Master's Thesis

Internship Report

Machine Learning based channel codes for the Binary Asymmetric Channel

Author:
Kevin Ibarra-Lancheros

Supervisors:
Ms. Meryem Benammar
Mr. Tarik Benaddi

November 2, 2020

Contents

List of Figures

Abbreviations and Notations

AWGN *Additive White Gaussian Noise*

BAC *Binary Asymmetric Channel*

BEP *Bit Error Probability*

BER *Bit Error Rate*

BLC *Block Linear Codes*

BLEP *Block Error Probability*

BLER *Block Error Rate*

BSC *Binary Simmetric Channel*

MAP *Maximum A Posteriori*

ML *Maximum Likelihood*

NN *Neural Network*

Notations

In this document, sets and alphabets are denoted in calligraphic letter, e.g., $\mathcal{X}$. Random variables are denoted by capital case letters, e.g., X and their realizations by lower case letters, e.g., x. the probability mass function of a random binary variable X is denoted $P_X(.)$, while the conditional probability mass function of a random binary variable X knowing Y, is denoted as $P_{X|Y}(.|.)$. An n-dimensional sequence is denoted $x^n = (x_1, x_2, \ldots, x_n)$ where x_i is the i-th component of the sequence. $Bern(\alpha)$ is the Bernoulli distribution with success probability α.

For a matrix $T_l^{\otimes m}$ denotes the l-fold Kronecker product of m copies of T. $\wedge$ and $\oplus$ are the logical operators *and* and *exclusive or*, respectively. The logical *negation* symbol is $^-$, e.g., $\overline{a}$ is the negation of a. A block code with blocklength n and message length k is noted as (n,k). A codebook denoted $C^{(n,k)}$ is the set of 2^k codewords of length n.

Chapter 1

Introduction

The Binary Asymmetric Channel(BAC) is the most general model for binary memoryless channels, it models for instance Free Space Optical (FSO) communications with On-Off Keying(OOK) or molecular communications. This channel model generalizes the Binary Symmetric Channel(BSC), as well as, some completely asymmetric channels such as the Z-Channel and the S-Channel. That is one of the reasons why defining an error correction coding scheme as optimal is hard, in the sense of minimal average error probability for the BAC. In this work, we investigate different manners of designing block error correction codes for short blocklength, in order to find the technique that behaves better over the BAC.

In 1948, Shannon introduces the concept of *channel capacity* as the ultimate rate at which information can be transmitted over a communication channel with a null probability if the blocklength tends to infinity, it is therefore interesting to evaluate how much this theoretical result can be applied in practice. Major families of codes, such as Turbo Codes(Glavieux and Bérou 1990), LDPC codes (Gallager 1960), and very lately, Polar codes (Arikan 2008), were developed throughout the years and proved to be capacity-achieving for classical channels such as the Additive White Gaussian Noise channel (AWGN), the Binary Symmetric Channel(BSC) and the Binary Erasure Channel (BEC). However, for asymmetric channels such as the BAC, and for finite blocklengths, these classical codes are not capacity-achieving.

In recent years, some research groups have invested their efforts in theoretical development of block coding techniques for asymmetric channels, some of them based on totally new approaches, such as [12] and [6], and others based on optimal coding techniques for symmetry channels, such as [11] and [9]. Nevertheless, new techniques have not been sufficiently developed for practical blocklength values and solutions based on known techniques involve extensive processing. In view of this we started with the study of design and implementation of block coding-decoding solutions based on Machine Learning algorithms,

namely, deep neural networks *autoencoders* for the BAC. We would like to respond to the following questions: What performance can reach these neural network-based block codes over the binary asymmetric channel? Can be these solutions considered both as optimal and capacity-achieving end-to-end schemes over the binary asymmetric channel?

The study of neural network-based solutions is interesting because they prove to be promising in the design of optimal channel codes for challenging channels such as the BAC and also because the coding-decoding complexity is made off-line alleviating processing requirements needed to perform these codes. Thereby these solutions could be used by equipment having a reduced processing capacity, such as IoT devices.

The remainder of this report is structured as follows: firstly, some comments about our notation are presented. Then in Section 2 we show the mathematical development used to find an analytical expression for channel capacity, optimal channel decoder and average block error probability for the binary asymmetric channel. In Section 3 we introduce some classical block coding techniques and subsequently a state of the art on optimal code constructions for the BAC ([12],[11]). All these codes are implemented to serve as a benchmark for later comparisons over the BAC. In Section 4 a state of the art on neural-based communication systems is conducted in order to understand the design of both neural network-based channel decoders and overall neural network-based autoencoders for the BAC ([8], [15]). Then, a neural-based channel decoder is developed for the encoder selected as a benchmark and compared with the optimal channel decoder. Later, an overall autoencoder will be developed and compared with channel codes chosen as a benchmark. Finally, Section 5 contains the conclusions of this work about the studied codes for the BAC.

Chapter 2

System model

For this entire project we consider a communication chain (source, channel and destination) as shown in figure 2.1. The binary sequence u^k represents the message the source wants to transmit across the channel, it is composed of k bits and follows an uniform distribution probability. This sequence u^k is the input of the channel *encoder*, which is represented by the function $f : \{0,1\}^k \to \{0,1\}^n$ that assigns a code x^n to every message u^k. Next, we have the channel, given by the conditional probability $P_{Y^n|X^n}(\cdot|\cdot)$ determined by channel conditions (environment, equipment, technology, among other factors). The last element in the communication chain is the decoder represented by the function $g : \{0,1\}^n \to \{0,1\}^k$ that receives the channel output y^n and maps it to an estimation of the original message $\hat{u}^k$. When $\hat{u}^k = u^k$ we communicate successfully.

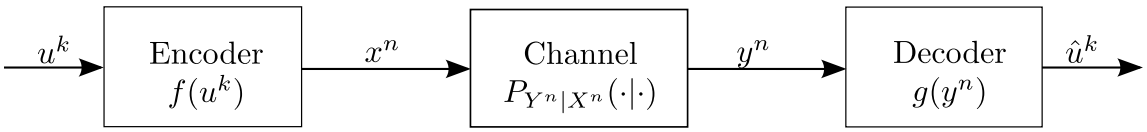


Figure 2.1: Encoding-decoding communication chain

In the following of this chapter we present the model of the channel, the channel capacity and a description of the optimal decoder for the BAC, we also introduce the method used to compare codes.

2.1 Definitions

2.1.1 Discrete Memoryless Channel (DMC)

In [12] is shown that the most fundamental model describing communications over a noisy channel is the so-called discrete memoryless channel, which consist of:

- a finite input alphabet $\mathcal{X}$,
- a finite output alphabet $\mathcal{Y}$, and
- a conditional probability distribution $P_{Y^n|X^n}(\cdot|x^n)$ for all $x^n \in \mathcal{X}$ such that:

$$P_{Y_t^n|X_1^n,X_2^n,...,X_t^n,Y_1^n,Y_2^n,...,Y_{t-1}^n}(y_t^n|x_1^n, x_2^n, ..., x_t^n, y_1^n, y_2^n, ..., y_{t-1}^n) = P_{Y^n|X^n}(y_t^n|x_t^n) \ \forall\, t.$$

Out of this result we notice that they are called memoryless channels because the t-th element of the output y^n depends only on the t-th element of the input x^n, i.e.,

$$P_{Y^n|X^n}(y^n|x^n) = \prod_{i=1}^{n} P_{Y|X}(y_i^n|x_i^n). \tag{2.1}$$

2.1.2 Channel coding problem

A coding scheme consists of:

- the *message set* $\mathcal{U}$ of 2^k equally likely random messages of length k, denoted u^k, thus, the message set length $|\mathcal{U}| = 2^k$,
- an *encoding function* $f : \mathcal{U} \to \mathcal{X}$ that maps every message $u^k \in \mathcal{U}$ into a codeword $x^n \in \mathcal{X}$,
- the *codebook* $\mathcal{C}$ consisting of 2^k codewords of length n, denoted x^n,
- a *decoding function* $g : \mathcal{Y} \to \hat{\mathcal{U}}$ that takes the channel output y^n and assign it an estimate message $\hat{u}^k \in \mathcal{U}$,
- the *estimate message set* $\hat{\mathcal{U}}$ of 2^k estimate messages, it contains the same elements than $\mathcal{U}$,
- the *probability of error* P_e, with regard to blocks is given by $P_e = P(\hat{u}^k \neq u^k) = P(\exists\, i,\ \hat{u}_i^k \neq u_i^k)$ and, concerning bits, it is defined as $P_e^{bit} = \frac{1}{k}\sum_{i=1}^{k} P(\hat{u}_i \neq u_i)$.

Decoding involves the estimation of a sequence $\hat{u}^k$ after observing a channel output y^n, which is correlated with a channel input x^n, which in turn is correlated through a coding rule with the original message u^k, establishing the following Markov chain

$$U^k \longleftrightarrow X^n \longleftrightarrow Y^n \longleftrightarrow \hat{U}^k. \tag{2.2}$$

Lemma 2.1 (Average block error probability P_e for a memoryless channel)
The average block error probability P_e is computed on all possible combinations of U^k, Y^n and $\hat{U}^k$ in which the decision rule $P_{\hat{U}^k|Y^n}$ makes a mistake, it is given by

$$P_e(P_{\hat{U}^k|Y^n}) = \sum_{y^n} P_{Y^n}(y^n) \sum_{u^k} P_{U^k|Y^n}(u^k|y^n) \sum_{\hat{u}^k} P_{\hat{U}^k|Y^n}(\hat{u}^k, y^n)\, \mathcal{I}(\hat{u}^k \neq u^k), \tag{2.3}$$

where $\mathcal{I}(S)$ is the indicator function, defined by

$$\mathcal{I}(S) \triangleq \begin{cases} 1 & \text{if the statement } S \text{ is true} \\ 0 & \text{if the statement } S \text{ is false.} \end{cases}$$

Proof. See appendix A.1 □

Code rate A rather important parameter of the coding scheme is the code rate R, defined as follows

$$R \triangleq \frac{k}{n}\ bits/channel\,use. \tag{2.4}$$

It determines the amount of transmitted information per channel utilisation when a coding scheme is used.

Channel coding theorem

We expect that $\hat{u}^k = u^k$, in order to achieve a successful transmission, in other words, we want an error probability minimal ($P_e \to 0$) using the channel as little as possible, that is, with the coding rate R as big as possible.

With that purpose, in [17], Claude Shannon showed that probability of error P_e tends zero if and only if the code rate R is lower than the channel capacity C.

2.1.3 Channel capacity

For a memoryless channel, the capacity is given by:

$$C = \max_{P_x} I(X;Y), \tag{2.5}$$

where $I(X;Y)$ is the mutual information between X and Y, channel input and output, respectively, mutual information is defined as a function of the entropy $H(\cdot)$ and conditional entropy $H(\cdot|\cdot)$, as follows

$$I(X;Y) = H(X) - H(X|Y), \text{ or} \tag{2.6}$$
$$I(X;Y) = H(Y) - H(Y|X). \tag{2.7}$$

In the following, we start by giving general considerations for the computation of the mutual information, then we solve the optimization problem in (2.5) to state the capacity for the BAC. Proofs are relegated to appendices.

Remark 1
There are two ways to compute the value of the $I(X;Y)$, see (2.6) *and* (2.7)*, they are different because*

- *$H(X)$: Easy to find, it depends only on the input distribution,*
  $H(X|Y)$: Very difficult to find, need to compute the posteriori distribution $P_{X|Y}$ from P_X and $P_{Y|X}$.
- *$H(Y)$: Annoying to find, need to compute the marginal distribution P_Y from P_X and $P_{Y|X}$,*
  $H(Y|X)$: Easy to find, the output distribution given the input is know (the channel model).

Since computing the marginal P_Y is much easier than looking backwards in the channel and computing the conditional $P_{X|Y}$, see [13], we choose to follow the second way to find a expression for mutual information, that is equation (2.7)*, from which we will obtain the channel capacity C defined in* (2.5)*.*

2.1.4 Optimal decoder

In the following we are going to present the decision problem associated to decoding. As described before, the objective is therefore to find the optimal decision rule that minimizes the error probability. For that, let us define a *stochastic decision rule* as a decision made generating random numbers to obtain $\hat{u}^k \in \mathcal{U}$ after observing Y^n and subject to a probability distribution $P_{\hat{U}^k|Y^n}(\hat{u}^k|y^n) = Q_{\hat{U}^k|Y^n}(\hat{u}^k, y^n)$ [14].

Theorem 2.1 (Optimal decoder)
The optimal decision rule for a decoder is a deterministic decision rule defined by

$$Q_{\hat{U}^k|Y^n}(\hat{u}^k, y^n) \triangleq \begin{cases} 1, & \text{if } \hat{u}^k = g_{MAP}(y^n) \\ 0, & \text{if } \hat{u}^k \neq g_{MAP}(y^n), \end{cases} \tag{2.8}$$

where the decision function (Maximum a posteriori) $g_{MAP}(y^n) : \mathcal{Y} \to \hat{\mathcal{U}}$ is given by

$$g_{MAP}(y^n) \triangleq arg \max_{\hat{u}^k} P_{U^k|Y^n}(\hat{u}^k|y^n). \tag{2.9}$$

Proof. See appendix A.2. □

Corollary 2.1 (Maximum likelihood equivalence)
Given the maximum a posteriori rule $g_{MAP}(y^n)$ *in* (2.9) *and the Maximum Likelihood rule* $g_{ML}(y^n)$*, as*

$$g_{ML}(y^n) \triangleq arg\max_{\hat{u}^k} P_{Y^n|U^k}(y^n|\hat{u}^k), \tag{2.10}$$

these rules are equivalents , i.e., $g_{MAP}(y^n) \equiv g_{ML}(y^n)$*, when each element in* $\mathcal{U}$ *is equiprobable.*

Proof. See appendix A.3. □

Implementation Issues

The joint probability could be expressed, as follows

$$P_{U^k,Y^n}(u^k, y^n) = \underbrace{P_{U^k}(u^k)}_{\substack{\text{a priori}\\ \text{distribution}}} \underbrace{P_{Y^n|U^k}(y^n|u^k)}_{\substack{\text{channel}\\ \text{distribution}}} = \underbrace{P_{Y^n}(y^n)}_{\substack{\text{marginal}\\ \text{distribution}}} \underbrace{P_{U^k|Y^n}(u^k|y^n)}_{\substack{\text{a posteriori}\\ \text{distribution}}},$$

since the maximum likelihood rule uses the a priori distribution P_{U^k}, which is known, and the channel distribution $P_{Y^n|U^k}$, which can be characterized, the ML rule is much easier to implement than the MAP. Because, in practical terms, the implementation of MAP rule could become very annoying when k grows, since the marginal distribution of Y^n is described by:

$$P_{Y^n}(y^n) = \sum_{u^k} P_{U^k|Y^n}(u^k|y^n)\, P_{U^k}(u^k),$$

and the cardinal of $\mathcal{U}$ grows exponentially. Therefore, the a posteriori distribution necessary for the MAP decision rule is costly to be calculated in an analytic way. For this reason different algorithms such as Viterbi's aims to compute $P_{U^k|Y^n}$ without marginalizing P_{Y^n}.

2.2 Binary Asymmetric Channel(BAC)

2.2.1 Channel model

In this section we present a special type of binary channel, which is the most general binary discrete memoryless channel, we refer to the *Binary Asymmetric Channel* (BAC). As we can see in the figure 2.2, it consists of a binary input x which has a probability ϵ_0 that a 0 flips into a 1 (i.e., $\epsilon_0 = P_{Y|X}(1|0)$) and a probability ϵ_1 that a 1 becomes a 0 at the output of the channel (i.e., $\epsilon_1 = P_{Y|X}(0|1)$).

