the encoding scheme sends over the data in the top $nC = 4$ bit channels (U_8, U_7, U_6, U_5), and 'sends' no information (or 0's) in the rest of the bits channels. Note that the order of the bit channel capacities is not monotonic, and in this case we would use the bits U_8, U_7, U_6, and U_4 to send over the data.

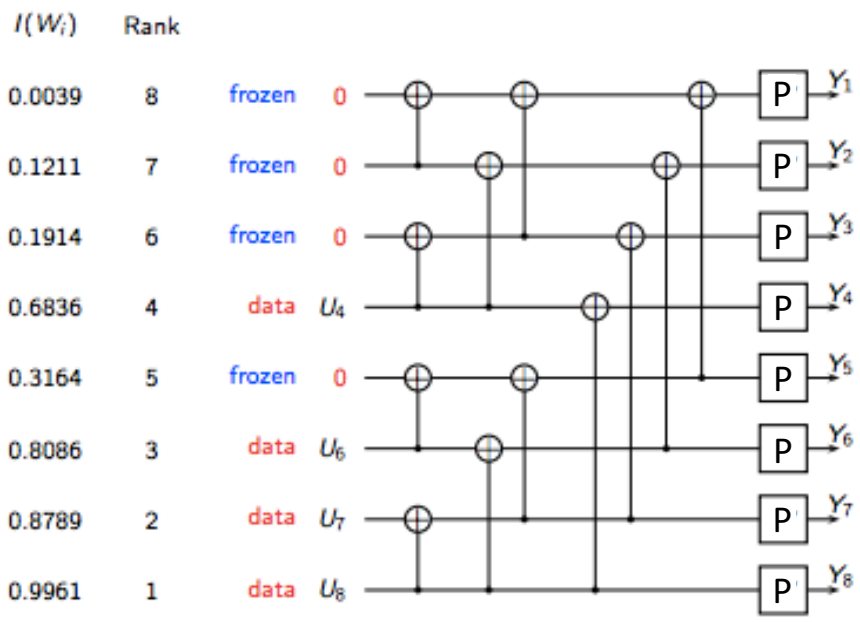


Figure 13.9: Diagram showing the polar code transformation for $P = \text{BEC}(1/2)$, $n = 8$. The bit channel capacities are shown in the '$I(W_i)$' column, and their rank, in order of descending magnitude, is shown in the 'Rank' column. The bits corresponding to high-capacity channels are the ones which should be used to send information (marked 'data'), and no information should be sent using bits corresponding to low-capacity channels (marked 'frozen').

In the above example, the sum of the capacities of the top four bit channels is only 80% of the total capacity and hence, block length of $n = 8$ is not large enough for the asymptotics of equations (13.1) to kick in. Therefore, in practice we use larger block lengths of size $n = 1024$.

Encoding Complexity

From Theorem **??** we know that encoding for polar codes is a linear transformation, and hence it can be achieved in $O(n^2)$ steps. However, in the previous subsection we showed that the generator matrix of polar codes has an interesting 'recursive' structure, and using this, the running time can be improved up to $O(n \log n)$.

13.5 Decoding

For the two channel case, decoding P^+ channel requires the knowledge of Y_1, Y_2, and V. Thus we must decode the P^- channel (which only depends on Y_1 and Y_2) before decoding P^+ to determine V. Notice that the lesser reliable channel P^- should be decoded before decoding P^+. Similarly in the four channel case (see Fig. 13.8) we have to first decode U_1 (least reliable channel). Then decode U_2, and finally decode U_3, then U_4. Alternative argument - the value of U_4 corresponds to the repetition code 'embedded' in X_1, X_2, X_3, and X_4, so it clearly must be decoded last.

Let us go through a detailed example for the $n = 8$ BEC(1/2) channel shown in Fig. 13.9. As per the previous section 13.4.2, we encode the message bits by 'freezing' (send 0) U_1, U_2, U_3, U_5 because these bit channels have the lowest capacity. The message is thus sent only over the U_4, U_6, U_7, U_8 (high capacity bit channels). In the following, we step through the decoding the output $Y_1, \cdots, Y_8$ to obtain back the message.

1. Decode U_1 (Frozen) $\rightarrow U_1 = 0$.
2. Decode U_2 (Frozen) $\rightarrow U_2 = 0$.
3. Decode U_3 (Frozen) $\rightarrow U_3 = 0$.
4. Decode U_4 (Data!) We use $Y_1, Y_2, \ldots, Y_8$, and U_1, U_2, U_3 to decode this signal. Let the decoded bit be denoted by $\hat{U}_4$
5. Decode U_5 (Frozen) $\rightarrow U_5 = 0$.
6. Decode U_6 (Data!) We use $Y_1, Y_2, \ldots, Y_8$, and U_1, U_2, U_3, U_4, U_5 to decode this signal. We know U_1, U_2, U_3, and U_5 since they are frozen, and we will use $(\hat{U}_4)$ to estimate $\hat{U}_5$.
7. Continue in the same fashion for U_7 and U_8.

So in general, we need know 'previous' bits $\hat{U}_1, \ldots, \hat{U}_{i-1}$ to decode U_i. From the above bit-by-bit decoding scheme, we conclude that decoding a message of length n requires $O(nm)$ steps, where m is the time required to decode each bit.

To find the complexity of decoding each bit, we look at the k^{th} splitting stage shown in Figure 13.10:

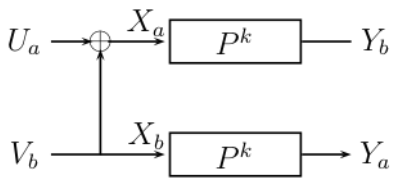


Figure 13.10: : k^{th} *splitting* stage.

Decoding comprises of two steps **(i)** Decoding U_a from Y_a, Y_b, and then **(ii)** decoding U_b from Y_a, Y_b, U_a. This procedure has to be repeated for $k-1$ lower splitting stages. Thus, the complexity of decoding in k^{th} stage = 2×complexity of decoding in $(k-1)^{th}$ stage, which implies m is equal to $2^k = n$. Thus, the total running time of this recursive algorithm is $O(n^2)$. Similar to encoding, the 'recursive' structure of the generator matrix can be exploited to reuse components and therefore reduce the running time from $O(n^2)$ to $O(n \log n)$.

With encoding and decoding running time as efficient as $O(n \log n)$, polar codes can be applied in practice. In fact, polar codes have been incorporated into the latest 5G wireless standard.

Bibliography

[Arikan] Arikan, Erdal. "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels." IEEE Transactions on Information Theory 55.7 (2009): 3051-3073.