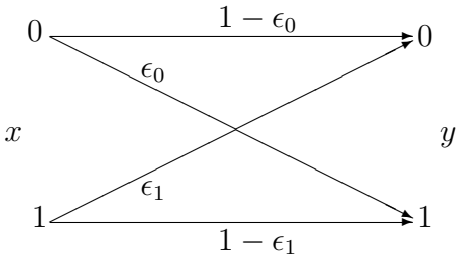
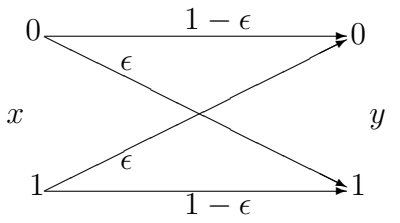
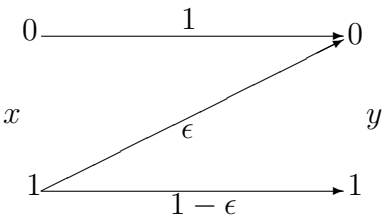


Figure 2.2: The Binary Asymmetric Channel (BAC)

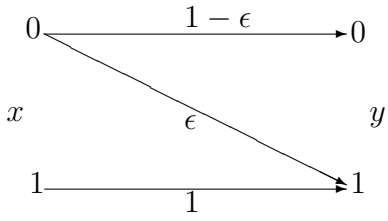
The channel parameters ϵ_0 and ϵ_1 are called the *Crossover Probabilities* and, as any probability, take values in $[0, 1]$. In general, we have the possibility of $\epsilon_0 \neq \epsilon_1$, which explains why this channel is called asymmetric. However, this model also contains well-known binary channel models, such as the *Binary Symmetric Channel* (BSC) when $\epsilon_0 = \epsilon_1$, the *Z-Channel* when $\epsilon_0 = 0$ and the *S-Channel* when $\epsilon_1 = 0$, see Figure 2.3.



(a) The Binary Symmetric Channel (BSC)



(b) The Z-Channel



(c) The S-Channel

Figure 2.3: Well-known binary channel models, which are special cases of the BAC.

The Z-Channel never alters 0 in its input, since $\epsilon_0 = 0$. In contrast, the input 1 is flipped to 0 with a probability ϵ. It also happens in the S-Channel with inverted values .

If $\epsilon_0 + \epsilon_1 = 1$, the channel is completely noisy and the communication is impossible, this statement will be proved later when we present the channel capacity.

2.2.2 Capacity

In order to compute the channel capacity for the BAC is necessary to solve the optimization problem stated in (2.5), we need thereby to find the input distribution $q = P_X(1)$ that maximizes the mutual information.

Lemma 2.2 (Output channel distribution P_Y)
Given an input channel $\mathcal{X} \sim Bern(q)$ and a BAC(ϵ_0,ϵ_1), where ϵ_0 and ϵ_1 are the channel crossover probabilities, then the output channel will follow a Bernoulli distribution with parameter u, i.e.,

$$\mathcal{Y} \sim Bern(u), \text{ where } u \triangleq (1-q)\,\epsilon_0 + (1-\epsilon_1)\,q. \tag{2.11}$$

Proof. The proof of this lemma follows by noticing that

$$\begin{aligned} u = P_Y(1) &= P_X(0)\,\epsilon_0 + P_X(1)(1-\epsilon_1), \\ &= (1-q)\,\epsilon_0 + (1-\epsilon_1)\,q. \end{aligned}$$

□

The result of the lemma 2.2 is used to find a expression for the mutual information of the BAC.

Lemma 2.3 (Mutual information of the BAC)
The mutual information for a BAC(ϵ_0,ϵ_1) with input $X \sim Bern(a)$, is given by:

$$I(X;Y)_{BAC} = h_2\big((1-q)\,\epsilon_0 + (1-\epsilon_1)\,q\big) - (1-q)h_2(\epsilon_0) - q\,h_2(\epsilon_1). \tag{2.12}$$

Proof. The proof of this lemma is detailed in the appendix A.4. □

Then, the expression (2.5) can be rewritten as

$$C_{BAC} = \max_q\, I(X;Y) = h_2\big((1-q^*)\,\epsilon_0 + (1-\epsilon_1)\,q^*\big) - (1-q^*)h_2(\epsilon_0) - q^*\,h_2(\epsilon_1), \tag{2.13}$$

where q^* is the optimal input distribution for $\epsilon_0 + \epsilon_1 < 1$.

Lemma 2.4 (Optimal input distribution)
The expression for the optimal input distribution q^*, for a BAC(ϵ_0,ϵ_1), is given by:

$$q^*(\epsilon_0,\epsilon_1) = P_X^*(1) = \frac{z - (z+1)\,\epsilon_0}{(z+1)(1-\epsilon_0-\epsilon_1)}, \tag{2.14}$$

where,

$$z \triangleq 2^{\frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{(1-\epsilon_1-\epsilon_0)}}. \tag{2.15}$$

Proof. See appendix A.5. □

In lemma 2.4 we point out that the optimal input distribution q^* is a function of the channel crossover probabilities, i.e., $q(\epsilon_0, \epsilon_1)$, this is represented in figure 2.4. We observe for the BSC the optimal distribution is a $Bern(0.5)$, for any value of $\epsilon_0 = \epsilon_1$. However, if we analyse some limit cases, such as $\epsilon_0 \to 0$ and $\epsilon_1 > 0$ (let call them "quasi Z-Channels"), we see that the optimal probability decreases until 0.4 when ϵ_1 grows, it is represented by deep blue in the figure. This makes completely sense, since in a Z-channel should be more convenient to transmit a code with more zeros than ones. On the other hand, for the 'quasi S-Channels', $\epsilon_1 \to 0$ and $\epsilon_0 > 0$, should be advantageous to transmit more ones than zeros, this is proved by the optimal distribution which has values around 0.6, represented by yellow in the figure.

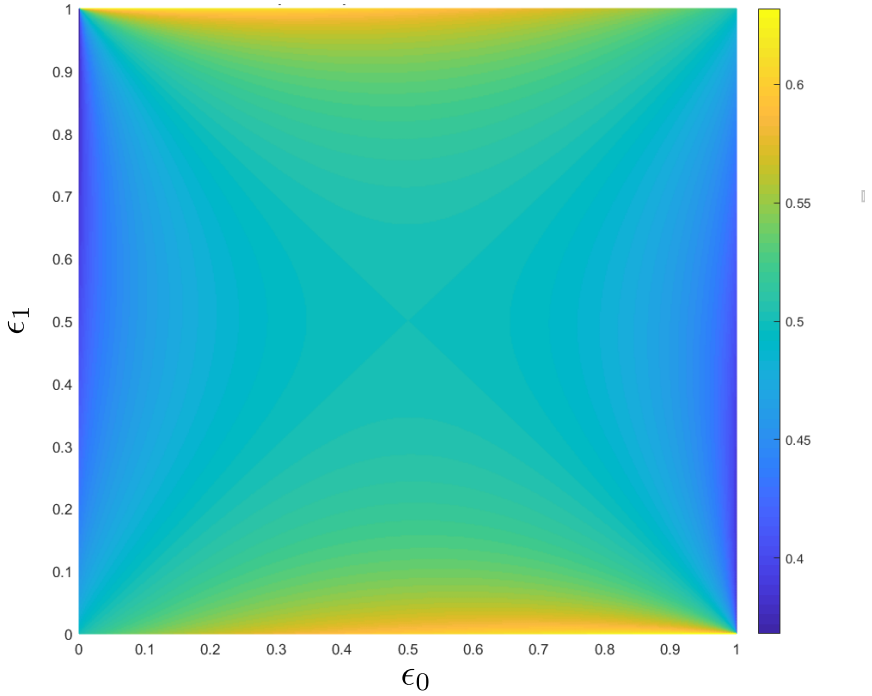


Figure 2.4: Optimal distribution as a function of ϵ_0 and ϵ_1

.

At this point and thanks to the results of lemmas 2.3 and 2.4, we can state the following theorem expressing a formula for the channel capacity for the binary asymmetric channel.

Theorem 2.2 (Binary asymmetric channel capacity)
The capacity for the BAC is given by:

$$C_{BAC} = \log_2\left(2^{\frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{(1-\epsilon_1-\epsilon_0)}} + 1\right) - \frac{(1-\epsilon_1)\, h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_0\, h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1}. \tag{2.16}$$

Proof. See appendix A.6. □

Corollary 2.2 (Symmetry in the capacity)
There are two axes of symmetry for the BAC capacity given by $\epsilon_0 = \epsilon_1$ *and* $\epsilon_1 = 1 - \epsilon_0$ *since*

$$C_{BAC}(\epsilon_0, \epsilon_1) = C_{BAC}(\epsilon_0, \epsilon_1) \text{ and} \tag{2.17}$$
$$C_{BAC}(\epsilon_0, \epsilon_1) = C_{BAC}(1 - \epsilon_0, \epsilon_1). \tag{2.18}$$

Proof. See appendix A.7. □

Figure 2.5 depicts the channel capacity as a function of ϵ_0 and ϵ_1, result of the theorem 2.2. Here, we notice the capacity is maximal, equal to 1 bit/s/Hz, when crossover probabilities are both equal to 0 or both equal to 1, in the latter case all bits would be flipped then we can switch them at detection. On the other hand, capacity is minimal, equal to zero, when $\epsilon_0 + \epsilon_1 = 1$, proving what we mentioned earlier.

Thanks to the results of the corollary 2.2 about the axes of symmetry in the capacity creating four regions, as drawn in figure 2.5. We can therefore restrict the values of channel parameters ϵ_0 and ϵ_1 without loss of generality, as follows:

$$\begin{aligned} 0 \leq \epsilon_1 \leq \epsilon_0 \leq 1 \\ \epsilon_1 \leq 1 - \epsilon_0. \end{aligned} \tag{2.19}$$

We have chosen the highlighted region, because it contains the S-Channel. However, we could have chosen any of them, without loss of generality.

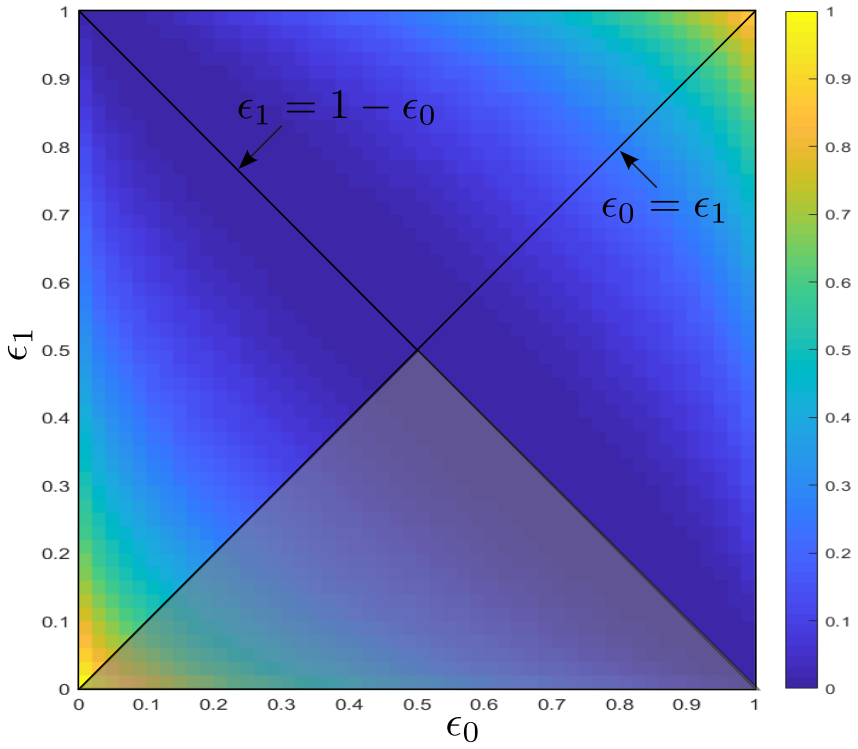


Figure 2.5: Capacity as a function of crossover probabilities (ϵ_0 and ϵ_1)

Finally, we compare channel capacity (with optimal q^*) and mutual information for an uniform distributed Bernoulli input ($q = 0.5$). This comparison is represented in figure 2.6, where $\epsilon_1 \in \{0, 0.1, 0.2, \ldots, 1.0\}$ and $\epsilon_0 \in [0, 1]$. Dotted curves represent channel capacity and continuous lines represent mutual information with an uniform distributed input. We observe a very small gap, actually, it can be only noticed for the case of $\epsilon_1 = 0$ (blue line). This has been mathematically demonstrated in [10], the conclusion is that maximal loss in capacity is around 5.8%, with respect to the uniform distribution. This value could seem modest or even negligible. However, considering bit rates used currently, a small improvement might become significant.

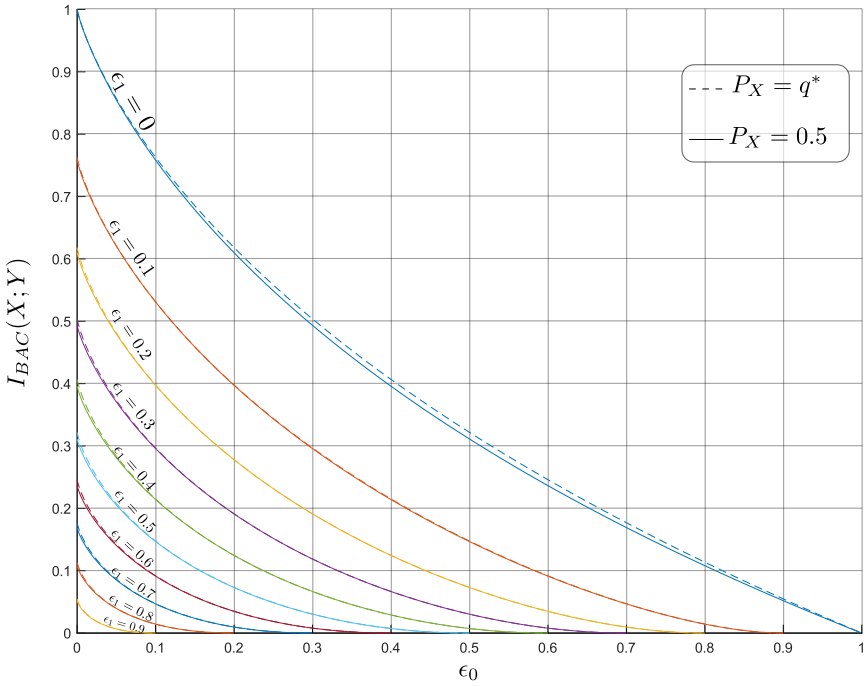


Figure 2.6: Mutual Information for the optimal distribution and uniform distribution BAC

Those results justify the search of an optimal system taking full advantage of the channel capabilities and this is exactly what we attempt by using non-linear coding techniques and later with the machine learning-based solutions.

2.2.3 Channel probability

In the following, we define an expression for the MAP decision rule g_{MAP} according to BAC parameters ϵ_0 and ϵ_1. Furthermore, we describe the average error probability as a function of the crossover probabilities of the BAC. For that, we need a formula for the channel probability $P_{Y^n|U^k}$.

Lemma 2.5 (Channel probability for the BAC)
For a binary asymmetric channel the conditional probability $P_{Y^n|X^n}$ of receiving y^n given the codeword x^n is given by

$$P_{Y^n|U^k}(y^n|u^k) = \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}(y^n,f(u^k))} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}(y^n,f(u^k))} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}(y^n,f(u^k))} (1-\epsilon_0)^n, \tag{2.20}$$

where d_{01}, d_{10} and d_{11} are described in (A.19) *in the appendix A.8.*

Proof. The proof of this lemma is detailed in appendix A.8. □

This result is important for the definition of the MAP function g_{MAP} and the average error probability P_e.

2.2.4 MAP decision function for the BAC

Lets start by the following remark.

Remark 2
Hereafter, the input U^k will be always considered as a uniformly distributed random variable, i.e., $P_{U^k}(u^k) = \frac{1}{2^k}$ (not to be confused with the optimal input channel distribution q, previously studied in last section).

Hence, the maximum likelihood is considered optimal and used in for the theorem state here below.

Theorem 2.3 (MAP decision function)
The function for the MAP over the BAC is given by,

$$g_{MAP}(y^n) = arg\max_{u^k} \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}\left(y^n, f(u^k)\right)} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}\left(y^n, f(u^k)\right)} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}\left(y^n, f(u^k)\right)},$$

where d_{01}, d_{10} and d_{11} are described in (A.19) *in the appendix A.8.*

Proof. The proof of this theorem starts by finding the expression for the maximum likelihood in a binary asymmetric channel, for this, we only need to replace (2.20) in the definition of the ML decision rule (2.10), obtaining that

$$g_{MAP}(y^n) = arg \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}\left(y^n, f(u^k)\right)} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}\left(y^n, f(u^k)\right)} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}\left(y^n, f(u^k)\right)} (1-\epsilon_0)^n,$$

the term $(1-\epsilon_0)^n$ does not depends on u^k. Hence, the expression can be simplified as

$$g_{MAP}(y^n) = arg\max_{u^k} \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}\left(y^n, f(u^k)\right)} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}\left(y^n, f(u^k)\right)} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}\left(y^n, f(u^k)\right)}.$$

□

This function is implemented in Python 3 and used as the decoding algorithm for different linear and non-linear codes proposed in this document.

2.2.5 Average block error probability

Taking the definition of average error probability given in (2.3) and considering a deterministic coding rule $Q_{\hat{U}^k|Y^n}$ as the one given in the theorem 2.1, one can redefine the average error probability as a function of the decoding rule $g(y^n)$ as

$$P_e\big(g(y^n)\big) \triangleq \sum_{y^n}\sum_{u^k} P_{Y^n}(y^n)\, P_{U^k|Y^n}(u^k|y^n)\big(1-\mathcal{I}(g(y^n)=u^k)\big), \tag{2.21}$$

applying Bayes' theorem and considering that $\mathcal{I}\big(g(y^n)\neq u^k\big) = 1-\mathcal{I}\big(g(y^n)=u^k\big)$, the average error probability can be expressed as

$$\begin{aligned} P_e\big(g(y^n)\big) &= \sum_{y^n}\sum_{u^k} P_{U^k}(u^k)\, P_{Y^n|U^k}(y^n|u^k)\,\mathcal{I}\big(g(y^n)\neq u^k\big), \\ &= \sum_{y^n}\sum_{u^k\neq g(y^n)} P_{U^k}(u^k)\, P_{Y^n|U^k}(y^n|u^k) \end{aligned}$$

and knowing that $P_{U^k}(u^k) = \frac{1}{2^k}$ because input messages are equiprobable, it can be written as

$$P_e\big(g(y^n)\big) = \frac{1}{2^k}\sum_{y^n}\sum_{u^k\neq g(y^n)} P_{Y^n|U^k}(y^n|u^k). \tag{2.22}$$

Replacing the channel probability for the BAC $P_{Y^n|X^n}$ defined in (2.20) into the expression (2.22), one obtains a expression for the average error probability for the BAC, given by

$$P_e\big(g(y^n)\big) = \frac{(1-\epsilon_0)^n}{2^k}\sum_{y^n}\sum_{u^k\neq g(y^n)} \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}\left(y^n,f(u^k)\right)} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}\left(y^n,f(u^k)\right)} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}\left(y^n,f(u^k)\right)}. \tag{2.23}$$

Let us introduce the *average success probability* P_s. As its name indicates, it defines the probability of successfully estimate $\hat{u}^k$. This concept is complementary to the average error probability and is defined by

$$P_s\big(g(y^n)\big) = \frac{(1-\epsilon_0)^n}{2^k}\sum_{y^n}\sum_{u^k= g(y^n)} \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}\left(y^n,f(u^k)\right)} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}\left(y^n,f(u^k)\right)} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}\left(y^n,f(u^k)\right)}, \tag{2.24}$$

in practice, we implement P_s instead of P_e because it consumes fewer computational resources and, eventually, we can get both of them since $P_e + P_s = 1$.

Remark 3

So far, the average block error probability P_e has been defined for bit sequences of k bits. Henceforth, the average block error probability will be recalled by its acronym (BLEP).

Similarly, it is possible to define an average bit error probability (BEP) that requires a special development of reasoning and mathematics, which is omitted in this document.

In addition to those definitions, in practical terms, we use the Block Error Rate (BLER) and the Bit Error Rate (BER) as approximations of BLEP and BEP, respectively.

They are given by

$$BLER = \frac{Number\ of\ mis\text{-}decoded\ blocks}{Number\ of\ transmitted\ blocks}$$

and

$$BER = \frac{Number\ of\ mis\text{-}decoded\ bits}{Number\ of\ transmitted\ bits}.$$

All these concepts will help us to measure and compare codes performance in the following.

Chapter 3

Classical methods

In the previous chapter, we showed the expression for channel capacity, from this concept we can extrapolate a maximal transmission rate C below which the communication has an error probability of 0, otherwise the error probability will be equal to 1. However, this is an asymptotic result for codes with blocklength $n \to \infty$. From a practical standpoint, the fact that n is finite has changed the approach. Nowadays, we investigate the maximal channel coding rate R achievable at a given blocklength n, aiming at a desired error probability [16].

In this document, we fix the blocklength n and the coding rate R to explore different coding schemes in order to find the best BER for the binary asymmetric channel. In the following, we present several coding mechanisms with codewords of the same length. This allows us to restrict our attention to block codes, which could be classified in linear and non-linear.

3.1 Linear codes

A linear code is defined as a code in which the sum of any two codewords is a codeword too [12]. In this document, we consider only binary codes, hence the sum can be expressed as an *exclusive or* between codewords. It is important to note that a linear code contains imperatively the all-zero codeword, because $x^n \oplus x^n = 0^n$ for any x^n.

There are two essential parameters for any code, the number of possible messages $|\mathcal{U}|$ and the blocklength n. The larger is $|\mathcal{U}|$, the more information is transmitted and the shorter is n, the less time is needed to transmit a message.

3.1.1 Block linear codes

This is a large family of linear error correcting codes which encodes the input sequences of k bits onto blocks of length n. In block coding, we can regard the encoding simply as a mapping from sequences $u^k \in \mathcal{U}$ to codewords $x^n \in \mathcal{X}$. On the receiver end the decoder takes the noisy codeword y^n and produces a sequence $\hat{u}^k$ [7]. A block error occurs when $\hat{u}^k \neq u^k$. These codes can be expressed in a compact way by defining the *generator matrix* $G^{n,k}$, then considering u^k and x^n as row vectors, we have that

$$x^n = u^k \, G^{n,k}, \tag{3.1}$$

where $u_i^k \, G^{n,k}$ is matrix multiplication using modulo-2 addition.

Then the design of a block linear code is reduced to finding the generator matrix that minimizes the error probability. For short blocklengths, such as considered in this document, there are many different code families, some of them aim to simplify coding or decoding algorithms. However, we will ignore the complexity of decoding, since we compare codes performance by using the optimal decoder for all of them.

In the following, we introduce briefly some of the most known block linear codes over a binary alphabet, and give the constraints on their parameters (n,k).

Hamming codes For any integer $r \geq 2$, we have message length $k = 2r - r - 1$ and block length $n = 2r - 1$. Hence, the rate of Hamming codes is $R = \frac{k}{n} = 1 - \frac{r}{(2r-1)}$. Hamming codes are considered as perfect codes, in the sense that they achieve the highest possible rate for codes with their block length and minimum distance between codewords. In the most basic form, these codes have been designed to correct up to 2 errors in a sequence of k bits, there are some extensions that attempt to correct more errors per sequence [7].

Hadamard codes For a given message length k, this code has block length $n = 2^k$. The block length is very large compared to the message length, but on the other hand, errors can be corrected even in extremely noisy conditions. Hence, the rate of Hadamard codes is $R = \frac{k}{n} = \frac{k}{2^k}$. The augmented Hadamard code improves a bit the rate $R = \frac{k+1}{2^k}$ [7].

Reed-Muller codes For given two parameters r and m, such that $0 \leq r \leq m$, we have message length $k = \sum_{i=0}^{r} \binom{m}{i}$ and block length $n = 2^m$. Hence the rate of Reed-Muller codes is $R = \frac{k}{n} = \frac{k}{2^m}$ [7].

BCH codes For a given positive integer m, we have a primitive binary BCH code with block length $n = 2^m - 1$ and distance at least d, then the message length k depends on the number of errors the code can correct. It is possible to design binary BCH codes that can correct multiple bit errors changing the distance d [7].

Polar codes Consider a polar code with block length n and message length k. We define the information bit indices I as a set of k elements, such that $I_i \in \{0, \ldots, n\}$. The remaining $n-k$ indices are called frozen bit indices defined as $F = \{0, \ldots, n\} \backslash I$. Here, n is a power of 2 and we define $m \triangleq log_2(n)$. For a (n,k) polar code, the generator matrix $G = (T_2^{\otimes n})\, I$. Therefore, given a sequence u^k, a codeword x^n is generated as:

$$x^n = u^k\, G^{n,k} = d^n\, T_2^{\otimes m},$$

where d^n is a vector of n bits including information bits such that $d_I = u^k$ and frozen bits $d_F = 0$. On the other hand, the matrix $T_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ is the Arikan's kernel for polar codes [18].

As shown, for polar codes the block length n is always a power of 2, however, the message length k could take any value lower than n. This makes the polar codes more flexible with respect to the rate $R = \frac{k}{n}$. Additionally, polar codes are also interesting because they have been shown optimal for all binary symmetric memoryless channel, such as the binary symmetric channel and the binary erasure channel, in terms of error probability and optimal input distribution [18].

Comparison of linear codes

Table 3.1 shows some possible combinations of message length and block length for listed codes. In order to compare the performance of two codes, we need to take the same rate and the same block length, otherwise the comparison is not fair, as shown in table 3.1, it is almost impossible for most of them. That is the reason why, in figure 3.1, we compare a Polar code $(8, 4)$ with a BCH code $(7, 4)$.

Table 3.1: Lengths for linear block codes

Code Family	Some code lengths (n, k)
Hamming	$(7, 4)$, $(15, 11)$, $(31, 26)$, $(63, 57)$, ...
Hadamard	$(8, 3)$, $(16, 4)$, $(32, 5)$, $(64, 6)$ ...
Augmented Hadamard	$(4, 3)$, $(8, 4)$, $(16, 5)$, $(32, 6)$ ...
Reed-Muller	$(16, 5)$, $(16, 11)$, $(16, 15)$, $(32, 6)$ ...
BCH	$(7, 4)$, $(9, 3)$, $(15, 11)$, $(15, 7)$ ...
Polar	$(4, k)$, $(8, k)$, $(16, k)$, $(32, k)$ ...

3.1.2 Software to design block linear codes

As shown, the design of a linear code for a desired message length k and block length n could become hard, then in this project we use a software called *MAGMA Computational Algebra*

System, which is distributed by the University of Sydney and developed by members of the mathematical community. MAGMA is a large, well-supported software package designed for computations in algebra and number theory. It contains a large list of useful functions for coding theory, it is available online via http://magma.maths.usyd.edu.au/calc/.

One of the most interesting functions of MAGMA is the *Best Known Linear Code* (BKLC), it incorporates databases[1] containing constructions of the best known linear codes over discrete Galois fields. As summarised in table 3.2, the $binary(GF(2))$ codes database is 100% complete, in the sense that it contains a construction for every set of parameters up to $n = 256$ and for $n < 31$ this codes are known to be optimal. The database for $ternary(GF(2))$ codes is more than 78% complete for every set of parameters up to $n = 243$, for block length up to $n = 21$ the codes are known to be optimal [4].

Table 3.2: MAGMA

	$(GF(2))$	$(GF(3))$
n_{max}	256	243
n_{opt}	31	21
n_{comp}	256	100
total	33152	29889
filled	100%	78.05%

An (n, k) linear code C is said to be a best known linear (n, k) code if C has the highest minimum weight among all known (n, k) linear codes.

Construction of a the Best Known Linear Code(BKLC) with MAGMA It is easy to use this function, we need to indicate the field F, in our case the Galois Field of two elements $GF(2)$, additionally, we need to specify just the block length n and the message length k. The function returns a generator matrix with the specified sizes. Here below, we show an example of how the MAGMA produces the best know linear codes for $n = 16$ and $k = 8$.

The program starts by creating a $(17, 9)$-linear code, then applies over this code an extension and obtain $\langle 1 \rangle$, a cyclic linear code $(18, 9)$ over $GF(2)$. Then, over $\langle 1 \rangle$ is applied a function of puncturing that reduces the size again to $(17, 9)$. Finally, this code $\langle 2 \rangle$ is shorted and we obtain the desired code $(16, 8)$. We present below the source code of this process, the functions of cyclic linear codes, extension, puncturing and shortening are well detailed in [4] the software handbook .

```
>>BKLC(GF(2), 16, 8)
```

[1]The construction of the MAGMA's BKLC database has been undertaken by developers from Sydney University and Karlsruhe University.

```
Construction of a [ 16 , 8 , 5 ] Code:
[1]:  [18, 9, 6] Linear Code over GF(2)
        Extend the QRCode over GF( 2) of length 17
[2]:  [17, 9, 5] Cyclic Linear Code over GF(2)
        Puncturing of [1] at { 18 }
[3]:  [16, 8, 5] Linear Code over GF(2)
        Shortening of [2] at { 17 }
[16, 8, 5] Linear Code over GF(2)
```

3.1.3 Numerical results

In figure 3.1, BER and BLER are used as a measure of the performance for the codes. The scenario is a BAC($\epsilon_0 \in [0,1]$,$\epsilon_1 = 0.001$). We present the average error rate when using three linear coding techniques, namely, polar codes, BKLC and BCH codes. For the first two techniques we have $n = 8$ and $k = 4$, for the third technique is used the most similar rate possible using BCH constructions, that is $n = 7$ and $k = 4$. We can notice the three coding schemes have a very similar performance, both in BER and BLER. Even if the BCH has a marginal gain compared to the others, the fact that $R = k/n$ is restricted to certain combinations, makes us decide to take as reference the polar codes with broader spectrum of possibilities.

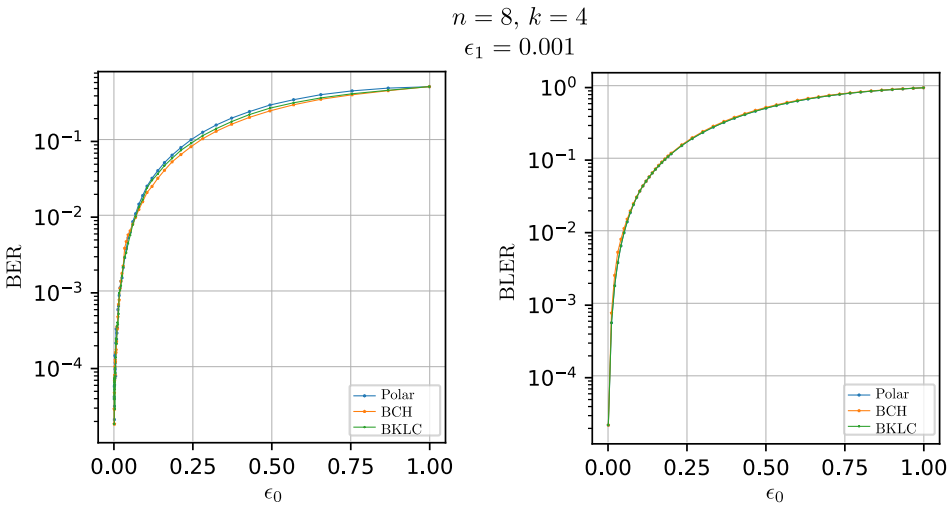


Figure 3.1: BER and BLER for polar code(8,4), BCH code(7,4) and BKLC(8,,) with MAP decoder

Similarly, in figure 3.2, we can see how these codes perform in relation to the average error rate. The scenario is the same, a BAC($\epsilon_0 \in [0,1]$, $\epsilon_1 = 0.001$). Once again the BCH

Code(15,4) achieves the same performance as Polar Code(16,4) and BKLC(16,4) for all values of ϵ_0, both in BER and in BLER, even for a smaller coding rate $R = k/n$.

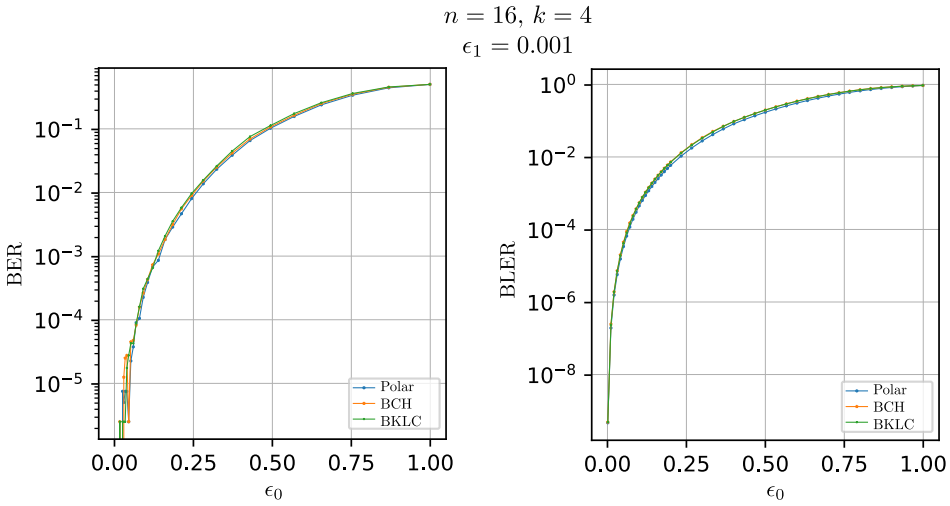


Figure 3.2: BER and BLER for polar code(16,4), BCH code(15,4) and BKLC(16,4) with MAP decoder

So far, we have carried out a survey of the block linear codes for the average error rate in order to have some guideline values for posterior mechanisms. However, according to result of chapter 2, we know that linear codes *do not* have the optimal probability to achieve the theoretical capacity. Next, we will introduce another type of codes that attempt to approach this distribution for the BAC without increasing average error.

3.2 Non-linear block codes

In contrast to the codes listed above, we present some non-linear codes, which means, the sum of any two codewords is not necessarily a codeword. For the remainder of this section, we propose a construction of non-linear block codes, then we introduce two coding techniques that enable reliable transmission reaching the optimal capacity distribution for an arbitrary discrete memoryless channel [11].

3.2.1 Flip codes

Flip codes are fully developed and analyzed in [6], here we reveal some elements that help us to understand their characteristics and some construction techniques.

By default, a flip code is composed by two codewords such that a codeword is the flipped version of the other. The flip code of type t for $t \in \{0, 1, \ldots, \lfloor \frac{n}{2} \rfloor\}$ is defined by the

following codebook matrix:

$$C_t^{(2,n)} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x \\ \bar{x} \end{pmatrix} \triangleq \begin{pmatrix} 0 & \dots & 0 & \overbrace{1 & \dots & 1}^{t\ columns} \\ 1 & \dots & 1 & 0 & \dots & 0 \end{pmatrix}.$$

Defining the column vectors

$$\left\{ c_1^{(2)} \triangleq \begin{bmatrix} 0 \\ 1 \end{bmatrix}, c_2^{(2)} \triangleq \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\},$$

in a flip code type t there are $n-t$ columns $c_1^{(2)}$ and t columns $c_2^{(2)}$.

Knowing that the BAC is a memoryless channel, it can be concluded that the order of the columns of the codebook matrix could be changed, keeping code's performance undisturbed. However, we retain this notation to facilitate their presentation.

There are two important aspects to point out about flip codes. Firstly, considering $t \neq 0$, this codebook does not contain the all zero codeword, then it is non-linear. Secondly, the fact that we describe this codebook as n vectors of length 2 instead of specifying each codeword of the codebook, allow us to consider the codebook *columnwise* instead of *rowwise*, this approach turns out to be more convenient, see [6].

Weak flip codes

The flip codes have a problem, by their definition, they *cannot* be extended to a situation with more than two codewords. Hence, we introduce a new approach based in the same idea, the *weak flip codes* for $|\mathcal{U}| \geq 2$, constructed solely by *weak flip candidates*.

A *weak flip candidate* is a column vector of length $M = 2^k$ with its first component equal to zero, such that it contains the same number of ones and zeros.

For example, if $k = 2$, then $M = 4$, the set of weak flip candidates $\mathcal{C}^{(M)}$ is composed by:

$$\mathcal{C}^{(4)} = \left\{ c_1^{(4)} \triangleq \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, c_2^{(4)} \triangleq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, c_3^{(4)} \triangleq \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \right\}.$$

The weak flip codebook of type (t_2,t_3) of four elements and block length n ($C_{(t_2,t_3)}^{(4,n)}$) is defined by a codebook matrix consisting of t_1 columns $c_1^{(4)}$, then t_2 columns $c_2^{(4)}$, and finally t_3 columns $c_3^{(4)}$, where $t_1 \triangleq n - t_2 - t_3$, this can be expressed as follows:

$$C^{(4,n)}_{(t_2,t_3)} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{bmatrix} \overbrace{c_1^{(4)} \dots c_1^{(4)}}^{t_1 \text{ col.}} & \overbrace{c_2^{(4)} \dots c_2^{(4)}}^{t_2 \text{ col.}} & \overbrace{c_3^{(4)} \dots c_3^{(4)}}^{t_3 \text{ col.}} \end{bmatrix}.$$

We can show that the cardinality of the set of weak flip candidate columns set grows rapidly, since it is given by

$$\left|\mathcal{C}^{(M)}\right| = \binom{2^k - 1}{2^{k-1}},$$

this induces some problems, because the selection among all these options is hard to optimize. This is the reason why, a fair way to choose the candidates has be proposed taking each candidate the same number of times. For $k = 3$ that implies $t_1 = t_2 = t_3$. However, for $k = 4$ that implies very low coding rates, $R = 4/6435$ for instance, making the fair weak flip codes useless from a practical standpoint [5].

Linear extension of flip codes

We propose a new method to construct new codes, based on the idea of flip codes in which we are able to choose the coding rate $R = \frac{k}{n}$.

For $k = 3$, the number of codewords $M = 8$, they are defined as a linear combination of the row vectors x_a^n and x_b^n, defined as

$$x_a^n = (0 \dots\dots\dots 00 \overbrace{1 \dots 11}^{t_a}),$$
$$x_b^n = (0 \dots 00 \overbrace{1 \dots\dots\dots 11}^{t_b}),$$

in addition to $\mathbf{0}$ and $\mathbf{1}$, the *n-zeros* and the *n-ones vectors*, for $n \in \mathbb{Z}$ and $n > k$. Hence, the codebook is defined by

$$C^{(8,n)}_{(t_a,t_b)} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ x_a \\ \overline{x_a} \\ x_b \\ \overline{x_b} \\ x_a \oplus x_b \\ \overline{x_a \oplus x_b} \\ \mathbf{1} \end{pmatrix} \triangleq \begin{bmatrix} 0 \dots \overbrace{\dots\dots\dots}^{t_b} \overbrace{\dots 00}^{t_a} \\ 0 \dots \dots \dots 00\,1 \dots 11 \\ 1 \dots \dots \dots 11\,0 \dots 00 \\ 0 \dots 00\,1 \dots \dots \dots 11 \\ 1 \dots 11\,0 \dots \dots \dots 00 \\ 0 \dots 00\,1 \dots 11\,0 \dots 00 \\ 1 \dots 11\,0 \dots 00\,1 \dots 11 \\ 1 \dots \dots\dots\dots \dots 11 \end{bmatrix}. \tag{3.2}$$

As we can notice, for the design of an extended flip code (n,3) we have just two parameters to optimize[2] t_a and t_b. In general, for the extended flip codes, the number of parameters grows linearly with k being equal to $k-1$.

To find a code for $k=4$ and $n=8$ we have implemented a greedy algorithm to optimize ($t_a,\ t_b,\ t_c$), the results where $t_a=2$, $t_b=4$ and $t_c=6$. The average error rate resultant of using this coding scheme over a BAC($\epsilon_0 \in [0,1]$, $\epsilon_1 = 0.001$) is shown in figure 3.8.

Repetition codes This is a coding technique, which consist of transmitting several copies of each bit to be sent.

In the extended flip codes, we can notice that after optimization, t_a is equal to $\lfloor n/k \rfloor$, $t_b = 2\,t_a$, $t_c = 3\,t_a$ and so forth. Consequently, reorganizing the codewords proposed in (3.2), we have that linear extension of flip codes is equivalent to repetition codes, regular if n is a multiple of k and irregular otherwise.

Here below, we present how the codes in (3.2) should be organized to obtain a repetition code scheme for $k=3$ and $n=9$. On the left-hand side, we have the set U^k that represents all possible messages of length k, each element of U^k is encoded in one codeword of the set X^n, following the order listed below,

$$U^k = \begin{bmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix} \Rightarrow X^n = \begin{pmatrix} \mathbf{0} \\ x_a \\ x_a \oplus x_b \\ x_b \\ \overline{x_b} \\ \overline{x_a \oplus x_b} \\ \overline{x_a} \\ \mathbf{1} \end{pmatrix} = \begin{bmatrix} \overbrace{000\,000\,\overbrace{000}^{t_a}}^{t_b} \\ 000\,000\,111 \\ 000\,111\,000 \\ 000\,111\,111 \\ 111\,000\,000 \\ 111\,000\,111 \\ 111\,111\,000 \\ 111\,111\,111 \end{bmatrix}.$$

In short, linear extended weak flip codes have exactly the same codebook structure than repetition codes. Due to the way these codes are defined, we have a linear coding technique.

3.2.2 Capacity approaching techniques

Now we introduce a new approach consisting of using a linear code, optimal for symmetric scenarios, along with a non-linear technique, in order to create codewords distributed

[2] *Optimal* in the sense of minimal error probability.

according to the optimal input distribution for capacity. This means, in the communication chain presented in figure 3.3, that $x^n = f(u^k)$ respects the optimal input distribution q^*, to that end, the encoder function f will be divided into two blocks, one that uses a linear coding scheme and another, applied before or after the linear encoding, that biases the codewords, in order to achieve the target input distribution.

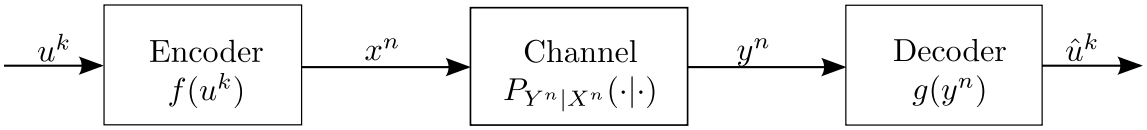


Figure 3.3: Encoding-decoding communication chain

In the following, we will present the Gallager's mapping for linear codes, proposed by Robert Gallager in 1968 [7], and the integrated scheme for polar codes [9].

Gallager's mapping

This solution uses a standard linear code and applies a non-linear mapper to the encoded bits. To that end, we use a linear code with blocklength $n\,t$, where t is the mapping parameter, then, we add a new block to the communication chain, as shown in figure 3.4, the mapper takes as input a block with length $n\,t$ and maps it into a n-length block. The function of this mapping can be chosen arbitrary to have in the output a set of codewords with the correct distribution.

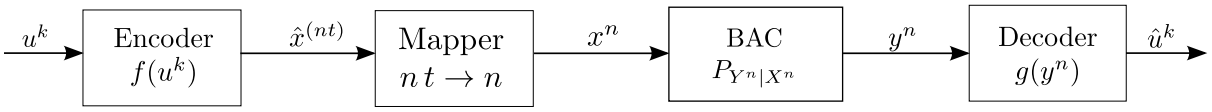


Figure 3.4: Encoding-decoding communication chain with mapping

We have implemented a simple version of this mapping algorithm, the function chosen is a threshold decision rule that takes t bits and depending on their position of in the classical binary representation, assigns or 1 or 0. An example of this algorithm is depicted in figure 3.5. Here, the mapping parameter t is equal to 3, there are 8 possible combinations in the input, we set the threshold equal to 5. In this case, the codewords distribution is biased with probability $q = 5/8 = 0.625$.

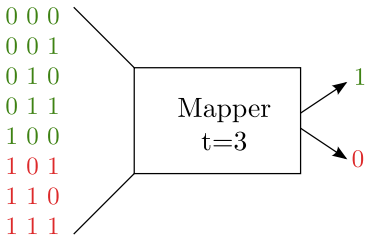


Figure 3.5: Gallager's mapping for $t = 3$

Notice that to find a good distribution probability we need to put a large blocklength linear encoder, which could be a problem for bigger systems.

For example, in order to design a code($n = 16$, $k = 4$), we apply the parameter $t = 5$ to a polar code(80,4) to convert it into a non-linear code(16, 4). The threshold used was 17, obtaining an input distribution probability of $q = 17/32 = 0.53125$. We can see that this is closer to the optimal-capacity distribution for some values of the crossover probabilities.

Integrated Scheme for polar codes

Another solution to is to apply the non-linear function before the linear code, this is proposed in the integrated scheme which was designed for polar codes. In this approach we add a mapper block to the communication chain before the encoder, see figure 3.6. The encoder is a normal polar code, but instead of setting the frozen bits up to zero, the idea here, is to select properly the frozen bits, in order to bias the codewords of the produced codebook.

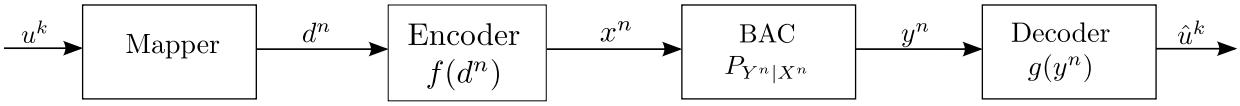


Figure 3.6: Integrated scheme communication chain

The design of this code consists of finding the index of information bits $\mathcal{I}$, uniformly random frozen bits $\mathcal{F}_r$ and the randomized rounding frozen bits $\mathcal{F}_d$. Each set of indexes is determined by two elements, the relation of the i-th bit with the $i-1$ bits transmitted beforehand and the relation of the i-th bit with both the $i-1$ bits transmitted beforehand and the received message y^n. Let us define four sets according to these ideas;

- $\mathcal{H}_X$ the set of indexes of u^k that are independent of $u_{1:i-1}$,
- $\mathcal{L}_X$ the set of indexes of u^k that are deterministic in function of $u^{1:i-1}$,
- $\mathcal{H}_{X|Y}$ the set of indexes of u^k that are independent of $u^{1:i-1}$ and y^n,

- $\mathcal{L}_{X|Y}$ the set of indexes of u^k that are deterministic in function of $u^{1:i-1}$ and y^n,

in [11], they develop the mathematical procedure to find them using the Bhattacharyya parameter.

Thus, the information bits set is defined by $\mathcal{I} = \mathcal{H}_X \cap \mathcal{L}_{X|Y}$, i.e., the set of bits that are a deterministic function of the output y^n only. Meanwhile, the two frozen bits set are defined by, $\mathcal{F}_r = \mathcal{H}_X \cap \mathcal{H}_{X|Y}$ and $\mathcal{F}_d = \mathcal{L}_X$. This is schematically depicted in figure 3.7.

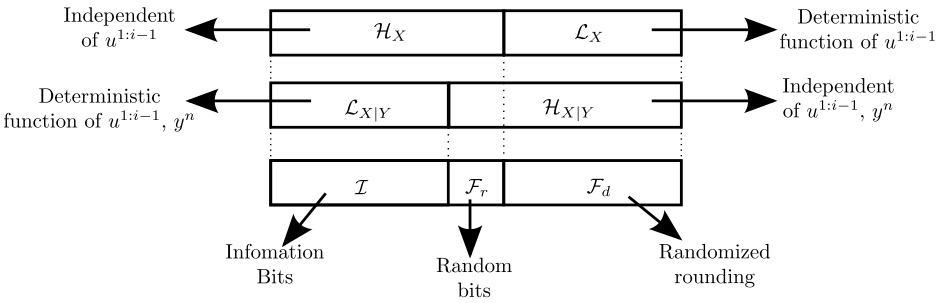


Figure 3.7: Integrated scheme over asymmetric channels

At this point, we know that in $\mathcal{I}$ and $\mathcal{F}_r$ we put the information bits and a set of uniformly random bits, respectively. On the other hand, for $\mathcal{F}_d$ we need to set the value of the i-th bit according to a "randomized random" rule. For the continuation of this internship, we propose to study this rule in more detail.

3.2.3 Numerical results

We notice, in figure 3.8, that compared to the polar code scheme the linear extension of flip codes has better results in terms of BER and BLER. However, if we analyse the generated codebook $C^{(8,4)}_{(t_1,t_2,t_3)}$, the probability distribution associated, assuming equally likely random messages, is $q = 0.5$. In other words, it does not have the optimal distribution for capacity over the BAC, either. On the other hand, the schemes using Gallager's mapping ($t = 10$) are able to reach the optimal distribution but the performance in terms of average error probability is worse than the polar codes, for low-noise channels, e.g. when $\epsilon_0 = 0.00$. Moreover, the association polar codes plus mapping reaches the same performance than linear extended flip codes for high-noise channels, e.g. when $\epsilon_0 \geq 0.75$.

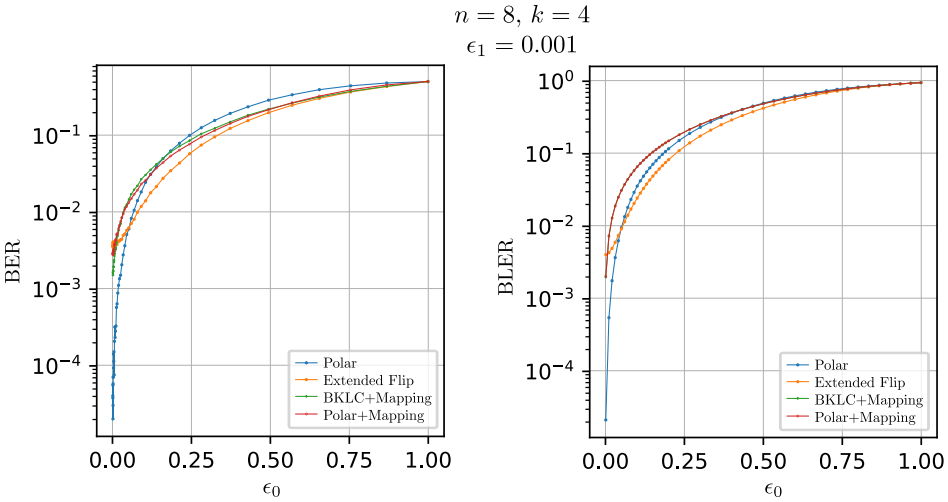


Figure 3.8: BER and BLER for BKLC(8,4), linear extended flip code(8,4), linear code plus mapping(8,4) and polar code plus mapping(8,4) with MAP decoder

In figure 3.9, it is shown that polar codes performs better than the non-linear schemes presented for low-noise channels, when $n = 16$ and $k = 4$. On the other hand, techniques using Gallager's mapping are marginally better if the channel is more noisy, e.g. $\epsilon_0 > 0.5$.

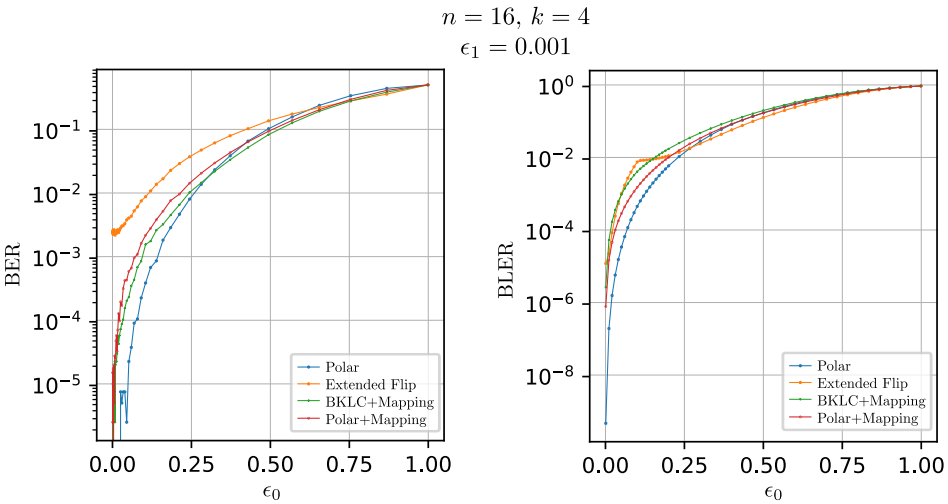


Figure 3.9: BER and BLER for BKLC(16,4), linear extended flip code(16,4, linear code plus mapping(16,4) and polar code plus mapping(16,4) with MAP decoder

To put it in a nutshell, the non-linear techniques presented in this document are a simple way to biased the codebook distribution reaching the optimal distribution for the capacity. Moreover, in terms of average error rate, these techniques are not much better than a normal polar.

Chapter 4

Machine learning-based block codes

In precedent sections it has been presented a state of the art on information theory and coding theory for the binary asymmetric channel, we can notice that it is a mature engineering field that looks for many different ways to improve the coding channel performance. Because of this, there is a high bar of performance that the paradigm of machine learning-based solution can reach or even outperform.

In this section, we start by introducing the basics of deep learning, then, a neural based channel decoder is developed for a polar code and compared with the *maximum a posteriori* channel decoder. Next, an end-to-end autoencoder architecture is introduced and compared with the previously established benchmark.

4.1 Deep learning basics

In general, a neural network with L layers describes a mapping $h(r_0;\theta) : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ of an input vector $r_0 \in \mathbb{R}^{N_0}$ to an output vector $r_L \in \mathbb{R}^{N_L}$ through L iterative processing steps, where $\theta = \{\theta_1, \ldots, \theta_L\}$ is the set of all parameters of the network and N_l the number of neurons per layer. The l-th layer is called *fully-connected* if its function is given by

$$h_l(r_{l-1}, \theta_l) = \sigma(W_l\, r_{l-1} + b_l),$$

where W_l is the set of weights, b_l is the set of bias and $\sigma(\cdot)$ is the activation function. The set of parameters for each layer is given by $\theta_l = \{W_l, b_l\}$. The activation function is capable to introduce a non-linearity to the network giving to neural networks the ability to map complex functions. The set of parameters θ is found through a training process using labeled data, i.e., the network takes an input and changes its set of parameter to get closer to a targeted output r_L^*. Accordingly, a neural network can be summarized as follows

$$h(r_0;\theta) = arg\min_{\theta} \mathcal{L}(r_L^*, r_L), \tag{4.1}$$

where $\mathcal{L}(\cdot,\cdot)$ is the so-called loss function. Another important element to consider for training is the optimizer that refers to the algorithm used to find the set of parameters θ, which minimizes the expression (4.1). All these elements are presented in more details in [15].

The loss function, the optimizer, the function activation and the number of neurons are a part of the so-called *hyperparameters*, the ensemble of parameters used to control the learning process.

4.2 Neural network-based decoder

The optimal decoder, i.e., maximum a posteriori (MAP) decoder, identifies the most likely message $u^k \in \mathcal{U}^k$ given an observation y^n, after an exhaustive search through all possible codewords. Consequently, the complexity of that increases exponentially with the dimension of the blocklength n. Then, the idea of having a neural network that approaches the MAP error probability with low complexity is interesting [2]. We expose here below the architectures and training parameters that make it possible.

Firstly, for a neural network-based decoder we need to define the training data (inputs and labels); The input is, as expected, the codebook with 2^k vectors of n noisy bits received from the channel. On the contrary,for the labels we use *one hot coding*, instead of using the vectors of length k, because it is proven that one hot coding allows to machine learning algorithms to do a better job in prediction. One hot coding consist on mapping each possible vector of length k (category) into a vector of 2^k elements set to zero except one, the index of this nonzero element is different each time. Secondly, we need to define the network architecture; For the model proposed we used only one hidden fully-connected layer, in addition to the input and output layers. The number of neurons for the input layer is the block length n and for the output is 2^k. For the hidden layer, we set the number of neurons as an hyper-parameter multiple of 2^k. The output layer is a fully connected layer with activation function *Softmax*, that returns a probability distribution over predicted output classes.

Moreover, for training we need to define what we call a *meta-model* that contains the decoder model and a stochastic layer that models the noise of the BAC. The latter is implemented by a *lambda* layer, a special type of machine learning layer.

In figure 4.1, we see a scheme of the training scenario. In there, the encoder is the polar encoder previously described, after that, the meta-model with the noise layer and the decoder model.

The following provides a non-exhaustive list of hyper-parameters, used to train neural network-based decoder for a polar code($n = 8$, $k = 4$):

- Number of epochs: 1000
- Batch size: 16000
- Optimizer: *Adam* with learning rate $lr = 0.001$
- Loss function: *Categorical crossentropy*
- Number of neurons in the hidden layer: 80
- Activation function for the hidden layer: ReLU

We need to take into account another parameters that we call the *training crossover probabilities* $\epsilon_{0,t}$ and $\epsilon_{1,t}$. For the binary symmetric channel, an optimal crosover probability minimizing the loss function $\epsilon_t = 0.07$ was proved in [2]. Keeping it in mind, we set the values $\epsilon_{1,t} = 0.07$ and $\epsilon_{0,t} = 0.25$, since we will asses this decoder over a BAC($\epsilon_0 \in [0, 1]$,$\epsilon_1 = 0.001$).

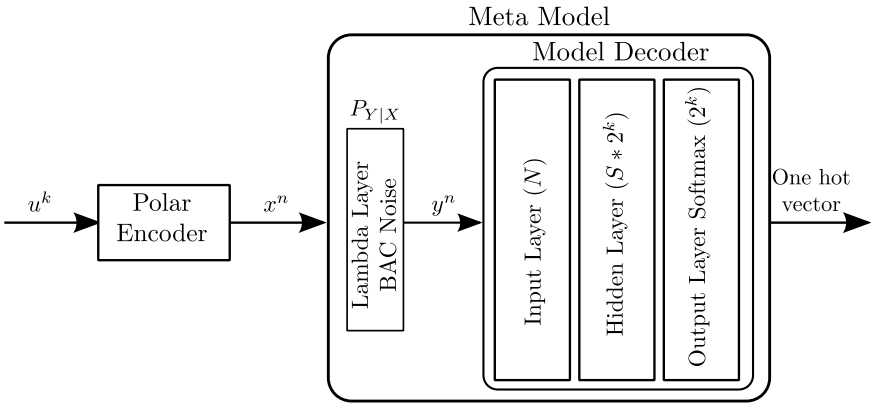


Figure 4.1: Training scenario for neural network-based decoder

In order to use the pre-trained decoder model, we can get rid of the meta model and retain only the decoder. In figure 4.2, it is shown the communication chain when using the neural network-based decoder. We have the polar encoder producing codewords that pass across the binary asymmetric channel. Next, the noisy bits enter to the decoder, which puts in the output a probability distribution of the prediction (one hot vector). Finally, we take the maximum argument and map it back to the associate k-length vector $\hat{u}^k$.

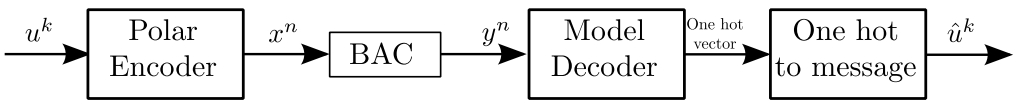


Figure 4.2: Communication chain for neural network-based decoder

Notice that the one hot vector need to be mapped back, however, the complexity of this block is negligible, even for larger codes.

For the binary asymmetric channel do not exist optimal statistics ($\epsilon_{0,t}$, $\epsilon_{1,t}$), which make the neural network-based decoder robust to all crossover probabilities[3]. This is the reason why we use a domain adaptation technique, consisting of giving to the system particular information about the crossover probabilities. In such a way, the model is able to adapt to the decoding problem over the BAC. For that purpose, we propose the training scenario depicted in figure 4.3. Here, we also have a meta model containing the model decoder and the noise layer but, in this case, the noise layer provides to the decoder model an estimation of the channel statistics ($\tilde{\epsilon}_0$) and the noisy bits y^n. The estimation $\tilde{\epsilon}_0$ has length 4 and indicates the range of used ϵ_0, it has been coded using the one hot coding technique. Consequently, the input layer in the decoder model has length $n + 4$. For training, the hyper-parameters are the same as before.

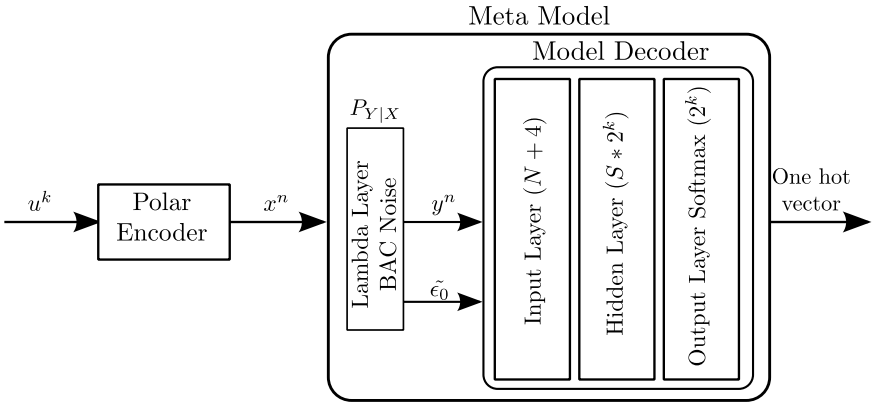


Figure 4.3: Polar encoder and neural network-based decoder scheme with channel estimation

The figure 4.4 shows the communication chain using the model decoder with channel estimation, once again we get rid of the meta channel and keep only the model decoder. In addition to the noisy bits y^n, the model decoder receives another input, the information about the channel crossover probability $\tilde{\epsilon}_0$, meanwhile, we set $\epsilon_{1,t} = 0.002$. In real applications this information need to be estimated directly from the channel and is a current field of research, for our work we consider the channel estimation is already done.

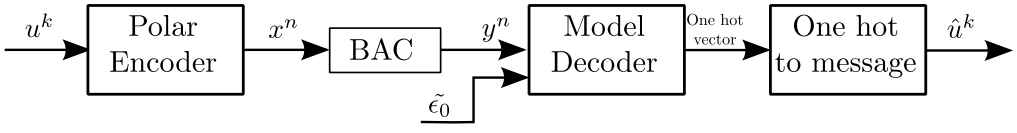


Figure 4.4: Communication chain for neural network-based decoder with channel estimation

4.2.1 Numerical results

In figure 4.5, we show the product of implementing the neural network-based decoder for the BAC and how it behaves in regards to the MAP when $n = 8$ and $k = 4$ over a BAC($\epsilon_0 \in [0, 1]$,$\epsilon_1 = 0.001$). When we use or not a channel estimation, we can notice a good performance for all ϵ_0, in both the BER and BLER. It seams unnecessary to use the channel estimation since we have (almost) the same performance than the MAP with and without it.

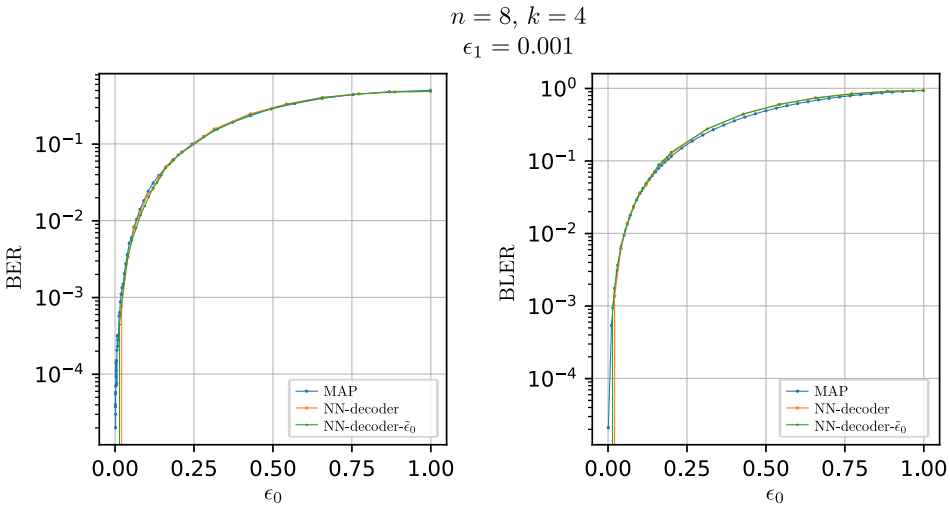


Figure 4.5: BER and BLER for polar code$(8, 4)$ with MAP decoder and NN-based decoder

For $n = 16$ and $k = 4$, the metrics of average error probability over the same set of channels BAC($\epsilon_0 \in [0, 1]$,$\epsilon_1 = 0.001$) are presented in figure 4.6. Now, there is a notable difference when the domain adaptation technique is used, particularly, for the BER.

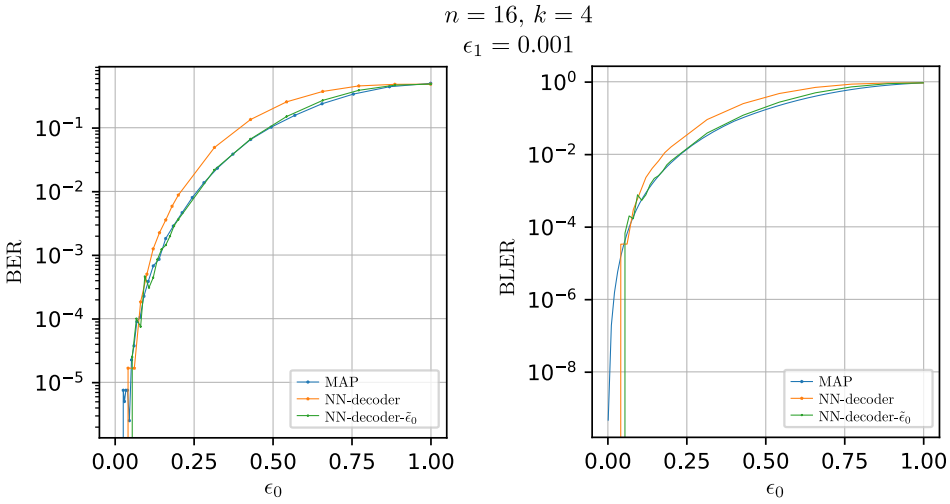


Figure 4.6: BER and BLER for polar code$(16, 4)$ with MAP decoder and NN-based decoder

In short, two techniques of neural network-based decoders have been presented and compared for two coding rates $R = k/n$. In both cases the approach that takes channel information, as input, behaves exactly as the MAP, whereas the more simple technique is not optimal for $R = 1/4$.

4.3 Neural network-based autoencoder

The autoencoders are a type of neural network architecture generally used to perform the identity mapping. In other words, the aim of an autoencoder is to learn to encode input data in small representation and decode it to reproduce the input at the output layer. This technique is used to identify linear and nonlinear correlation in the data [1]. In a sense this is the same as the coding problem, reproducing the input message at the output of the communication chain. Therefore, the autoencoders are considered as a possible alternative to find an optimal solution for complex channels such as the binary asymmetric channel. This section offers a proposal for the development of this solution, recalling that machine learning solutions are broad.

In the first place, we define the training data (inputs and labels). In this case, we use one hot coding of length 2^k as input, as well as, for the expected output. Secondly, we introduce the architecture of the models, we have a *model encoder* and a *model decoder*, as their names indicate one is used for coding and one for decoding section. The encoder is a shallow and width model, this means, a model with a solely hidden fully-connected layer of 2^k times S neurons, where S is a hyper-parameter, then we add a *Batch Normalization* layer that reduces training and prediction time, making the neural network more stable. The input layer contains 2^k and the output layer n neurons. The model decoder also

comprises a solely hidden fully-connected layer of 2^k times S neurons followed by a Batch Normalization layer. This model has n neurons in the input and 2^k neurons in the output.

For training, we define a meta model(see figure 4.7), it contains the models for the encoder and decoder, in addition to a stochastic layer that models the binary asymmetric noise.

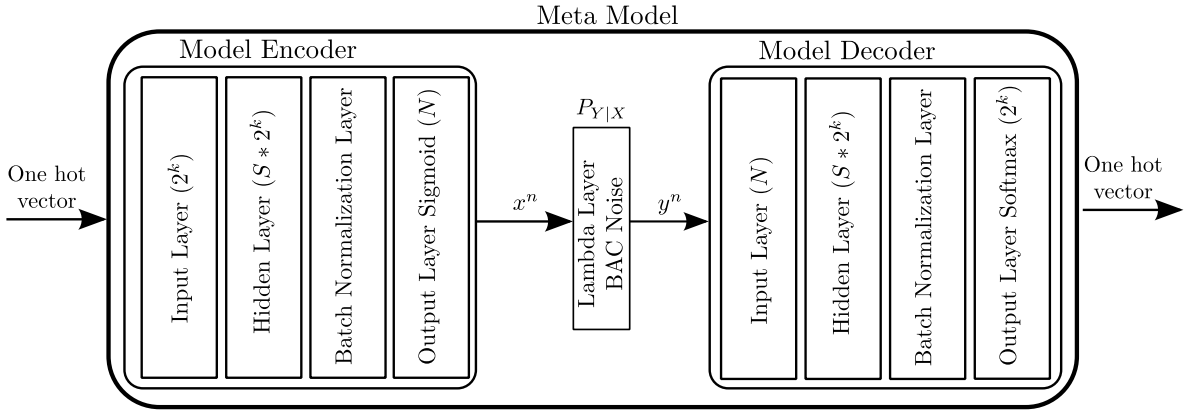


Figure 4.7: Autoencoder layers scheme

The following provides a non-exhaustive list of hyper-parameters, used to train neural network-based autoencode for a code($n = 8$, $k = 4$):

- Number of epochs: 1000
- Batch size: 16000
- Optimizer: *Adam* with learning rate $lr = 0.001$
- Loss function: *Categorical crossentropy*
- Number of neurons in the hidden layer: 80
- Activation function for the hidden layer: Softplus

Additionally, we set the values $\epsilon_{1,t} = 0.07$ and $\epsilon_{0,t} = 0.25$, since we design this encoder and this decoder to perform correctly over a BAC($\epsilon_0 \in [0, 1]$,$\epsilon_1 = 0.001$).

In figure 4.8, we show how to use the pre-trained models. In this communication chain, first of all, we need to map the bit sequence u^k to its respectively one hot vector. Next, we use the model encoder to obtain the codeword x^n, this latter pass through the BAC and the received noisy codeword y^n is used as input for the model decoder, which returns a one hot vector which, in turn, is mapped to an estimated sequence $\hat{u}^k$.

Figure 4.8: Communication chain for neural network-based encoder and decoder

The block *one hot to message* applies the inverse function of the block *message to one hot*.

A second approach is similar to the one used for the neural network-based decoder, it consists of giving to the system particular information about the crossover probabilities. The models aims to adapt itself to encoding and decoding problems over the BAC. With that in mind, we propose the training scenario depicted in figure 4.9. In this case, we also have a meta model containing the model encoder, model decoder and noise layer but, in this case, an extra input provides an estimation of the channel statistics ($\tilde{\epsilon}_0$). The estimation $\tilde{\epsilon}_0$ has length 4 and indicates the range of the used ϵ_0, it has been coded using the one hot coding technique. Consequently, the input layer in the encoder has length $2^k + 4$ and the input layer in the decoder has length $n + 4$. For training, the hyper-parameters are the same as established before.

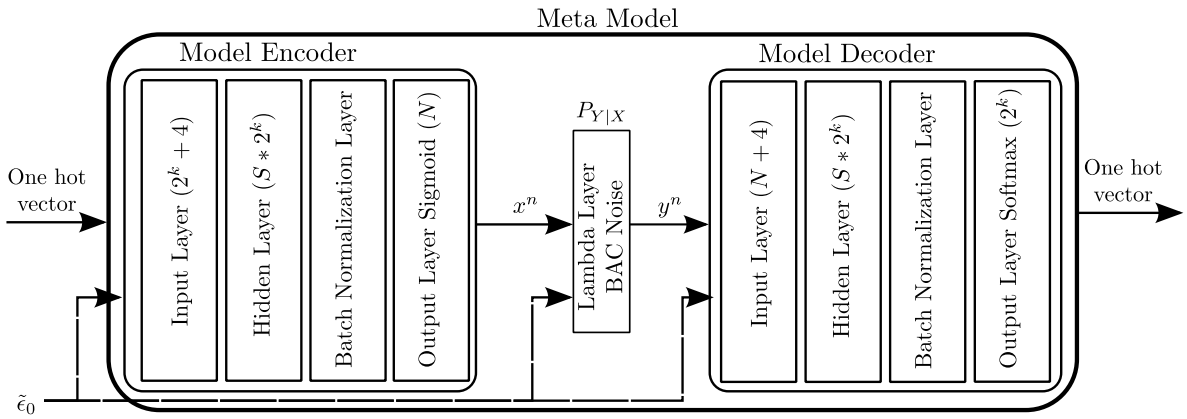


Figure 4.9: Autoencoder layers scheme with channel estimation

The communication chain shown in figure 4.10 depicts how to use the models pre-trained, therein, we can observe the extra-input in the models, it represents the estimation of the channel. As we said before, the way how this estimation is obtained is far beyond the scope of this project.

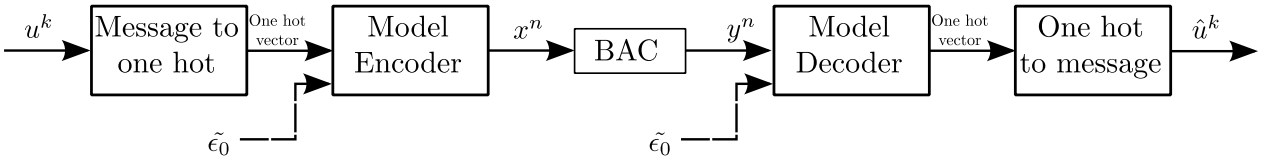


Figure 4.10: Communication chain for neural network-based encoder and decoder with channel estimation

4.3.1 Numerical results

A comparison of the performance (BER and BLER) of the coding structures learnt by the autoencoder architectures is presented in figure 4.11. We have the polar codes (benchmark), the autoencoder and the autoencoder with estimation of the channel when $n = 8$ and $k = 4$ over a BAC($\epsilon_0 \in [0, 1]$,$\epsilon_1 = 0.001$). Concerning the simpler autoencoder, one can notice a performance very similar to the polar codes for all ϵ_0, with a higher loss around $\epsilon_0 = 0$, in both the BER and BLER. With respect to the autoencoder with channel estimation, on green in the figure, we can notice a marginal improvement when the channel is more asymmetrical, e.g., $\epsilon_1 = 0.001$ and $\epsilon_0 \approx 0.25$.

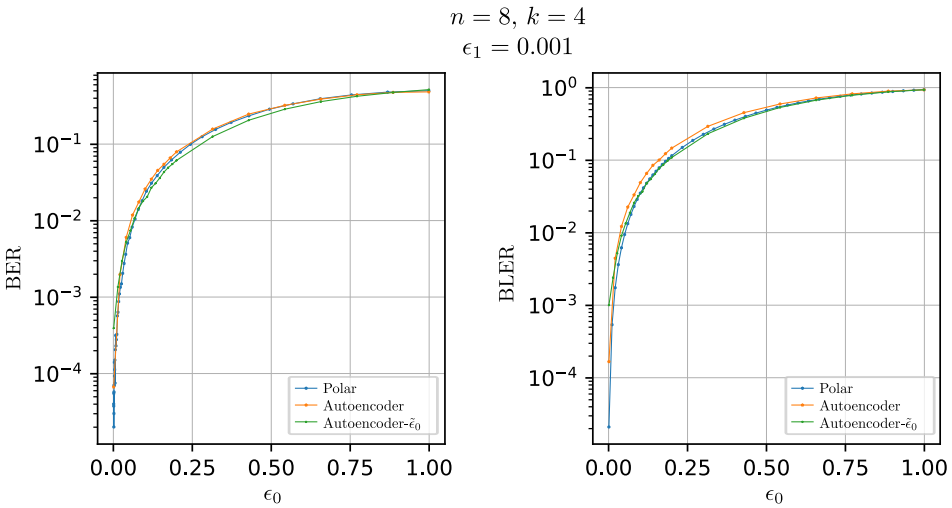


Figure 4.11: BER and BLER for Polar Code$(8, 4)$ and autoencoder$(8, 4)$

For $n = 16$ and $k = 4$, the metrics of average error probability over the same set of channels, BAC($\epsilon_0 \in [0, 1]$,$\epsilon_1 = 0.001$), are presented in figure 4.6. Now, the difference is notable when the we use a domain adaptation technique, particularly, for the BER. Nevertheless, this improvement is not meaningful in practice.

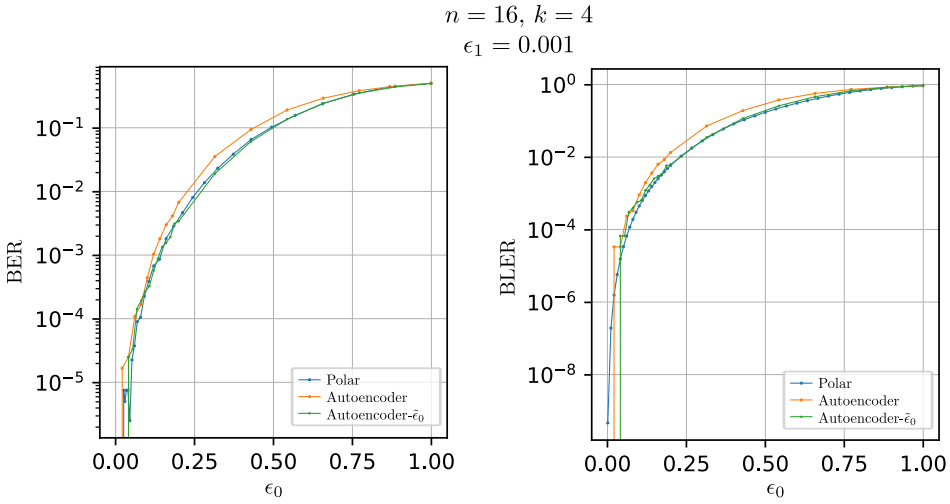


Figure 4.12: BER and BLER for Polar Code$(16, 4)$ and autoencoder$(16, 4)$

In terms of codebook probability distribution, both the simpler autoencoder and the autoencoder with channel estimation have non-uniform probabilities. In the first scenario ($n = 8,\ k = 4$), the resulting codebook has a probability of 0.53 and the second one has a probability of 0.57, this means the autoencoder with channel channel estimation produces a codebook closer to the optimal capacity distribution $q^* \to 0.6$ (for $\epsilon_1 \to 0$ and $\epsilon_0 = 0.5$, see figure 2.4).

In brief, the machine learning-base solution presented were proved to have a performance similar to polar codes for the BAC. However, the autoencoder with channel estimation has a small advantage over the others in terms of the average error rate, besides that, according to its codebook, it has an input distribution closer to the optimal than the others.

The objective for the future work is to develop another architectures attempting to improve significantly the average error probability. For that, another set of hyper-parameters need to be tested or, why not, another kind of neural network. The first option is in process of implementation, so far with no better results than those presented.

Chapter 5

Conclusions

This work investigated multiple approaches to design a block code that minimizes the average error probability over the binary asymmetric channel. It has been proved that well-known linear codes do not have the optimal distribution that might enable them to reach the theoretical capacity. It was proved that another kind of constructions can perform better, specifically the linear extended flip codes. The objective in this research internship was to develop an end-to-end neural network-based autoencoder that learns a completely unknown coding structure. It was found that it is possible to train a machine learning model that replicates the best known coding-decoding systems, in terms of average error probability.

In summary, the results obtained during this study are the base of a work that needs to be continued. Especially, if we consider the immense potential described in literature for autoencoders in communication systems. Also, because machine learning models used in this project were simple and does not explore all the possible hyperparameters or more matured training algorithms.

Bibliography

[1] Kramer M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233 – 243, 1991.

[2] M. Benammar and P. Piantanida. Optimal training channel statistics for neural-based decoders. *52nd Asilomar Conference on Signals, Systems, and Computers*, pages 2157–2161, 2018.

[3] M. Benammar and P. Piantanida. On robust deep neural decoders. *53rd Asilomar Conference on Signals, Systems, and Computers*, pages 527–531, 2019.

[4] W.; Fieker C. Cannon, J.; Bosma and A. Steel. *Handbook of Magma Functions, Volume 13 Coding Theory and Cryptography.* Sidney, 2008.

[5] H. Y. Chen, P. N.; Lin and S. M. Moser. Weak flip codes and applications to optimal code design on the binary erasure channel. *2012 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012.

[6] H. Y. Chen, P. N.; Lin and S. M. Moser. Optimal ultrasmall block-codes for binary discrete memoryless channels. *IEEE Transactions on Information Theory*, 59(11):7346–7378, 2013.

[7] Robert G. Gallager. *Information Theory and Reliable Communication.* John Wiley and Sons, Inc., Massachusetts Institute of Technology, 1968.

[8] S.; Hoydis J. Gruber, T.; Cammerer and S. ten Bring. On deep learning-based channel decoding. *51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1 – 6, 2017.

[9] J. Honda and H. Yamamoto. Polar coding without alphabet extension for asymmetric models. *IEEE Transactions on Information Theory*, 59(12):7829 – 7838, 2013.

[10] E. E. Majani and H. Rumsey. Two results on binary input discrete memoryless channels. *IEEE International Symposium on Information Theory*, 1991.

[11] S. H. Mondelli, M.; Hassani and R. L. Urbanke. How to achieve the capacity of asymmetric channels. *IEEE Transactions on Information Theory*, 64(5):3371 – 3393, 2018.

[12] P. N. Moser, S. M.; Chen and H. Y. Lin. Error probability analysis of binary asymmetric channels. *IEEE Transactions on Information Theory*, 2012.

[13] Stefan M. Moser. *Information Theory Lecture Notes*. ETH Zürich and NCTU, Zürich and Taiwan, 2020.

[14] J. Muramatsu and S. Miyake. Stochastic decision with stationary memoryless sequence. *IEEE International Symposium on Information Theory*, 2019.

[15] T. O'Shea and J. Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563 – 575, 2017.

[16] H. Polyanskiy, Y.; Vincent Poor and Verdu S. Channel coding rate in the finite blocklength regime. *IEEE Transactions on Information Theory*, 56(5):2307–2359, 2010.

[17] C. E. Shannon. A mathematical theory of communications. *Bell System Technical Journal*, 1948.

[18] Y. Vangala, H.; Hong and E. Viterbo. Efficient algorithms for systematic polar encoding. *IEEE Communications Letters*, 20(1):17 – 20, 2016.

Appendices

Appendix A

Proofs

A.1 Average block error probability

Let us begin the proof by finding the joint probability of the channel, which is given by

$$P_{U^k,X^n,Y^n,\hat{U}^k}(u^k,x^n,y^n,\hat{u}^k) = \\ P_{Y^n}(y^n)\,P_{\hat{U}^k|Y^n}(\hat{u}^k|y^n)\,P_{U^k|Y^n,\hat{U}^k}(u^k|y^n,\hat{u}^k)\,P_{X^n|Y^n,\hat{U}^k,U^k}(x^n|y^n,\hat{u}^k,u^k),$$

where U^k and $\hat{U}^k$ are conditionally independent given Y^n and X^n, according to Markov chain (2.2). Hence, we have that $P_{U^k|Y^n,\hat{U}^k}(u^k|y^n,\hat{u}^k) = P_{U^k|Y^n}(u^k|y^n)$.

On the other hand, in view of a deterministic coding rule, i.e.,

$$P_{U^k|X^n}(u^k|x^n) = \begin{cases} 1, \text{ if } x^n = f(u^k) \\ 0, \text{ if } x^n \neq f(u^k), \end{cases}$$

we have that $P_{X^n|Y^n,\hat{U}^k,U^k}(x^n|y^n,\hat{u}^k,u^k) = 1$ then the joint probability of the channel can be written as

$$P_{U^k,X^n,Y^n,\hat{U}^k}(u^k,x^n,y^n,\hat{u}^k) = P_{Y^n}(y^n)\,P_{\hat{U}^k|Y^n}(\hat{u}^k|y^n)\,P_{U^k|Y^n}(u^k|y^n). \tag{A.1}$$

By definition,

$$P_e = P(\hat{u}^k \neq u^k) \triangleq \sum_{y^n}\sum_{u^k}\sum_{\hat{u}^k} P_{U^k,X^n,Y^n,\hat{U}^k}(u^k,x^n,y^n,\hat{u}^k)\,\mathcal{I}(\hat{u}^k \neq u^k),$$

then, replacing the expression for the joint probability (A.1) on the definition of block error probability P_e, we find the expression for average error probability given by

$$P_e = \sum_{y^n} P_{Y^n}(y^n) \sum_{u^k} P_{U^k|Y^n}(u^k|y^n) \sum_{\hat{u}^k} P_{\hat{U}^k|Y^n}(\hat{u}^k|y^n)\,\mathcal{I}(\hat{u}^k \neq u^k).$$

A.2 Optimal decoder

The proof of this theorem follows by considering that $\sum_{\hat{u}^k} Q_{\hat{U}^k|Y^n}(\hat{u}^k, y^n)\,\mathcal{I}(\hat{u}^k \neq u^k) = \big(1 - Q_{\hat{U}^k|Y^n}(u^k, y^n)\big)$, by random variable properties. Then, the average error probability P_e, as a function of the decision rule $Q_{\hat{U}^k|Y^n}$, can be written as

$$P_e(Q_{\hat{U}^k|Y^n}) = \sum_{y^n} P_{Y^n}(y^n) \sum_{u^k} P_{U^k|Y^n}(u^k|y^n) \left(1 - Q_{\hat{U}^k|Y^n}(u^k, y^n)\right),$$

or, which is the same

$$P_e(Q_{\hat{U}^k|Y^n}) = 1 - \sum_{y^n} P_{Y^n}(y^n) \sum_{u^k} P_{U^k|Y^n}(u^k|y^n)\, Q_{\hat{U}^k|Y^n}(u^k, y^n). \tag{A.2}$$

Defining $g(y^n) \triangleq g_{MAP}(y^n)$ and replacing it in (A.2), one obtains

$$\begin{aligned} P_e(Q_{\hat{U}^k|Y^n}) = 1 - \sum_{y^n} P_{Y^n}(y^n) \Bigg(& \sum_{u^k \neq g_{MAP}(y^n)} P_{U^k|Y^n}(u^k|y^n)\, Q_{\hat{U}^k|Y^n}(u^k, y^n) \\ & + P_{U^k|Y^n}(g_{MAP}(y^n)|y^n)\, Q_{\hat{U}^k|Y^n}(g_{MAP}(y^n), y^n) \Bigg), \end{aligned}$$

replacing $Q_{\hat{U}^k|Y^n}(g_{MAP}(y^n), y^n)$ by $1 - \sum_{u^k \neq g_{MAP}(y^n)} Q_{\hat{U}^k|Y^n}(u^k, y^n)$, we have

$$\begin{aligned} P_e(Q_{\hat{U}^k|Y^n}) = 1 - \sum_{y^n} P_{Y^n}(y^n) \Bigg(& \sum_{u^k \neq g_{MAP}(y^n)} P_{U^k|Y^n}(u^k|y^n)\, Q_{\hat{U}^k|Y^n}(u^k, y^n) \\ & + P_{U^k|Y^n}(g_{MAP}(y^n)|y^n) \left(1 - \sum_{u^k \neq g_{MAP}(y^n)} Q_{\hat{U}^k|Y^n}(u^k, y^n)\right) \Bigg), \end{aligned}$$

regrouping,

$$\begin{aligned} P_e(Q_{\hat{U}^k|Y^n}) = & 1 - \sum_{y^n} P_{Y^n}(y^n)\, P_{U^k|Y^n}(g_{MAP}(y^n)|y^n) \\ & + \sum_{y^n} P_{Y^n}(y^n) \sum_{u^k \neq g_{MAP}(y^n)} Q_{\hat{U}^k|Y^n}(u^k, y^n) \Big(P_{U^k|Y^n}(g_{MAP}(y^n)|y^n) - P_{U^k|Y^n}(u^k|y^n) \Big), \end{aligned}$$

in this result of average error derived from its definition, where $P_{U^k|Y^n}(g_{MAP}(y^n)|y^n) - P_{U^k|Y^n}(u^k|y^n) \geq 0$ implies that $P_e(Q_{\hat{U}^k|Y^n})$ is minimized only when $Q_{\hat{U}^k|Y^n}(u^k, y^n) = 0$ for all (u^k, y^n) such that $P_{Y^n}(y^n) > 0$ and $P_{U^k|Y^n}(g_{MAP}(y^n)|y^n) > P_{U^k|Y^n}(u^k|y^n)$ (i.e., $u^k \neq g_{MAP}(y^n)$). In short, if we accept the definition of the theorem, the minimal average error probability is given by

$$P_e = 1 - \sum_{y^n} P_{Y^n}(y^n) \max_{\hat{u}^k} P_{U^k|Y^n}(\hat{u}^k|y^n). \tag{A.3}$$

This completes the proof that MAP decision rule minimizes the average error probability for any encoded transmission.

A.3 Maximum likelihood equivalence

The proof of the corollary 2.1 follows from the definition of conditional probability

$$P_{U^k|Y^n}(\hat{u}^k|y^n) = \frac{P_{\hat{U}^k,Y^n}(\hat{u}^k, y^n)}{P_{Y^n}(y^n)},$$

then, the MAP decision rule (2.9) can be expressed as

$$g_{MAP}(y^n) = arg\,\max_{\hat{u}^k} \frac{P_{U^k,Y^n}(\hat{u}^k, y^n)}{P_{Y^n}(y^n)},$$

as $P_{Y^n}(y^n)$ does not depend on $\hat{u}^k$, then it can be ignored in the maximization, as follows,

$$\begin{aligned} g_{MAP}(y^n) &= arg\,\max_{\hat{u}^k} P_{U^k,Y^n}(\hat{u}^k, y^n), \\ &= arg\,\max_{\hat{u}^k} P_{U^k}(\hat{u}^k)\, P_{Y^n|U^k}(y^n|\hat{u}^k). \end{aligned} \tag{A.4}$$

In (A.4), if U^k is a binary uniformly distributed random variable then the probability $P_{U^k}(u^k) = cte$. Consequently, it does not affect in the maximization either, thus we obtain

$$g_{MAP}(y^n) = arg\,\max_{\hat{u}^k} P_{Y^n|U^k}(y^n|\hat{u}^k) = g_{ML}(y^n).$$

Therefore, one can affirm that the Maximum Likelihood is an optimal decision rule if, and only if, a equiprobable input is injected into the system.

A.4 Binary asymmetric channel mutual information

Let us start following the remark 1.

The first term in (2.7) is the entropy of channel output $H(Y)$, defined by:

$$H(Y) = -\sum_{y} P_Y(y)\, \log_2(P_Y(y)),$$

Considering its probability (2.11), we have that

$$\begin{aligned} H(Y) &= -u\, \log_2(u) - (1-u)\, \log_2(1-u), \\ &= h_2(u), \\ &= h_2\big((1-q)\,\epsilon_0 + (1-\epsilon_1)\,q\big), \end{aligned} \tag{A.5}$$

where $h_2(\cdot)$ is the binary entropy function, detailed in appendix B.

The second term in (2.7) is the conditional entropy of the output knowing the input $H(Y|X)$, defined by:

$$H(Y|X) = \sum_{x} P_X(x) H(Y|X=x), \tag{A.6}$$

this will be computed in two steps, evaluating the possible values of x.

For $x = 0$,

$$\begin{aligned} H(Y|X=0) &= -\sum_{y} P_{Y|X}(y|0)\, \log_2\big(P_{Y|X}(y|0)\big), \\ &= -(1-\epsilon_0)\, \log_2(1-\epsilon_0) - \epsilon_0\, \log_2(\epsilon_0), \\ &= h_2(\epsilon_0), \end{aligned} \tag{A.7}$$

and for $x = 1$

$$\begin{aligned} H(Y|X=1) &= -\sum_{y} P_{Y|X}(y|1)\, \log_2\big(P_{Y|X}(y|1)\big), \\ &= -\epsilon_1\, \log_2(\epsilon_1) - (1-\epsilon_1)\, \log_2(1-\epsilon_1), \\ &= h_2(\epsilon_1), \end{aligned} \tag{A.8}$$

then, considering (A.7) and (A.8), the conditional entropy (A.6) can be written as:

$$H(Y|X) = (1-q)\, h_2(\epsilon_0) + h_2(\epsilon_1)\, q. \tag{A.9}$$

Finally, replacing the output entropy (A.5) and the channel conditional entropy (A.9) into the formula (2.7) for mutual information, one obtains:

$$I(X;Y)_{BAC} = h_2\big((1-q)\epsilon_0 + q(1-\epsilon_1)\big) - (1-q)\, h_2(\epsilon_0) - h_2(\epsilon_1)\, q.$$

Remark 4 (Special cases of mutual information)
We can precise the following conclusions from the result of the appendix A.4:

- *Considering $\epsilon_0 = \epsilon_1 = \epsilon$, i.e. a binary symmetric channel BSC(ϵ), replacing in* (2.12)*, we obtain:*

$$I(X;Y)_{BSC} = h_2\big((1-q)\,\epsilon + (1-\epsilon)\,q\big) - h_2(\epsilon),$$

  the formula which defines the mutual information for the BSC.

- *Considering $\epsilon_0 + \epsilon_1 = 1$ then $u = \epsilon_0$, the mutual information is given by:*

$$\begin{aligned} I(X;Y) &= h_2(\epsilon_0) - (1-q)\,h_2(\epsilon_0) - h_2(\epsilon_1)\,q, \\ &= \big(h_2(\epsilon_0) - h_2(\epsilon_1)\big)\,q, \\ &= \big(h_2(1-\epsilon_1) - h_2(\epsilon_1)\big)\,q = 0, \end{aligned}$$

  since $h_2(1-\epsilon_1) = h_2(\epsilon_1)$.

  In the literature, this type of channel is called completely noisy channels, because the mutual information between the input and output is always zero, thereby the channel capacity is also zero.

A.5 Optimal Input Distribution q

In order to maximize the mutual information $I(X;Y)$; firstly, we apply the partial derivative with respect to q, then, we set it equal to zero and, finally, we find the value of q that fulfils the equation.

Using the expression of mutual information given in (2.12), we have that

$$\frac{\partial I(X;Y)}{\partial q} = \frac{\partial}{\partial q}\Big(h_2(u)\Big) - \frac{\partial}{\partial q}\Big((1-q)\,h_2(\epsilon_0)\Big) - \frac{\partial}{\partial q}\Big(q\,h_2(\epsilon_1)\Big), \tag{A.10}$$

where the partial derivative of the binary entropy $h_2(u)$ can be decomposed into two terms, as follows

$$\frac{\partial}{\partial q}\Big(h_2(u)\Big) = -\frac{\partial}{\partial q}\Big(u\,\log_2(u)\Big) - \frac{\partial}{\partial q}\Big((1-u)\,\log_2(1-u)\Big),$$

we have for the first term that

$$\frac{\partial}{\partial q}\Big(u\,\log_2(u)\Big) = \frac{\big(1-\epsilon_1-\epsilon_0\big)\Big(ln\big((1-q)\,\epsilon_0+(1-\epsilon_1)\,q\big)+1\Big)}{ln(2)}$$

and for the second term that

$$\frac{\partial}{\partial q}\Big((1-u)\,\log_2(1-u)\Big) = -\frac{\big(1-\epsilon_1-\epsilon_0\big)\Big(ln\big((1-q)(1-\epsilon_0)+q\,\epsilon_1\big)+1\Big)}{ln(2)}.$$

Then, replacing these results in (A.10), we have

$$\begin{aligned}\frac{\partial I(X;Y)}{\partial q} =& -\frac{\big(1-\epsilon_1-\epsilon_0\big)\Big(ln\big((1-q)\,\epsilon_0+(1-\epsilon_1)\,q\big)+1\Big)}{ln(2)} \\ &+\frac{\big(1-\epsilon_1-\epsilon_0\big)\Big(ln\big((1-q)(1-\epsilon_0)+q\,\epsilon_1\big)+1\Big)}{ln(2)} + h_2(\epsilon_0) - h_2(\epsilon_1),\end{aligned}$$

regrouping, we can rewrite it as

$$\begin{aligned}\frac{\partial I(X;Y)}{\partial q} =& \frac{\big(1-\epsilon_1-\epsilon_0\big)\Big(ln\big((1-q)(1-\epsilon_0)+q\,\epsilon_1\big)+1-ln\big((1-q)\,\epsilon_0+(1-\epsilon_1)\,q\big)-1\Big)}{ln(2)} \\ &+ h_2(\epsilon_0) - h_2(\epsilon_1),\end{aligned}$$

or, which is the same

$$\begin{aligned}\frac{\partial I(X;Y)}{\partial q} =& \big(1-\epsilon_1-\epsilon_0\big)\Big(\log_2\big((1-q)(1-\epsilon_0)+q\,\epsilon_1\big) - \log_2\big((1-q)\,\epsilon_0+(1-\epsilon_1)\,q\big)\Big) \\ &+ h_2(\epsilon_0) - h_2(\epsilon_1).\end{aligned}$$

Now, we set this result equal to zero and find the value of q.

$$\big(1-\epsilon_1-\epsilon_0\big)\Big(\log_2\big((1-q)(1-\epsilon_0)+q\,\epsilon_1\big)-\log_2\big((1-q)\,\epsilon_0+(1-\epsilon_1)\,q\big)\Big)+h_2(\epsilon_0)-h_2(\epsilon_1)=0,$$

passing some elements at the other side of the equation, we have

$$\frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{(1-\epsilon_1-\epsilon_0)}=\log_2\big((1-q)\,\epsilon_0+(1-\epsilon_1)\,q\big)-\log_2\big((1-q)(1-\epsilon_0)+q\,\epsilon_1\big),$$

then, applying the power of 2 in both sides of the equation, the result is

$$2^{\frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{(1-\epsilon_1-\epsilon_0)}}=\frac{(1-q)\,\epsilon_0+(1-\epsilon_1)\,q}{(1-q)(1-\epsilon_0)+q\,\epsilon_1},$$

uniquely, with $\epsilon_0+\epsilon_1<1$.

Let us define the auxiliary variable z as

$$z\triangleq 2^{\frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{(1-\epsilon_1-\epsilon_0)}},$$

then, by induction

$$z=\frac{(1-q)\,\epsilon_0+(1-\epsilon_1)\,q}{(1-q)(1-\epsilon_0)+q\,\epsilon_1}. \tag{A.11}$$

There, it only remains to find the value of q^* optimal in (A.11), we likewise find the expression for $1-q^*$, as

$$\begin{aligned} q^*=P_X^*(x=1)&=\frac{z-(z+1)\,\epsilon_0}{(z+1)(1-\epsilon_0-\epsilon_1)}\\ (1-q^*)=P_X^*(x=0)&=\frac{1-(z+1)\,\epsilon_1}{(z+1)(1-\epsilon_0-\epsilon_1)}. \end{aligned} \tag{A.12}$$

A.6 Binary asymmetric channel capacity

The proof of this theorem resides in a simple replacement of q^*, the result obtained in (2.14), in the expression for the mutual information (2.13).

Let us begin by finding the expression for $u^* = u(q^*)$, replacing (2.14) in the definition of u, we obtain:

$$u^* = (1 - q^*)\,\epsilon_0 + (1 - \epsilon_1)\,q^* = \frac{z}{z+1}, \tag{A.13}$$

using this result in (2.13), one obtains:

$$C_{BAC}(\epsilon_0, \epsilon_1) = h_2\left(\frac{z}{z+1}\right) - (1 - q^*)\; h_2(\epsilon_0) - q^*\; h_2(\epsilon_1). \tag{A.14}$$

Then, remembering the definition of the binary entropy, the first term of the capacity can be expressed as:

$$\begin{aligned} h_2\left(\frac{z}{z+1}\right) &= -\frac{z}{z+1}\;\log_2\left(\frac{z}{z+1}\right) - \frac{1}{z+1}\;\log_2\left(\frac{1}{z+1}\right), \\ &= -\frac{\log_2\left(\left(\frac{z}{z+1}\right)^z\left(\frac{1}{z+1}\right)\right)}{z+1}, \\ &= \log_2(z+1) - \frac{z\;\log_2(z)}{z+1}, \end{aligned}$$

and knowing from (2.15) that $\log_2(z) = \frac{h_2(\epsilon_0) - h_2(\epsilon_1)}{(1-\epsilon_0-\epsilon_1)}$, we replace it and obtain the expression for this binary entropy

$$h_2\left(\frac{z}{z+1}\right) = \log_2(z+1) - \frac{z\;h_2(\epsilon_0) - z\;h_2(\epsilon_1)}{(z+1)(1-\epsilon_0-\epsilon_1)}. \tag{A.15}$$

Now we can replace this result and those from (2.14) in the capacity expression (A.14), the we obtain that

$$C_{BAC}(\epsilon_0, \epsilon_1) = \log_2(z+1) - \frac{z\;h_2(\epsilon_0) - z\;h_2(\epsilon_1)}{(z+1)(1-\epsilon_0-\epsilon_1)} - \frac{\big(1 - (z+1)\,\epsilon_1\big)\;h_2(\epsilon_0) + \big(z - (z+1)\,\epsilon_0\big)\;h_2(\epsilon_1)}{(z+1)(1-\epsilon_0-\epsilon_1)},$$

that can be simplified as

$$C_{BAC}(\epsilon_0, \epsilon_1) = \log_2(z+1) - \frac{(1-\epsilon_1)\;h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_0\;h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1}.$$

Finally, replacing the value of z, obtained in (2.15), ze have the expression of capacity for the binary asymmetric channel:

$$C_{BAC}(\epsilon_0, \epsilon_1) = \log_2\left(2^{\frac{h_2(\epsilon_0) - h_2(\epsilon_1)}{(1-\epsilon_1-\epsilon_0)}} + 1\right) - \frac{(1-\epsilon_1)\;h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_0\;h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1},$$

where it is important to notice that the capacity for these channels also depends on the crossover probabilities.

A.7 Symmetry in the capacity

- To prove that $C_{BAC}(\epsilon_0, \epsilon_1) = C_{BAC}(\epsilon_1, \epsilon_0)$, i.e., symmetrical about $\epsilon_0 = \epsilon_1$, we replace ϵ_0 for ϵ_1, and vice versa, in (2.16), then we have that:

$$\begin{aligned} C_{BAC}(\epsilon_1, \epsilon_0) &= log_2\left(\frac{1}{z}+1\right) - \frac{(1-\epsilon_0)\,h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_1\,h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1}, \\ &= log_2(z+1) - \frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1} - \frac{(1-\epsilon_0)\,h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_1\,h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1}, \end{aligned}$$

  which can be simplified as

$$C_{BAC}(\epsilon_1, \epsilon_0) = log_2\,(z+1) - \frac{(1-\epsilon_1)\;h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_0\;h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1}, \tag{A.16}$$

  obtaining an expression for the capacity $C_{BAC}(\epsilon_1, \epsilon_0)$ equal to $C_{BAC}(\epsilon_0, \epsilon_1)$, according to (2.16):

- To prove that $C_{BAC}(\epsilon_0, \epsilon_1) = C_{BAC}(1-\epsilon_0, 1-\epsilon_1)$, i.e., symmetrical about $\epsilon_0+\epsilon_1 = 1$, we replace ϵ_0 for $1-\epsilon_0$, and ϵ_1 for $1-\epsilon_1$, in the (2.16), then one obtains:

$$C_{BAC}(1-\epsilon_0, 1-\epsilon_1) = log_2\left(\frac{1}{z}+1\right) + \frac{\epsilon_1\,h_2(1-\epsilon_0)}{1-\epsilon_0-\epsilon_1} - \frac{(1-\epsilon_0)\,h_2(1-\epsilon_1)}{1-\epsilon_0-\epsilon_1},$$

  remembering the symmetry of the binary entropy, i.e., $h_2(\epsilon) = h_2(1-\epsilon)$, then

$$C_{BAC}(1-\epsilon_0, 1-\epsilon_1) = log_2(z+1) - \frac{h_2(\epsilon_0)-h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1} + \frac{\epsilon_1\,h_2(\epsilon_0)}{1-\epsilon_0-\epsilon_1} - \frac{(1-\epsilon_0)\,h_2(\epsilon_1)}{1-\epsilon_0-\epsilon_1},$$

  likewise, simplifying we obtain the same expression for the capacity, as in (2.16):

$$C_{BAC}(1-\epsilon_0, 1-\epsilon_1) = log_2\,(z+1) - \frac{1-\epsilon_1}{1-\epsilon_0-\epsilon_1}\;h_2(\epsilon_0) + \frac{\epsilon_0}{1-\epsilon_0-\epsilon_1}\;h_2(\epsilon_1). \tag{A.17}$$

Similar demonstrations are omitted for the cases $C_{BAC}(\epsilon_0, 1-\epsilon_1)$ and $C_{BAC}(1-\epsilon_0, \epsilon_1)$.

A.8 Channel probability for the BAC

Considering that the BAC is a memoryless channels and there is not feedback in it, the conditional probability for the sequences can be written as:

$$P_{Y^n|X^n}\left(y^n|f(u^k)\right) = \prod_{i=1}^{n} P_{Y|X}(y_i^n|f(u^k)_i), \tag{A.18}$$

where y_i^n is the i-th element of the block y^n.

In a BAC each element of the product in (A.18) can be expressed as:

$$P_{Y^n|X^n}\left(y_i^n|f(u^k)_i\right) = \epsilon_0^{y_i^n \wedge \overline{f(u^k)_i}} (1-\epsilon_0)^{\overline{y_i^n} \wedge \overline{f(u^k)_i}} \epsilon_1^{\overline{y_i^n} \wedge f(u^k)_i} (1-\epsilon_1)^{y_i^n \wedge f(u^k)_i}$$

Let $\overline{a}$ be the *negation* of a and $\wedge$ be the logic operator *and*. Then the product becomes:

$$\begin{aligned} P_{Y^n|X^n}(y^n|f(u^k)) &= \prod_{i=1}^{n} \epsilon_0^{y_i^n \wedge \overline{f(u^k)_i}} (1-\epsilon_0)^{\overline{y_i^n} \wedge \overline{f(u^k)_i}} \epsilon_1^{\overline{y_i^n} \wedge f(u^k)_i} (1-\epsilon_1)^{y_i^n \wedge f(u^k)_i} \\ &= \epsilon_0^{\sum_{i=1}^n y_i^n \wedge \overline{f(u^k)_i}} (1-\epsilon_0)^{\sum_{i=1}^n \overline{y_i^n} \wedge \overline{f(u^k)_i}} \epsilon_1^{\sum_{i=1}^n \overline{y_i^n} \wedge f(u^k)_i} (1-\epsilon_1)^{\sum_{i=1}^n y_i^n \wedge f(u^k)_i}. \end{aligned}$$

To simplify our notation, we introduce the following elements:

$$\begin{aligned} d_{01}\left(y^n, f(u^k)\right) &= \sum_{i=1}^{n} \overline{y_i^n} \wedge f(u^k)_i \\ d_{10}\left(y^n, f(u^k)\right) &= \sum_{i=1}^{n} y_i^n \wedge \overline{f(u^k)_i} \\ d_{11}\left(y^n, f(u^k)\right) &= \sum_{i=1}^{n} y_i^n \wedge f(u^k)_i, \\ d_{00}\left(y^n, f(u^k)\right) &= \sum_{i=1}^{n} \overline{y_i^n} \wedge \overline{f(u^k)_i} = n - d_{11}\left(y^n, f(u^k)\right) - d_{01}\left(y^n, f(u^k)\right) - d_{10}\left(y^n, f(u^k)\right). \end{aligned} \tag{A.19}$$

The conditional probability of the received vector given the sent codeword can now be written as

$$P_{Y^n|X^n}\left(y^n|f(u^k)\right) = \epsilon_0^{d_{01}\left(y^n, f(u^k)\right)} (1-\epsilon_0)^{d_{00}\left(y^n, f(u^k)\right)} \epsilon_1^{d_{01}\left(y^n, f(u^k)\right)} (1-\epsilon_1)^{d_{11}\left(y^n, f(u^k)\right)},$$

or, which is the same, factoring $(1-\epsilon_0)$,

$$P_{Y^n|X^n}\left(y^n|f(u^k)\right) = \left(\frac{\epsilon_0}{1-\epsilon_0}\right)^{d_{01}\left(y^n, f(u^k)\right)} \left(\frac{\epsilon_1}{1-\epsilon_0}\right)^{d_{10}\left(y^n, f(u^k)\right)} \left(\frac{1-\epsilon_1}{1-\epsilon_0}\right)^{d_{11}\left(y^n, f(u^k)\right)} (1-\epsilon_0)^n.$$

Appendix B

Binary Entropy Function

Let $h_2(\epsilon)$ be the *binary entropy function*, which is defined by:

$$h_2(\epsilon) \triangleq -\epsilon \cdot \log_2(\epsilon) - (1-\epsilon) \cdot \log_2(1-\epsilon), \; for \; \epsilon \in [0,1].$$

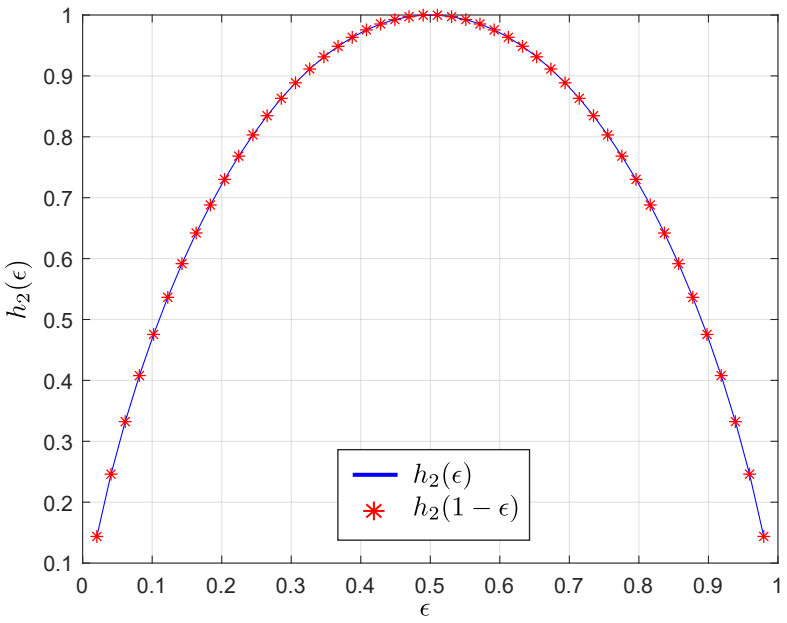


Figure B.1: Binary Entropy

As we can see in figure B.1, this function is symmetric around $\epsilon = 0.5$, thus $h(\epsilon) = h(1-\epsilon) \; for \; \epsilon \in [0,1]$. We can notice that the maximum of this function is found when $\epsilon = 0.5$, because under this condition, the transmitted bit is completely uncertain.